System Identification for Temporal Networks

Sergey Shvydun and Piet Van Mieghem

Abstract—Modelling temporal networks is an open problem that has attracted researchers from a diverse range of fields. Currently, the existing modelling solutions of time-evolving graphs do not allow us to provide an accurate graph sequence. In this paper, we examine the network dynamics from a system identification perspective. We prove that any periodic graph sequence can be accurately modelled as a linear process. We propose two algorithms, called Subspace Graph Generator (SG-gen) and Linear Periodic Graph Generator (LPG-gen), for modelling periodic graph sequences and provide their performance on artificial graph sequences. We further propose a novel model, called Linear Graph Generator (LG-gen), that can be applied to non-periodic graph sequences. Our experiments on artificial and real networks demonstrate that many temporal networks can be accurately approximated by periodic graph sequences.

Index Terms—Temporal Networks, Network Dynamics, System Identification.

1 Introduction

Many real systems such as social, financial, biological and technological systems can be represented as networks, where the nodes are the elements of the system and the links are interactions between them. The power of networks resides in their ability to provide insights into complex system structure and to model complex dynamics such as diffusion and contagion. However, most studies in graph theory are performed under the assumption that the structure of the network is static, which is not true for most real systems. Indeed, in many applications, the systems are dynamic in nature and evolve over time which, in turn, affects their topology and the processes that propagate over the network. Such an observation leads to the fact that many practical problems can be solved more accurately if the time-evolving structure of the graph is taken into account. The ability to model real systems facilitates our understanding of the nature and the timing of observed evolution, but it may also provide some useful intuition about the future behaviour of the system, thereby making valuable predictions. Moreover, understanding the graph evolution can identify system malfunction or security intrusion.

Understanding the evolution of networks is still an open problem. Most papers on network dynamics focus on temporal link prediction using matrix factorization, probabilistic approaches, spectral clustering, time series or deep learning methods that fail to capture global topological features and non-linear varying temporal patterns of the network [1]. Other approaches focus on the graph level and consists in producing generative graph sequences that mimic the real-world networks in terms of certain topological features such as the number of links, clustering coefficient, degree distribution, connected components or motifs [2], [3], [4], [5], [6], [7], [8], [9]. Various approaches have been proposed to model particular dynamical processes such as human mobility [10], [11], [12], [13] or communication networks [14]. These models are mostly probabilistic and activity-driven (nodes and links are active or inactive within some time intervals) or spatiotemporal (nodes are changing their position in space with respect to some trajectory) [15], [16]. For instance, Chang et al. [11] introduce a Markov Modulated Process (MMP), where states of the Markov chain encode certain modifications to the original graph. It is shown that an MMP model captures global graph properties, but it does not provide an accurate topology of the graph and cannot be applied for modelling processes that have seasonality or spikes around certain events. To the best of our knowledge, none of the existing models generates graph sequences that resemble the real graph in terms of its set of adjacency matrices. Moreover, most models do not provide knowledge about the underlying process.

This paper is aimed as one step forward towards a deeper understanding of network evolutionary processes. We examine network dynamics from a system identification perspective. Based on the observed graph sequence, we attempt to model the graph evolution as a linear process. The goal of the study is to find a process that generates such graph sequences accurately.

Many real world systems possess a dynamics that repeats during a certain period of time due to a daily rhythm. Examples of such quasi-periodic systems are road traffic, computer networks, logistics, human mobility, social interactions and many others. Ma and Hellerstein [17] show that periodic patterns often lead to actionable insights about the evolutionary process. We show that any periodic graph evolution can be described by a linear time-invariant process.

Our major contributions can be summarized:

- We model the graph dynamics as a *linear process* using a system identification approach in Section 3.2.
- We prove in Section 3.4 that any periodic graph sequence can be modelled exactly. We provide information about the dimensionality of the system that produces the exact graph dynamics.
- We propose SG-gen, LPG-gen and LG-gen algorithms to model periodic and non-periodic graph sequences (see Sections 3.2, 3.5, 3.6).
- We test SG-gen, LPG-gen and LG-gen algorithms on various artificial and real graph sequences and illustrate their reachable accuracy (Sections 3.7-3.8).

S. Shvydun and P. Van Mieghem are with the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628 CD, Delft, The Netherlands.
E-mail: S.Shvydun; P.F.A.VanMieghem@tudelft.nl.

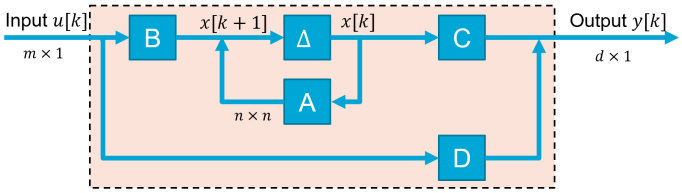

Fig. 1. Visualization of an LTI model. The symbol Δ represents a delay.

The paper is organized as follows. In Section 2, we provide basic information about time-invariant state-space model and subspace methods for system identification. In Section 3, we propose the SG-gen model for graph generation and apply it to artificial graph dynamics. Next, we consider the properties of the SG-gen model and provide the exact solution for periodic graph sequences (LG-gen). We also consider the application of LG-gen to quasi-periodic artificial networks and real graphs. Finally, Section 4 concludes. Additionally, we introduce the notation to the reader in the Appendix A.

2 Linear Time-Invariant State-Space Model

2.1 Problem statement

The dynamics of a linear system in discrete time k is defined by a linear time-invariant (LTI) state-space model [18]:

$$\begin{cases} x[k+1] = A \cdot x[k] + B \cdot u[k], \\ y[k] = C \cdot x[k] + D \cdot u[k], \end{cases} \tag{1}$$

where $u[k] \in \mathbb{R}^m$ is the input vector, $y[k] \in \mathbb{R}^d$ is the output vector, $x[k] \in \mathbb{R}^n$ is the state vector and where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{d\times n}$ and $D \in \mathbb{R}^{d\times m}$ are time-invariant matrices that define the relation between input, output and state vectors. The dimension n of the state vector $x[k]$ defines the order of an LTI system. Equivalently, (1) is

$$\begin{bmatrix} x[k+1] \\ y[k] \end{bmatrix} = Q \cdot \begin{bmatrix} x[k] \\ u[k] \end{bmatrix}, \quad \text{with } Q = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{2}$$

The block matrix Q is an $(d+n)\times(n+m)$ system matrix. The scheme of an LTI system from [19] is represented in Fig. 1.

In general, as shown by Verhaegen and Verdult [18], there are different state representations that yield the same dynamic relation between observations $u[k]$ and $y[k]$. In fact, any system

$$\begin{cases} x_p[k+1] = A_p \cdot x_p[k] + B_p \cdot u[k], \\ y[k] = C_p \cdot x_p[k] + D_p \cdot u[k], \end{cases}$$

with $A_p = P^{-1}AP$, $B_p = P^{-1}B$, $C_p = CP$, $D_p = D$ and $x_p[k] = P^{-1}x[k]$ for any non-singular similarity matrix P is an equivalent system to (1) since it produces the same output $y[k]$ given the same input $u[k]$ and merely relabels their vector components. Therefore, it is only possible to identify the matrices A, B, C, D up to a similarity (or state) transformation P and any LTI system with system matrices (A_p, B_p, C_p, D_p) and state vector x_p is equivalent to (1).

The identification of an LTI state-space model in (1) from input and output measurements can be solved using subspace identification methods [20]. The subspace methods are based on the fact that, by storing the input and output data in structured block Hankel matrices, it enables to retrieve certain vector subspaces that are related to the system matrix Q of an LTI model. More precisely, the input and output measurements can be stored in structured block Hankel matrices as follows:

$$Y_{i,s,h} = \begin{bmatrix} y[i] & y[i+1] & \cdots & y[i+h-1] \\ y[i+1] & y[i+2] & \cdots & y[i+h] \\ \vdots & \vdots & \ddots & \vdots \\ y[i+s-1] & y[i+s] & \cdots & y[i+h+s-2] \end{bmatrix},$$

where $Y_{i,s,h} \in \mathbb{R}^{sd\times h}$, h and s are parameters of an LTI model (in general, $s \ll h$). The blocked Hankel matrix $U_{i,s,h} \in \mathbb{R}^{sm\times h}$ constructed from $u[t]$ is defined in a similar way. Block Hankel matrices allow us to rewrite (1) as

$$Y_{i,s,h} = \Gamma_s \cdot X_{i,h} + H_s \cdot U_{i,s,h}, \tag{3}$$

where $X_{i,h} = \begin{bmatrix} x[i] & x[i+1] & \cdots & x[i+h-1] \end{bmatrix} \in \mathbb{R}^{n\times h}$, $\Gamma_s = \begin{bmatrix} C & CA & \dots & CA^{s-1} \end{bmatrix}^T \in \mathbb{R}^{sd\times n}$ and $H_s \in \mathbb{R}^{sd\times sm}$ are correspondingly the extended observability and the block Toeplitz matrices derived from A, B, C, D:

$$H_s = \begin{bmatrix} D & 0 & \cdots & 0 \\ CB & D & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{s-2}B & CA^{s-1}B & \cdots & D \end{bmatrix}.$$

Several solutions of (3) have been presented in [19], [20], [21], [22], [23], [24]. These methods express the row space of $Y_{i,s,h}$ as a linear combination of row spaces of $X_{i,h}$ and $U_{i,s,h}$. Subspace methods benefit from reliable numerical algorithms such as LQ decomposition and the singular value decomposition (SVD), which are non-iterative and do not need nonlinear optimization techniques. One of the most prevailing algorithms for subspace system identification is a numerical algorithm for subspace state space system identification (N4SID) [24], which is described in Appendix B. The N4SID algorithm can be extended to identify an LTI system corrupted by process and measurement noise [18].

3 Application of the LTI Model to Temporal Networks

3.1 Preliminaries

We consider a temporal graph, denoted by $G_k(\mathcal{N},\mathcal{L}_k)$, consisting of a set $\mathcal{N}$ of N nodes (vertices) connected by a set $\mathcal{L}_k$ of L_k links (edges) at discrete time k. The graph G_k is described by an $N \times N$ adjacency matrix A_k whose elements $a_{ij}(k)$ are either one or zero depending on whether there is a link between nodes i and j or not. For simplicity, we assume that the set of nodes is fixed and the graph is undirected, then all adjacency matrices $A_k = A_k^T$ are real symmetric matrices. Additionally, we denote by $\mathcal{L} = \bigcup_{k=1}^{T} \mathcal{L}_k$ the union of all L links that emerged in graphs over T time slots with $L \leq \binom{N}{2}$. The problem lies in identifying the underlying process that generates the sequence $G_1, G_2, \dots, G_T$ of graphs over T time slots.

3.2 Subspace model for temporal networks

We assume that the graph dynamic is described by an LTI state-space model. The $L \times 1$ binary vector $a[k]$ specifies[1]

1. The vector $a[k]$ does not preserve information about the position of links in G_k, except if we choose its dimension equal to $\binom{N}{2}$, in which case a component $a_i[k]$ corresponds to an element $(A_k)_{pr}$ in the adjacency matrix.

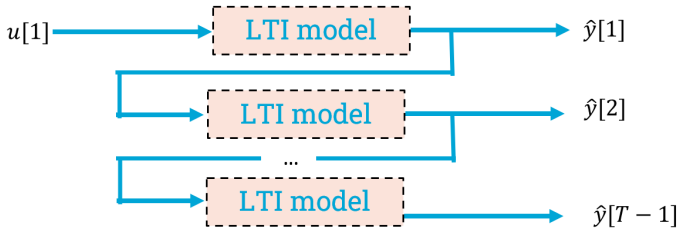

Fig. 2. Output generation using SG-gen model.

the links in graph G_k at discrete time k with component $a_i[k] = 1$ if the i-th link of the set $\mathcal{L}$ is present in graph G_k, otherwise $a_i[k] = 0$. Then we can construct $T-1$ input-output sequence $\{(u[k], y[k])\}_{k=1}^{T-1}$, where $u[k] = a[k]$ and $y[k] = a[k+1]$. In other words, the input of the LTI model at discrete time k is the structure of graph G_k and the output is the structure of graph G_{k+1}. The N4SID algorithm estimates the system matrix Q that best approximates the graph G_{k+1} given the real graph G_k.

Here, we propose a novel model, called *Subspace Graph Generator (SG-gen)*, that generates the whole graph sequence using the identified LTI model. The SG-gen model assumes that the estimated output vector $\widehat{y}[k]$ at discrete time k equals the input vector $\widehat{u}[k+1]$ at discrete time $k+1$, i.e., $\widehat{y}[k] = u[k+1]$. The governing equation (2) simplifies to

$$\begin{bmatrix} x[k+1] \\ \widehat{y}[k] \end{bmatrix} = \begin{bmatrix} x[k+1] \\ u[k+1] \end{bmatrix} = Q \cdot \begin{bmatrix} x[k] \\ u[k] \end{bmatrix}, \tag{4}$$

where Q is an $(n+L)\times(n+L)$ system matrix. If we denote the block vector $v[k] = \begin{bmatrix} x[k] \\ u[k] \end{bmatrix}$, then the law (4) of SG-gen is

$$v[k+1] = Q \cdot v[k], \tag{5}$$

whose solution follows by iteration as

$$v[k+1] = Q^k \cdot v[1]. \tag{6}$$

The output sequence $\widehat{y}[1], \ldots, \widehat{y}[T-1]$ corresponding to the graphs $\widehat{G}_2, \ldots, \widehat{G}_T$ can be computed via (6). The SG-gen model is visualized in Fig. 2.

The rationale behind the SG-gen model is that if the LTI model accurately identifies the structure of graph G_{k+1} from graph G_k, it can also generate the whole graph sequence $G_2, \ldots, G_T$ using the initial structure of the graph G_1.

The SG-gen model has some limitations. SG-gen is not identical to a classical LTI system, which only makes a 1-step prediction and does not use the estimated output $\widehat{y}[k]$ as the input for the next time slot $k+1$. On the contrary, SG-gen relates to an autonomous system, does not require any external inputs after the 1st time slot and may produce infinite graph sequences. Finally, since the SG-gen model is based on the N4SID algorithm, it requires input parameters (the size of the block Hankel matrix) and should satisfy all assumptions of N4SID.

Property of the SG-gen linear system. If the $q \times q$ system matrix Q with $q = n + L$ has q distinct eigenvalues $\lambda_1, \ldots, \lambda_q$, then the matrix Q can be written [25] as

$$Q = X \Lambda Y^T,$$

where the matrices X and Y contain the right- and left-eigenvectors of Q, respectively, in their columns and obey $XY^T = I$ and $\Lambda = diag(\lambda_i)$ is a diagonal matrix of eigenvalues of Q. Since $Q^k = X\Lambda^k Y^T$ and applied to (6), it follows that

a) $v[k] \to 0$ for $k \to \infty$ iff $|\lambda_i| < 1, \forall i = 1, \ldots, q$.
b) $v[k] \to \infty$ for $k \to \infty$ if $\exists i$ such that $|\lambda_i| > 1$.

Hence, this property of SG-gen constitutes the main limitation, because not all graph dynamics can be generated by SG-gen. Indeed, if there exists at least one eigenvalue larger than 1, then the long-term evolution of SG-gen will tend to infinity and the SG-gen model becomes useless. If all eigenvalues are less than 1, the SG-gen model will eventually generate empty graphs, which are not suitable for real-world networks. Therefore, SG-gen imposes strict requirements on the graph dynamics, namely $|\lambda_i| = 1$ for all $1 \le i \le q$, and only periodic graph sequences can be modelled by SG-gen (see Section 3.4).

Finally, we assess the performance of SG-gen by the *mean square error (MSE)*

$$MSE(y, \widehat{y}) = \frac{1}{T-1} \sum_{k=1}^{T-1} \sum_{i=1}^{L} (y_i[k] - \widehat{y}_i[k])^2, \tag{7}$$

where $y[k]$ and $\widehat{y}[k]$ are the real and estimated vectors corresponding to the graph G_{k+1} at time $k+1$. Thus, the accuracy of SG-gen is estimated based on all output measurements.

3.3 Experiments on periodic graph sequences

We apply the SG-gen model to various periodic graph sequences. We assume that the temporal graph has a period p; thus, $G_k = G_{k+p}$ for $\forall k = 1, \ldots, T-p$. We begin with periodic sequences, where the graph G_{k+1} is constructed from graph G_k by adding or removing only one link. Next, we consider more comprehensive periodic sequences, where we make multiple random changes or none in graph G_k to generate the next graph G_{k+1}.

- *Graph dynamic №1:* we arbitrary enumerate all possible links $L = N(N-1)/2$ in the graph and add 1 link per time slot to the graph in the prescribed order. Once the graph G_k equals the complete graph K_N, precisely one link is removed per time slot in the reversed order. Once the graph is empty, we repeat the graph generation process. The described process has a period $p = 2L$. The SG-gen model has been tested on networks containing up to 25 nodes.
- *Graph dynamic №2:* the dynamical process is the same as for the graph dynamic №1, but we also allow the graph to keep its structure unchanged for some time slots. For instance, we assume that the graph does not change for 6 time slots if it is empty, for 4 time slots if it is complete and for 2 time slots if it contains half the number of possible links. The period of the process is $p = 2L + 12$. The SG-gen model has been tested on networks containing up to 10 nodes.
- *Graph dynamic №3:* we start with an arbitrary graph G_1, then produce up to 1 random change in a graph structure per time slot but after p time slots, it becomes identical to G_1. The period of the process is $p = 2L$. The SG-gen model has been applied to graphs with up to 11 nodes while the MSE is averaged across 500 iterations.

A detailed information about the experiments on periodic data is provided in the Appendix C. Overall, the SG-gen model almost exactly predicts the graphs sequence $G_2, \ldots, G_T$ from graph G_1 for all graph dynamics, i.e., $MSE\,(y, \widehat{y}) \approx 0$. Moreover, we observe that the choice of the initial graph G_1 does not affect the results of the experiment. The experiments indicate that the minimal number of periods to identify the system matrix Q with accurate predictions does not exceed 3 for the graph dynamic №1, 9 for the graph dynamic №2 and 7 for the graph dynamic №3. The sequence of state vectors $x[1], \ldots x[T-1]$ has the same period p.

Next, we examine how the dimension n of the state vector $x[k]$ and the order of the system matrix Q depend upon the number L of links in a temporal graph. For the graph dynamic №1, the state vector has L coordinates and Q is a $p \times p$ matrix. For the graph dynamic №2, the dimension of the state vector is $n = p - L - 1$ and Q is a $(p-1) \times (p-1)$ matrix. For the graph dynamic №3, the dimension n of the state vector satisfies $|p - L - n| \leq 1$ while the order of Q is either p or $p-1$.

The experiments in Appendix C show that SG-gen generates many periodic graph sequences accurately. However, we have not yet proved that SG-gen can model *all* periodic graph sequences, because the SG-gen model has limitations. First, the identification of the system matrix Q is constructed based on a 1-step prediction from the real input. Thus, there is no guarantee that the identified matrix Q will accurately generate $G_2, \ldots, G_T$ sequence from G_1. Second, since the SG-gen model is based on the N4SID algorithm, the performance of SG-gen depends on the choice of input parameters such as the number of observations in the dataset and the number of rows in block Hankel matrices. In general, there is no prior information about these parameters, consequently, we determine them experimentally by finding the values that minimize the MSE in (7). In Section 3.4, we prove that all periodic graph sequences can be modelled by SG-gen and we propose the LPG-gen algorithm that identifies accurately the system matrix Q without subspace methods.

3.4 Properties of a Linear System for Periodic Data

First, we analyse the $(n+L) \times (n+L)$ system matrix Q. Since the vector state $x[k]$ has a period p, it holds that $v[k+p] = v[k]$, for $\forall k = 1, \ldots, T-p$. Combined with $v[k+p] = Q^p \cdot v[k]$ in (6), the matrix Q satisfies $Q^p = I$, where I is the identity matrix. The eigenvalue equation $Q \cdot z = \lambda z$, where λ is an eigenvalue of Q belonging to eigenvector z, is equivalent to $Q^p \cdot z = \lambda^p z$ for non-negative integer p and leads, with $Q^p = I$, to $\lambda^p - 1 = 0$, whose solution is $\lambda = e^{2\pi i m/p}$ where $m = 0,\ 1, \ldots, p-1$. In other words, all $p = n + L$ eigenvalues of the matrix Q are unique and evenly spaced around the unit circle in the complex plane with $\lambda = 1$ as real eigenvalue corresponding to $m = 0$. Only if p is even, then $\lambda = -1$ is the other possible real eigenvalue for $m = p/2$.

The *(k+1)*-th output of SG-gen is given by $v[k+1] = Q^k v[1]$ in (6), where $v[1]$ is a concatenation of vectors $x[1]$ and $u[1]$. In general, vector $u[1]$ is an $L \times 1$ vector that corresponds to the graph G_1, while the $n \times 1$ vector $x[1]$ is unknown. Thus, we need to determine the dimension n as well as the values of the vector $x[1]$.

Lemma 1. *If the $(n+L) \times 1$ vectors sequence $v[1], \ldots, v[T]$ has a period p, then $v[k+1] = Q^k v[1]$ in (6) implies that*

a) *$Q^p = I$. The matrix Q is called $(p+1)$-potent [26];*
b) *All vectors $v[k]$ with $k = 1, \ldots, T$ are eigenvectors of matrix $Q^p = I$, because $v[k+p] = Q^p \cdot v[k] = v[k]$.*

Since the vectors $v[1], \ldots, v[T]$ are periodic, it is sufficient to consider only first p vectors during 1 period. We rewrite (5) as

$$\begin{cases} v[k+1] = Q \cdot v[k], \ \ \forall k = 1, \ldots, p-1, \\ v[p+1] = Q \cdot v[p] = v[1], \end{cases}$$

or, equivalently

$$[v[2] \ \ldots v[p] \ \ v[1]] = Q \cdot [v[1] \ \ldots \ v[p-1] \ \ v[p]]. \quad (8)$$

We denote the two $(n+L) \times p$ matrices $V_1 = [v[1] \ \ldots v[p-1] \ \ v[p]]$ and $V_2 = [v[2] \ \ldots v[p] \ \ v[1]]$ such that (8) becomes $V_2 = Q \cdot V_1$.

Lemma 2. *Let $v[1], \ldots, v[p]]$ be a linearly independent set of $p \times 1$ vectors or, equivalently, the matrices V_1 and V_2 are of full rank. Then, there exists a unique matrix Q satisfying (8), which is defined by*

$$Q = V_2 \cdot V_1^{-1}. \quad (9)$$

Relation (9) is crucial for periodic graph sequences. Indeed, if vectors $v[1], \ldots, v[p]$ are linearly independent in p-dimensional vector space, any periodic graph sequence can be accurately modelled by the system matrix Q. This result explains our previous findings in Section 3.3, where the order of the system matrix Q is exactly p and the dimension n of the state vector $x[k]$ is $p - L$. Thus, p is a sufficient dimension for the SG-gen model to exactly determine the output of the periodic sequence. In fact, the identification of matrix Q can be performed by constructing the $n \times 1$ vectors $x[1], \ldots, x[p]$ such that the vectors $v[1], \ldots, v[p]$ will be linearly independent. The system matrix Q can be computed using (9). Consequently, the N4SID algorithm is not required for periodic sequences. In general, it is not evident that p is a minimal order of Q. Experiments show that some graph dynamics can be accurately generated by mapping vectors $v[1], \ldots, v[p]$ into an r-dimensional vector space with $r < p$. Theorem 3 determines the minimal order of matrix Q.

We denote by $C(V)$ the $(n+L) \times p$ *circulant* matrix of the vectors $V = \{v[1], \ldots, v[p]\}$,

$$C(V) = \begin{bmatrix} v[1] & v[2] & \cdots & v[p-1] & v[p] \\ v[2] & v[3] & \cdots & v[p] & v[1] \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ v[p] & v[1] & \cdots & v[p-2] & v[p-1] \end{bmatrix}.$$

Theorem 3. *Let $V = \{v[1], \ldots, v[p]\}$ be a set of $r \times 1$ vectors with $r \leq p$. If there exists a non-zero matrix Q that satisfies (8), then the minimal order r of the system matrix Q is defined as*

$$r = rank(C(U)),$$

where $C(U)$ is a $(p \cdot L) \times p$ circulant matrix of input vectors $U = \{u[1], \ldots, u[p]\}$ corresponding to the graphs $G_1, \ldots, G_p$. The $r \times 1$ vectors $v[1], \ldots, v[p]$ must satisfy

$$C(V) \cdot \begin{bmatrix} \alpha_1 \\ \ldots \\ \alpha_p \end{bmatrix} = 0. \quad (10)$$

Theorem 3 implies that $v[1], \ldots, v[p]$ can always be mapped into an r-dimensional vector space with $r \leq p$. The rows of the matrix $C(U)$ contains all the dynamics of links within p time slots, hence, we call the rank of $C(U)$ the *complexity* of the periodic process. By introducing vectors $x[1], \ldots, x[p]$, satisfying (10), we can uniquely identify the system matrix Q.

Theorem 4. *Let $V = \{v[1], \ldots, v[p]\}$ be a set of $r \times 1$ vectors that satisfies (10) and $r \leq p$. Then, there exists a unique matrix Q, satisfying (8),*

$$Q = [v[k_1 + 1] \ \ldots \ v[k_r + 1]] \cdot [v[k_1] \ \ldots \ v[k_r]]^{-1}, \quad (11)$$

where $v[k_1], \ldots \ v[k_r]$ are r linearly independent vectors.

Theorems 3-4 are proved in Appendix D.

3.5 Linear Periodic Graph Generator (LPG-gen)

Theorems 3-4 are the basis for a general algorithm that identifies the system matrix Q for periodic graph sequences. Moreover, the rows of the matrix $C(U)$ from Theorem 3, which are linearly independent from the first row space of matrix $[u[1], \ldots, \ u[p]]$, can used as state vectors. Algorithm 1 proposes the *Linear Periodic Graph Generator* (LPG-gen) that produces accurately any periodic graph sequence:

Algorithm 1 Linear Periodic Graph Generator (LPG-gen)

Input: vectors $u[1], \ldots, u[T]$ corresponding to $G_1, \ldots, G_T$.
Output: system matrix Q and initial state vector $x[1]$.

1. Define the minimal period p such that $u[k] = u[k+p]$, for $\forall k = 1, \ldots, T - p$.
2. Solve $\alpha_1, \ldots, \alpha_p$ in the system
$$C(U) \cdot \begin{bmatrix} \alpha_1 \\ ... \\ \alpha_p \end{bmatrix} = 0. \quad (12)$$
The order r of the matrix Q equals the rank $C(U)$. Define the pivot variables $\alpha_{k_1}, \ldots, \alpha_{k_r}$ from (12).
3. Initialize the state vectors $x[k_1], \ldots, x[k_r]$ to make vectors $v[k_1], \ldots, v[k_r]$ linearly independent. The remaining $p - r$ state vectors $x[k]$ with $k = 1, ..., p$ and $k \notin \{k_1, \ldots, k_r\}$ are initialized with respect to (10).
4. Compute the system matrix Q via (11).

Return: $Q, \ x[1]$.

Similarly to the SG-gen model, LPG-gen is designed only for periodic graph sequences. However, LPG-gen ensures that any periodic graph sequences can be modelled accurately and the order r of the matrix Q is minimal. Additionally, LPG-gen does not require input parameters.

The computational complexity of LPG-gen. Step 1 requires at most $\frac{L \cdot p \cdot T}{2}$ operations. Steps 2-3 takes at most $2Lp^3$ operations [27]. Finally, the worst-case running time of step 4 is $2p^3$. Therefore, the estimated computational complexity of LPG-gen is $O(LpT + Lp^3)$. Experimentally, LPG-gen has a lower runtime than SG-gen (see the Appendix E).

3.6 Linear Graph Generator (LG-gen)

In previous Sections, we have discussed only perfect periodic graph sequences that obey $G_k = G_{k+p}$ for any discrete time $k = 1, \ldots, T - p$ and period $p < T$. Most real-world systems are not perfectly periodic, hence, the graphs G_k and G_{k+p} are not exactly the same. Therefore, we extend the LPG-gen model to non-periodic graph sequences.

Since LPG-gen accurately produces any periodic graph sequence, it serves as a basis for its extension. We borrow a powerful idea from Fourier analysis that has found application in many disciplines: most signals can be decomposed into periodic waveforms. Zygmund [28, Chapter 17] treats Fouries series in the m-dimensional Euclidean space and states that the extension of a single variable $m = 1$ to the case of several variables $m > 1$ is 'generally' the same.

We apply the *periodicity transform* (PT) that decomposes signals into basic periodic components by projecting onto a set of periodic subspaces [29]. More precisely, the periodicity transform iteratively defines the closest p-periodic vector to the initial vector and then applies the PT to the residuals. Contrary to the Fourier transform, periodic subspaces are not orthogonal, hence, the periodicity transform does not in general provide a unique representation. However, Sethares and Staley [29] show that, in many cases, the periodicity transform provides a clearer explanation of the underlying nature of the signals than the Fourier transform.

The *Linear Graph Generator* (LG-gen) approximates the real graph sequence $G_1, \ldots, G_T$ by a set of periodic graph sequences. We denote by $G_1^{(i)}, \ldots, G_T^{(i)}$ an i-th periodic graph sequence satisfying $G_k^{(i)} = G_{k+p_i}^{(i)}$ for any $k = 1, T - p_i$ with period p_i. Then, we approximate a graph G_k at discrete time k by l periodic graph sequences:

$$G_k \approx \sum_{i=1}^{l} G_k^{(i)}. \quad (13)$$

The rationale behind the LG-gen model is that many non-periodic graph evolutions may contain periodic patterns that describe certain dynamical processes in the network. The observed dynamics with periodic patterns can be approximated by (13), while each periodic sequence $G_1^{(i)}, \ldots, G_T^{(i)}$ for $1 \leq k \leq T$ is computed by LPG-gen.

The LG-gen model is proposed in Algorithm 2. First, LG-gen identifies the optimal period length p_1 and periodic graph sequence $G_1^{(1)}, \ldots, G_T^{(1)}$ that best approximates the initial graph sequence $G_1, \ldots, G_T$. The obtained periodic graph sequence is computed by LPG-gen. Next, we construct the residual graph sequence $G_1 - G_1^{(1)}, \ldots, G_T - G_T^{(1)}$ which corresponds to information about graph structure that is not captured by the periodic graph sequence $G_1^{(1)}, \ldots, G_T^{(1)}$. Next, the same procedure is iteratively applied to estimate the residual sequence. The algorithm continues until l periodic graph sequences are obtained. If the residual sequence $G_1 - \sum_i G_1^{(i)}, \ldots, G_T - \sum_i G_T^{(i)}$ is an empty graph sequence, then $G_k = \sum_{i=1}^{l} G_k^{(i)}$ is exact for a finite l. Algorithm 2 can be also applied to weighted and directed networks.

An important part of LG-gen is the identification of periodic patterns in temporal networks. Andres et al. [30] identify periodic time scales by computing the Fourier transform of the function that measures the portrait divergence between successive temporal networks. Here, we apply the Algorithm 3, called *LocalMin*, that iteratively defines the optimal period p^* based on the vectors $u[1], \ldots, u[T]$. First,

Algorithm 2 Linear Graph Generator (LG-gen)

Input: vectors $u[1], \dots, u[T]$ corresponding to $G_1, \dots, G_T$, number of periodic graph sequences l.
Output: system matrices $Q_1, \dots, Q_l$ and initial state vectors $x_1[1], \dots, x_l[1]$.

1. $u \leftarrow [u[1], \dots, u[T]]$
2. for $i \leftarrow 1$ to l
   a) Determine the period p_i for u
   $p_i \leftarrow$ LocalMin(u).
   b) Construct the average sequence with period p_i
   $\bar{u} \leftarrow [\bar{u}[1], \dots, \bar{u}[T]]$.
   c) Identify the system that produces $\bar{u}$:
   $Q_i, x_i[1] \leftarrow$ LPG-gen($\bar{u}$).
   d) Update $u \leftarrow u - \bar{u}$

Return: $Q_1, \dots, Q_l$ and $x_1[1], \dots, x_l[1]$.

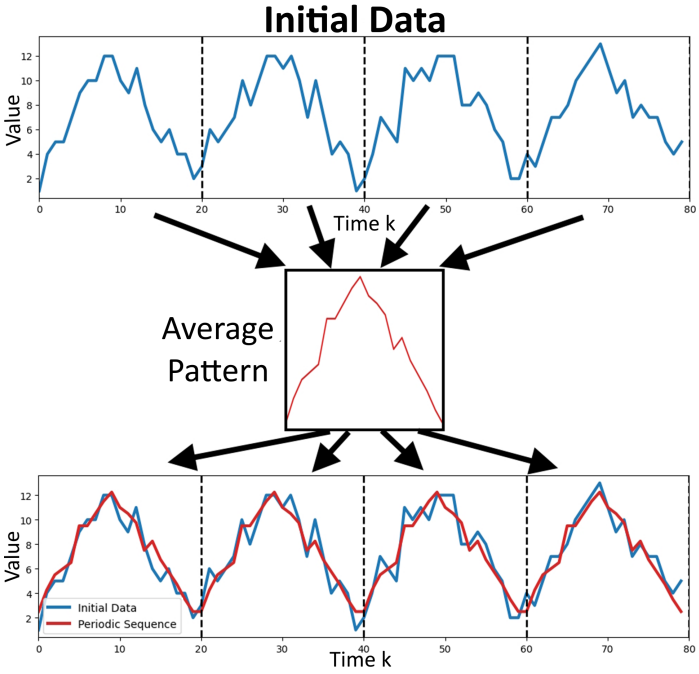

Fig. 3. Identification of period for non-periodic data.

for each possible period $p = 1, \dots, T$, LocalMin constructs the average periodic pattern $\bar{u} = (\bar{u}[1], \dots, \bar{u}[p])$ by

$$\bar{u}[k] = \frac{1}{n_k} \sum_{i=0}^{n_k - 1} u[k + i \cdot p], \tag{14}$$

where $n_k = [(T-k)/p] + 1$ is the total number of vectors $u[1], \dots, u[T]$ that corresponds to the k-th element of the periodic graph sequence $\bar{u}$. Here $[s]$ is the largest integer smaller than or equal to s. The general idea of the average periodic sequence construction is presented in Fig. 3. Second, LocalMin replicates average patterns $\bar{u}$ until T time slots and computes the MSE between $u[1], \dots, u[T]$ and each replicated periodic sequence $\bar{u}[1], \dots, \bar{u}[T]$. Finally, the optimal period p^* is given by

$$p^* = \arg \max_{p \le T/2} \frac{count[p]}{[(T-1)/p] + 1},$$

where $count[p]$ computes the total number of timestamps $k \in \{p, 2p, \dots, ([(T-1)/p] + 1) \cdot p\}$ corresponding to the local minima of the MSE.

The computational complexity of LG-gen is $O(l \cdot L \cdot T^2 + L \cdot \sum_{i=1}^{l} p_i^3)$ where $l \cdot L \cdot T^2$ operations are needed by

Algorithm 3 Identification of periodic patterns (LocalMin)

Input: vectors $u[1], \dots, u[T]$ corresponding to $G_1, \dots, G_T$
Output: period p^*

$loss \leftarrow 0_{T \times 1}$
for $p \leftarrow 1$ to T
  a) Construct the average periodic sequence by (14)
  $\bar{u} \leftarrow [\bar{u}[1], \dots, \bar{u}[T]]$.
  b) $loss[p] \leftarrow mse(u, \bar{u})$
$count \leftarrow 0_{T/2 \times 1}$
for $p \leftarrow$ to $T/2$
  $j \leftarrow p$
  while $j < T$
    if $loss[j] < loss[j-1]$ and $loss[j] < loss[j+1]$
      $count[p] \leftarrow count[p] + 1$
    $j \leftarrow j + p$
  $count[p] \leftarrow \frac{count[p]}{[(T-1)/p]+1}$
$p^* \leftarrow \arg\max_p (count[p])$

Return: p^*.

the LocalMin algorithm. An important feature of LocalMin is its convergence. Theorem 5 guarantees the decrease in the residual values of the remaining graph sequence $G_1 - \sum_i G_1^{(i)}, \dots, G_T - \sum_i G_T^{(i)}$. We also prove that the periodic sequence based on the average provides the largest decrease of the MSE. Sethares amd Staley [29] have shown that the average periodic pattern is an orthogonal projection of the input vector onto the p-periodic subspace.

Theorem 5. *Let $u = \{u[1], \dots, u[T]\}$ be a set of T vectors with dimension L corresponding to $G_1, \dots, G_T$. For any non-zero average sequence $\bar{u} = \{\bar{u}[1], \dots, \bar{u}[T]\}$ with an arbitrary period $p \le T$, it holds that*

a) $\|u - \bar{u}\| \le \|u\|$,
b) $\|u - \bar{u}\| < \|u - \tilde{u}\|$.

where $\tilde{u} = \{\tilde{u}[1], \dots, \tilde{u}[T]\}$ is any sequence of period p such that $\bar{u} \neq \tilde{u}$.

The proof of Theorems 5 is deferred to Appendix F. Additionally, Appendix F.3 examines the minimal period p to obtain a non-zero average periodic sequence and discusses the convergence of LG-gen for arbitrary vector sequences.

3.7 Experiments on non-periodic graph sequences

We consider two graph dynamics that are almost periodic. For simplicity, both dynamics are based on the periodic graph dynamic №1, but the choice of the periodic dynamic does not influence the results of the section.

- *Graph dynamic №4:* we assume that q links evolve periodically with respect to the graph dynamic №1, while the remaining links in the graph G_k evolve randomly at each time slot k (see Fig. 4).
- *Graph dynamic №5:* we consider the graph dynamic №1 and then perform q random changes in the graph structure G_k at each time slot k (see Fig. 5).

The graph dynamic №4 indicates how the performance of the SG-gen and LG-gen models is affected by the presence of random links. The SG-gen model has been tested on graphs with up 20 nodes. SG-gen can produce an exact dynamic for q periodic links if the number of random links is low. For

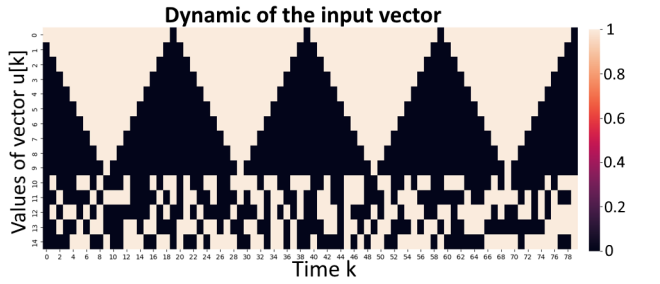

Fig. 4. Graph Dynamic №4 (6 nodes). Among 15 possible links only 10 links have periodic patterns while 5 other links evolve randomly.

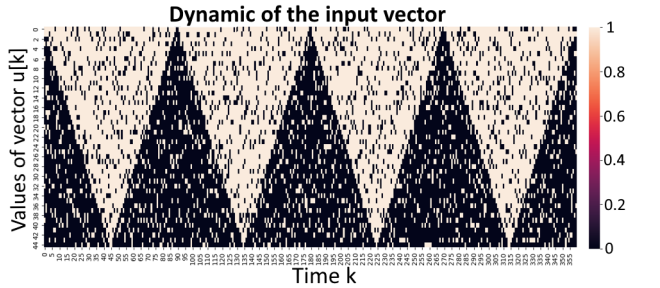

Fig. 5. Graph Dynamic №5 (10 nodes). 8 random changes are performed at each time slot.

a graph dynamic from Fig. 4 the only difference occurs for the last 5 links ($MSE \approx 0.93$). However, the increase of the number of random links results in the inability to identify periodic behaviour for q links by the SG-model.

Fig. 6 illustrates the results of LG-gen on the graph from Fig. 4 for $l = 1$ and $l = 3$. LG-gen for $l = 1$ is comparable to SG-gen. However, if the dynamic is approximated by 3 periodic graph sequences, LG-gen provides a much better performance ($MSE \approx 0.59$). Overall, the increase of parameter l leads to a more accurate graph identification (e.g.: $MSE \approx 0.2$ for $l = 8$). The first periodic graph sequence captures the periodic behaviour of q links while other periodic graph sequences model the dynamics for the last 5 links. Finally, LG-gen still captures periodic patterns for q links even if 85% of links evolve at random.

The SG-gen model was applied to the graph dynamic №5 and did not provide an accurate performance ($MSE \approx 3.92$). SG-gen provides an empty graph after 7 time slots.

Fig. 7 illustrates the results of LG-gen on the graph from Fig. 5 for *l=1* and *l=5*. Similarly to the graph dynamic №4, the first periodic graph sequence of LG-gen approximates the original almost periodic dynamic while additional periodic graph sequences are used to model the remaining changes. The MSE equals 2.08 for *l=1*, 1.02 for *l=5* and 0.43 for *l=10*. Moreover, if we apply the rounding to the sum of generated sequences, 7 periodic graph sequences are sufficient to provide an ideal performance for unweighted

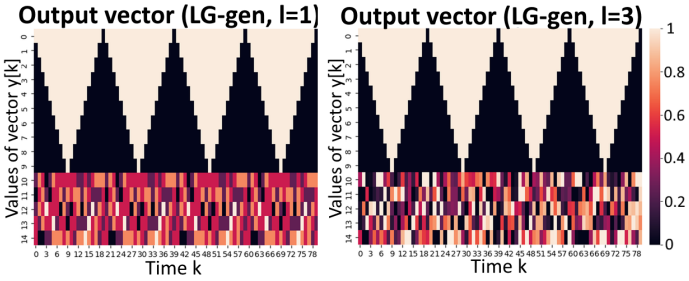

Fig. 6. Output of the LG-gen model for the Graph Dynamic №4 (6 nodes).

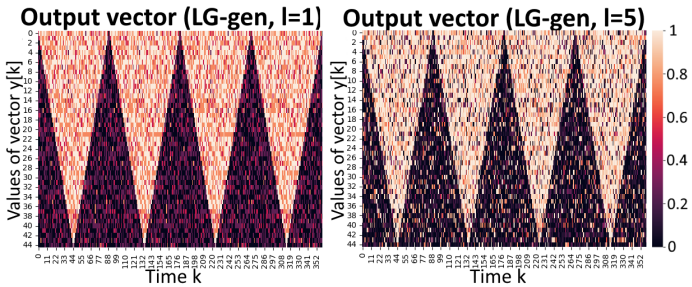

Fig. 7. Output of the LG-gen model for the Graph Dynamic №5 (10 nodes).

TABLE 1
MSE for LyonSchool network.

Model	Number of periodic graphs sequences l								
	1	2	3	4	5	6	7	8	9
No rounding	26.5	20.6	16.4	15.1	13.1	11.8	11.2	10.9	10.7
With rounding	37	16.7	6.8	3.4	1.07	0.3	0.1	0.04	0.01

graph, i.e., $G_k = round\left(\sum_{i=1}^{7} G_k^{(i)}\right)$ for any $k = 1, \ldots, T$, where $round(\cdot)$ denotes for rounding to the nearest integer.

3.8 Experiments on Real Data

We now examine the performance of LG-gen to real networks. We consider several face-to-face interaction networks, collected in a school, in a hospital and in the workspace by the SocioPatterns project[2]. The data were gathered using wearable RFID badges, which assess the proximity of two individuals with a probability in excess of 99% over an interval of 20 seconds [31]. We show that real networks can be accurately modelled by the LG-gen model.

1. A school in Lyon. The LyonSchool dataset [32] contains the contact events between 242 individuals (232 children and 10 teachers) in a primary school in Lyon, France, during two days in October 2009. Each time slot of the network corresponds to a 20-second interval $[t - 20s, t]$ while the connections between nodes represent active contacts during the interval. All the contacts have occurred between 10:30 and 19:20. Thus, total number of time slots is 3,180. The total number of links is 6,594,492 with approximately 2,110 connections per time slot. The total number of unique links in the graph is 26,594. The average number of changes between two adjacent graphs G_k and G_{k+1} is 1,245.

Intuitively, since the contact data is based on a two-days period, we assume that the period should not exceed 1 day. Thus, we test LG-gen with 1,590 time slots (≈8 hours 50 minutes) as an upper bound of the period. The initial MSE between LyonSchool and a zero graph sequence is 43.7.

The results of the LG-gen model are presented in Table 1. Additionally, Fig. 8 illustrated the performance of LG-gen in terms of the dynamics of the link count and the MSE, which is computed between original and generated graph sequences via (7). First, the LG-gen model captures around 30% of links if it approximates the real dynamics using only 1 periodic graph sequence. Second, we observe that the rounding provides a lower MSE for any $l \geq 2$. Fig. 9 illustrates that the MSE decreases exponentially if the rounding is applied at the final stage of LG-gen. For $l = 5$, the difference between LG-gen and real network is about

2. http://www.sociopatterns.org/

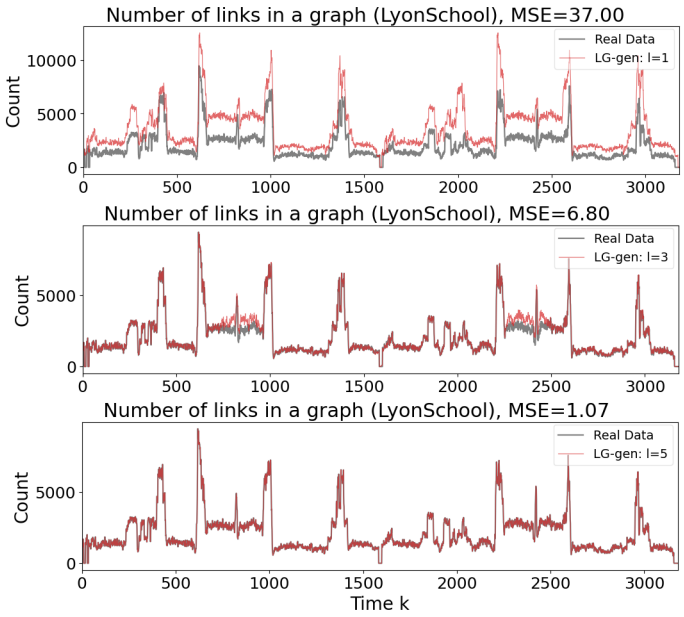

Fig. 8. Performance of LG-gen (with rounding) on LyonSchool.

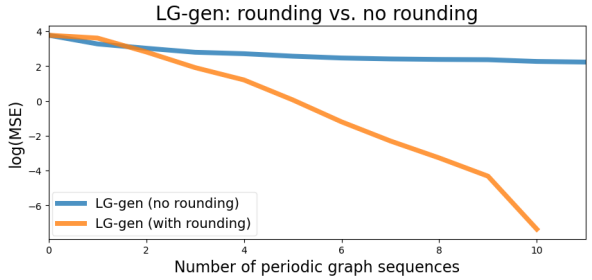

Fig. 9. The logarithm of the MSE on LyonSchool network.

1.15 links in average. The period of 4 graph sequences is in the range from 1,488 to 1,590 time slots (≈1 day cycle), the remaining period is 819 time slots (≈1/2 day cycle).

Finally, the LG-gen model generates the graph sequence accurately for *l=11*, i.e., $G_k = round(\sum_{i=1}^{11} G_k^{(i)})$, for any $k = 1, \dots, T$. Thus, the LyonSchool networks can be accurately modelled by the LG-gen model.

2. Hospital Data. The dataset [33] contains information about contacts between patients, patients and health-care workers (HCWs) and among HCWs in a hospital ward in Lyon, France over 4 days (from December 6, 2010 at 1:00 pm to December 10, 2020 at 2:00 pm). The total number of nodes is 75. Each time slot of the network corresponds to a 20-second interval $[t-20s, t]$ while connections between nodes represent active contacts during different time intervals. The total number of time slots is 17,396 (9,453 time slots with contacts). The total number of contacts is 32,424. There are 4 peaks in the data that correspond to daily cycles. Therefore, we can assume that the dynamics can be approximated by 1 day periods. The initial MSE between Hospital network and a zero graph sequence is 0.95.

Table 2 demonstrates the performance of LG-gen. The increase of l decrease the MSE and for $l \geq 5$ we obtain almost exact matching. In general, the rounding at the final

TABLE 2
MSE for Hospital network.

Model	**Number of periodic graphs sequences** l								
	1	2	3	4	5	6	7	8	9
No rounding	0.89	0.87	0.78	0.68	0.6	0.55	0.51	0.47	0.43
With rounding	0.95	0.91	0.28	0.06	0.03	0.01	0.005	0.002	0.001

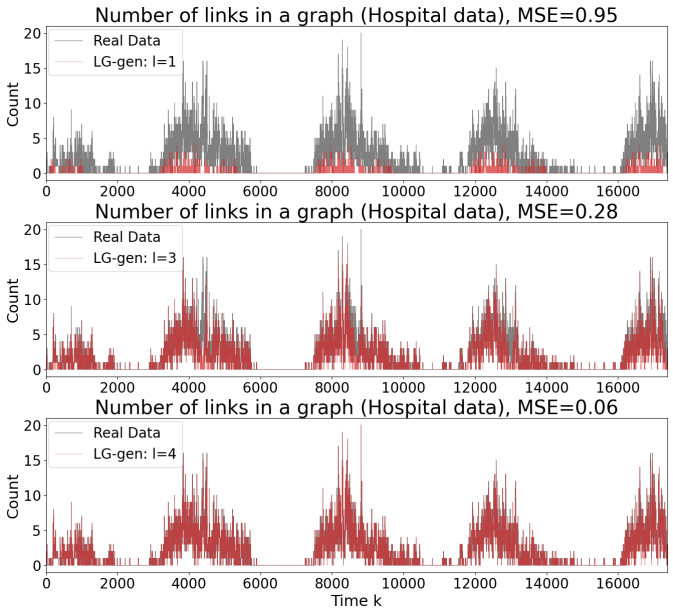

Fig. 10. Performance of LG-gen (with rounding) on Hospital network.

TABLE 3
MSE for InVS13 network.

Model	**Number of periodic graphs sequences** l							
	1	2	3	4	5	6	7	8
No rounding	0.018	0.014	0.012	0.01	0.008	0.007	0.00064	0.0058
With rounding	0.024	0.012	0.007	0.002	0.001	0.0005	0.0004	0

step of LG-gen produces a lower MSE for all l excluding *l=1*. Even for *l=9* the generated graph sequence is not exact, which can be explained by the fact that contacts in hospital have less periodic patterns in general, thus, requiring more periodic sequences to capture such dynamics. Finally, the period of all approximated graph sequences varies from 3950 to 4,320 time slots (≈1 day cycle).

Fig. 10 illustrates information about the number of links in generated graph (after rounding) and in the real data. We observe that for $l = 5$ LG-gen shows a good correspondence with the real Hospital network.

3. InVS13 dataset. The dataset contains the human contact events between 95 individuals in the office building in France in 2013, namely a building of the *Institut de veille sanitaire* [34]. The total number of time slots is 51,120 (20,129 time slots have at least one contact) that correspond to a 2 week period. We consider 10 working days from 9.00 a.m. to 9.00 p.m (23,220 time slots in total). Additionally, we have excluded the first 100 minutes from our experiment, which looked anomalous, as there have been observed around 264 contacts per time slot for the first 300 time slots and only 15.9 contacts per time slot for the remaining time slots. In average, the number of changes between graphs G_k and G_{k+1} is 10.7. We observe a weekly periodicity in the data, thus, we have tested the LG-gen model with 10,805 time slots (5 working days) as an upper bound of the maximum period. The MSE between the workspace network and a zero graph sequence is approximately 0.0245.

The performance of the LG-gen model for $l \leq 8$ is provided in Table 3. Again, we observe that the initial graph can be well approximated by a set of periodic graph sequences. The rounding at the final step provides a lower MSE for all l excluding *l=1*. For *l=5* the MSE is 0.001 which corresponds to approximately 525 link difference in

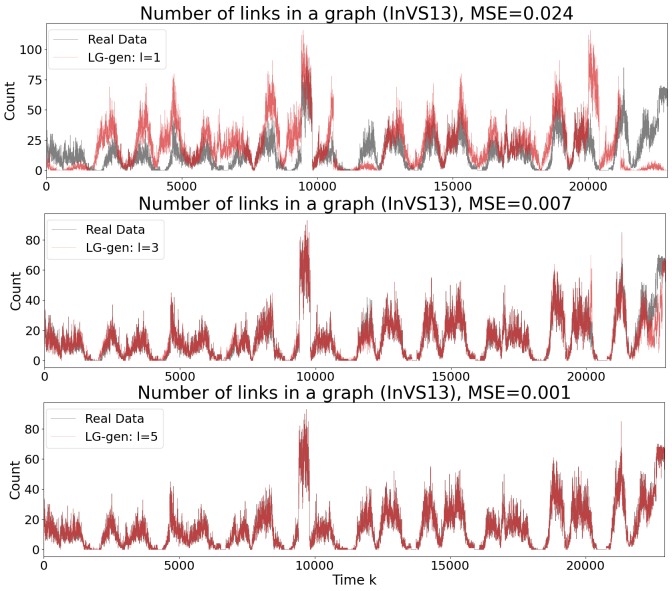

Fig. 11. Performance of LG-gen (with rounding) on InVS13 network.

total during 22,920 time slots. For *l=8* the generated graph sequence is exact if the rounding step is applied. The period of all approximated graph sequences varies from 7,219 (≈3.3 days cycle) to 10,796 time slots (≈5 days cycle). Finally, Fig. 11 demonstrates the difference in the number of changes between original data and LG-gen with rounding.

Overall, the LyonSchool, Hospital and InVS13 networks can be accurately approximated by a small number of periodic graph sequences.

4 Discussion

We have modelled the dynamics of temporal networks as a *linear process* and have proposed 3 algorithms[3] to generate various graph sequences $G_1, \ldots, G_T$ accurately:

1) *Subspace Graph Generator (SG-gen)* for periodic data, which defines the matrix Q by the N4SID algorithm;
2) *Linear Periodic Graph Generator (LPG-gen)* for periodic data, which defines the matrix Q via (11);
3) *Linear Graph Generator (LG-gen)* for non-periodic data, which approximates graph G_k at discrete time k by l periodic graph sequences while each periodic sequence is computed by LPG-gen.

We started with periodic graph sequences and have proved that *any* p-periodic dynamics can be accurately reproduced by SG-gen and LPG-gen. Both algorithms are designed only for periodic dynamics and generate exactly the same graph sequences while LPG-gen has a lower runtime and does not require any additional input parameters. The order of the system matrix Q is defined by the rank of the circulant matrix $C(U)$ of input vectors $U = \{u[1], \ldots, u[p]\}$. Additionally, we propose the rank of $C(U)$ as a measure of the *complexity* of the periodic process.

The LG-gen model has been tested on artificial and real networks. The results of our experiments on artificial data demonstrate a good performance of LG-gen. Since many real systems have periodic patterns, we observe that most of them can be well approximated by the relatively small number of periodic graph sequences.

3. The Python code is available at https://github.com/SergSHV/

The strength of SG-gen, LPG-gen and LG-gen is their simplicity as they are linear. An important issue of LG-gen is the minimal number l of periodic graph sequences that are used for the approximation. We observe that the rounding at the final step may dramatically (about exponentially!) decrease the required value l. In fact, if all residuals are in the interval (-0.5; 0.5), LG-gen does not require any additional steps. Finally, all the algorithms can be applied to directed and weighted networks.

There are some limitations of the LG-gen model. A first concern is the assumption that the graph dynamic is almost periodic. If the process is stable and does not contain any cyclic events, the performance of LG-gen will decrease. However, the presence of periodic behaviour in many real world networks (such as human mobility, social interactions, traffic networks, etc.) makes the LG-gen model quite useful. Another drawback of LG-gen is its high computational complexity. Identification of periodic cycles in the dataset is not a trivial problem, the current implementation requires $l \cdot L \cdot T^2$ operations where L is the number of links in a graph, T is the total number of time slots and l is the number of periodic graph sequences. Periodicity identification is one of the future steps of the research. Moreover, the order of the matrix Q depends on the complexity of the observed dynamics and is limited by the period p, which can be a very large number in general. For instance, if time slots corresponds to minutes, the maximal size of the matrix Q in the case of the weekly periodicity will be around $10^4 \times 10^4$. One way to solve this problem is to decrease the period length of the graph sequence (e.g.: daily cycles instead of weekly cycles) or to decrease the number of time slots (e.g.: time interval is 60 seconds instead of 20 seconds). Finally, SG-gen, LPG-gen and LG-gen assume that the nodes in the network remain the same in each time slot. In real systems, however, both nodes and links are added or removed over time. To remediate this problem, we can consider nodes as isolated during discrete times of inactivity.

We would like to point out that our research presents a conceptual framework showing that periodic graph sequences can be modelled as a linear process while non-periodic graph sequences can be accurately represented as a linear combination of periodic sequences.

Acknowledgments

The authors are grateful to M. Verhaegen, G. Leus and S. Dahlgren for their useful comments and suggestions. P. Van Mieghem is supported by the European Research Council under the European Union's Horizon 2020 research and innovation program (Grant Agreement 101019718).

References

[1] A. Divakaran and A. Mohan, "Temporal Link Prediction: A Survey", *New Gener. Comput.*, 38, 213–258, 2020.
[2] F. Bois and G. Gayraud, "Probabilistic generation of random networks taking into account information on motifs occurrence," *J Comput Biol*: 22(1):25-36, 2015
[3] S. Purohit, L.B. Holder & G. Chin, "Temporal graph generation based on a distribution of temporal motifs", in *Proceedings of the 14th International Workshop on Mining and Learning with Graphs*, 2018.
[4] A. Bojchevski, O. Shchur, D. Zügner and S. Günnemann, "NetGAN: Generating Graphs via Random Walks", in *Proceedings of the 35th International Conference on Machine Learning*, PMLR 80:610-619, 2018.

[5] D. Zhou, L. Zheng, J. Han and J. He, "A Data-Driven Graph Generative Model for Temporal Interaction Networks", in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20),* 401–411, 2020.

[6] G. Zeno, T. La Fond and J. Neville, "Dymond: Dynamic motif-nodes network generative model", in *Proceedings of the Web Conference 2021,* 718–729, 2021.

[7] Y. Du, S. Wang, X. Guo, H. Cao, S. Hu, J. Jiang, A. Varala, A. Angirekula and L. Zhao, "GraphGT: Machine Learning Datasets for Graph Generation and Transformation", in *NeurIPS Datasets and Benchmarks,* 2021.

[8] A. Longa, G. Cencetti, S. Lehmann, A. Passerini and B. Lepri, "Neighbourhood matching creates realistic surrogate temporal networks", *CoRR abs/2205.08820*, 2022.

[9] P. Liu & A. Sariyüce, "Using Motif Transitions for Temporal Graph Generation", in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23),* 1501–1511, 2023.

[10] H. Barbosa, M. Barthelemy, G. Ghoshal, C. James, M. Lenormand, T. Louail, R. Menezes, J. Ramasco, F. Simini and M. Tomasini, "Human mobility: Models and applications," *Physics Reports*, 734, 1-74, 2018.

[11] B. Chang, L. Yang, M. Sensi, M. Achterberg, F. Wang, M. Rinaldi and P. Van Mieghem, "Markov Modulated Process to Model Human Mobility", in *Studies in Computational Intelligence,* 1072, 607–618, 2022.

[12] A. Panisson, A. Barrat, C. Cattuto, W. Van den Broeck, G. Ruffo and R. Schifanella,"On the dynamics of human proximity for data diffusion in ad-hoc networks", *Ad Hoc Networks,* 10(8):1532-1543, 2012.

[13] G. Mauro, M. Luca, A. Longa, B. Lepri and L. Pappalardo, "Generating mobility networks with generative adversarial networks",*EPJ Data Sci.,* 11(58), 2022.

[14] G. Laurent, J. Saramäki & M. Karsai, "From calls to communities: a model for time-varying social networks", *Eur.Phys.J.B,* 88, 301, 2015.

[15] P. Holme, "Modern temporal network theory: a colloquium", *The European Physical Journal B,* 88:1–30, 2001.

[16] D. Ghosh, M. Frasca, A. Rizzo, S. Majhi, S. Rakshit, K. Alfaro-Bittner and S. Boccaletti, "The synchronized dynamics of time-varying networks", *Physics Reports,* 2022, 949, 1-63.

[17] S. Ma and J.L. Hellerstein, "Mining partially periodic event patterns with unknown periods", in *Proceeding of the 2001 international conference on data engineering (ICDE'01),* 205–214, 2001.

[18] M. Verhaegen and V. Verdult, *Filtering and System Identification: A Least Squares Approach,* Cambridge, UK: Cambridge University Press, 2007.

[19] P. Van Overschee and B. De Moor, *Subspace Identification for Linear Systems: Theory, Implementation, Applications,* Kluwer Academic, Dordrecht, The Netherlands, 1996.

[20] B. De Moor, J. Vandewalle, M. Moonen, L. Vandenberghe and P. Van Mieghem, "A Geometrical Strategy for the Identification of State Space Models of Linear Multivariable Systems with Singular Value Decomposition", in *Proceedings of the 8th IFAC/IFORS Symposium on Identification and System Parameter Estimation,* 493-497, 1988.

[21] M. Moonen, B. De Moor, L. Vandenberghe and J. Vandewalle, "On and offline identification of linear state space models", *International Journal of Control,* 49(1): 219-232, 1989.

[22] W. Larimore "Canonical variate analysis in identification, filtering, and adaptive control", in *29th IEEE Conference on Decision and Control,* vol. 2, 596-604, 1990.

[23] M. Verhaegen and P. Dewilde, "Subspace model identification Part 1. The output-error state-space model identification class of algorithms", *International Journal of Control,* 56, 1187-1210, 1992.

[24] P. Van Overschee and B. De Moor, ""N4SID" subspace algorithms for the identification of combined deterministic stochastic system", *Automatica,* 30(1):75–93, 1994.

[25] P. Van Mieghem, *Graph Spectra for Complex Networks,* Cambridge: Cambridge University Press, 2010.

[26] O.M. Baksalary and G. Trenkler, "On k-potent matrices," *Electronic Journal of Linear Algebra*, 26, 446-470, 2013.

[27] S. Boyd and L. Vandenberghe, *Introduction to applied linear algebra: vectors, matrices, and least squares,* Cambridge, UK: Cambridge University Press, 2018.

[28] A. Zygmund, *Trigonometric series,* Vol. I and II (2nd ed.), Cambridge University Press, 1968.

[29] W.A. Sethares and T.W. Staley, "Periodicity transforms," *IEEE Transactions on Signal Processing,* 47(11): 2953-2964, 1999.

[30] E. Andres, A. Barrat, and M. Karsai, "Detecting periodic time scales in temporal networks," *arXiv:2307.03840*, 2023.

[31] J. Stehlé, N. Voirin, A. Barrat, C. Cattuto, L. Isella, J-F. Pinton, M. Quaggiotto, W. Van den Broeck, C. Régis, B. Lina and P. Vanhems, "High-Resolution Measurements of Face-to-Face Contact Patterns in a Primary School," *PLoS ONE,* 6(8): e23176,2011.

[32] M. Génois and A. Barrat, "Can co-location be used as a proxy for face-to-face contacts?", *EPJ Data Sci.*, 7, 11, 2018.

[33] P. Vanhems, A. Barrat, C. Cattuto, J.F. Pinton, N. Khanafer, C. Régis, B. Kim, B. Comte and N. Voirinet N., "Correction: Estimating Potential Infection Transmission Routes in Hospital Wards Using Wearable Proximity Sensors", *PLOS ONE,* 8(9), 2013.

[34] M. Génois, C. Vestergaard, J. Fournet, A. Panisson, I. Bonmarin, and A. Barrat, "Data on face-to-face contacts in an office building suggest a low-cost vaccination strategy based on community linkers", *Network Science,* 3(3), 326-347, 2015.

Sergey Shvydun received the Masters degree in Business Informatics (with distinction, 2014) and PhD in Applied Mathematics (cum laude, 2020) from the HSE University, Moscow, Russia. He is postdoctoral researcher at the Delft University of Technology since 2023. His main research interests include network science, machine learning, operations research, and social choice theory. Before joining Delft, he worked as an associate professor at the HSE University and as a senior research fellow at the Institute of Control Sciences of the Russian Academy of Science. Sergey is the author of more than 25 papers in peer-reviewed journals and edited volumes and 1 book on centrality in networks.

Piet Van Mieghem is professor at the Delft University of Technology with a chair in telecommunication networks and chairman of the section Network Architectures and Services (NAS) since 1998. His main research interests lie in the modelling and analysis of complex networks (such as infrastructural, biological, brain, social networks) and in new Internet-like architectures and algorithms for future communications networks. The focus of his chair is broadened from telecommunication networks to Network Science.

He is the author of four books: Performance Analysis of Communications Networks and Systems, Data Communications Networking, Graph Spectra for Complex Networks and Performance Analysis of Complex Networks and Systems.

He is a board member of the Netherlands Platform of Complex Systems, a steering committee member of the Dutch Network Science Society , an external faculty member at the Institute for Advanced Study (IAS) of the University of Amsterdam and an IEEE Fellow. He was awarded an Advanced ERC grant 2020 for ViSiON, Virus Spread in Networks.

Professor Van Mieghem received a Master degree (Magna cum Laude) and a Ph.D. degree (Summa cum Laude with Congratulations) in Electrical Engineering from the K.U.Leuven (Belgium) in 1987 and 1991, respectively. Before joining Delft, he worked at the Interuniversity Micro Electronic Center (IMEC) from 1987 to 1991. During 1993 to 1998, he was a member of the Alcatel Corporate Research Center in Antwerp where he was engaged in performance analysis of ATM systems and in network architectural concepts of both ATM networks (PNNI) and the Internet. He was a visiting scientist at MIT (department of Electrical Engineering, 1992-1993) and a visiting professor at UCLA (department of Electrical Engineering, 2005), at Cornell University (Center of Applied Mathematics, 2009), at Stanford University (department of Electrical Engineering, 2015) and at Princeton University (department of Electrical and Computer Engineering, 2022).

Currently, he serves on the editorial board of the OUP Journal of Complex Networks. He was member of the editorial board of Computer Networks (2005-2006), the IEEE/ACM Transactions on Networking (2008-2012), the Journal of Discrete Mathematics (2012-2014) and Computer Communications (2012-2015).

APPENDIX A
SYMBOLS

Only when explicitly mentioned, will we deviate from the symbols outlined here.

General Notation

k	discrete time

System Identification

$u[k]$	$m \times 1$ input vector at time k
$y[k]$	$d \times 1$ output vector at time k
$x[k]$	$n \times 1$ state vector at time k
$v[k]$	concatenation of vectors $u[k]$ and $x[k]$
n	order of the system, dimensionality of state vector
s	number of rows in Hankel matrices
h	number of columns in Hankel matrices
$Q=\begin{bmatrix} A & B \\ C & D \end{bmatrix}$	system matrix of the linear time-invariant state-space model

Temporal Graphs

G_k	temporal graph at time k
$\mathcal{N}$	set of all nodes in a temporal graph
N	number of nodes in a temporal graph, $N = \lvert\mathcal{N}\rvert$
$\mathcal{L}_k$	set of links in G_k
L_k	number of links in G_k, $L_k = \lvert\mathcal{L}_k\rvert$
$\mathcal{L}$	set of all links in G_k, $\mathcal{L} = \bigcup_{k=1}^{T} \mathcal{L}_k$
A_k	an $N \times N$ adjacency matrix of G_k with elements $a_{ij}(k)$
$C(X)$	$n \times p$ circulant matrix of state vectors $X = \{x[1], \ldots, x[p]\}$
$C(U)$	$L \times p$ circulant matrix of input vectors $U = \{u[1], \ldots, u[p]\}$
$C(V)$	$(n + L) \times p$ circulant matrix of vectors $V = \{v[1], \ldots, v[p]\}$
$a[k]$	$L \times 1$ binary vector with $a_i[k] = 1$ if the i-th link of the set $\mathcal{L}$ is present in graph G_k, otherwise $a_i[k] = 0$.
l	number of periodic graph sequences for the LG-gen model
p	period of a temporal graph
r	minimal order of matrix Q, complexity of periodic process
T	total number of observed graphs $G_1, G_2, \ldots, G_T$
V_1	$(L + n) \times p$ matrix of vectors with columns $[v[1] \ \ldots v[p-1] \ v[p]]$
V_2	$(L + n) \times p$ matrix of vectors with columns $[v[2] \ \ldots v[p] \ v[1]]$

APPENDIX B
NUMERICAL ALGORITHM FOR SUBSPACE STATE SPACE SYSTEM IDENTIFICATION (N4SID)

The N4SID algorithm is based on oblique projection. The oblique projection refers to the non-orthogonal projection, which is formed by projecting the vector space W_1 onto the vector space W_2 along the vector space W_3. The N4SID algorithm is always convergent (non-iterative) and numerically stable [1]. It can be also applied for subspace identification with process and measurement noise. The N4SID algorithm makes the following assumptions [2]:

Assumption 1. The state vector is sufficiently excited, i.e.,

$$rank\left(X_{0,h}\right) = n.$$

If the assumption 1 holds, the system is called reachable, that is, for any $k_1 < k_2$ there exists a sequence of input vectors $u\left[k_1\right], \ldots, u\left[k_2 - 1\right]$ that will transfer the system from state $x\left[k_1\right]$ to state $x[k_2]$.

Assumption 2. The input sequence $u[t] \in \mathbb{R}^m$ should satisfy the persistently exciting condition of order s. In other words, the blocked Hankel matrix $U_{0,s,h}$ is of full rank:

$$rank\left(U_{0,s,h}\right) = s \cdot m$$

Assumption 3. The row vectors of $X_{0,h}$ and $U_{0,s,h}$ are linearly independent, or there is no linear feedback from the states to the inputs, i.e.,

$$span\left(X_{0,h}\right) \cap span\left(U_{0,s,h}\right) = \{0\}.$$

The N4SID algorithm considers a special structure of block Hankel matrices. More precisely, it uses input-output estimation and a state vector to construct matrices

$$\begin{aligned} U_p &= U_{0,s,h-s}, & U_f &= U_{s,s,h-s}, \\ Y_p &= Y_{0,s,h-s}, & Y_f &= Y_{s,s,h-s}, \\ X_p &= X_{0,h-s}, & X_f &= X_{s,h-s}, \end{aligned}$$

where the subscripts p and f denote the past and future, respectively. These matrices can be combined into the past and future data matrices $W_p = \begin{bmatrix} U_p \\ Y_p \end{bmatrix}$ and $W_f = \begin{bmatrix} U_f \\ Y_f \end{bmatrix}$. The key observation is that, under assumptions 1-3 with s replaced by $2s$, there is no overlap between row spaces of W_p and U_f, $span\left(W_p\right) \cap span\left(U_f\right) = \{0\}$. Moreover, as shown in [2], the state vector X_f connects the past and future as it is a basis of the intersection of the past and future subspaces, i.e., $X_f = span(W_p) \cap span(W_f)$.

To estimate the system matrix Q, the N4SID algorithm estimates the state vector X_f using the idea of oblique projection: the output matrix Y_f is approximated by a matrix $\widehat{Y}_f$ which is an oblique projection of Y_f onto W_p along U_f (denoted by $\widehat{E}_{||U_f}\{Y_f|W_p\}$) and an oblique projection of Y_f onto U_f along W_p (denoted by $\widehat{E}_{||W_p}\{Y_f|U_f\}$), i.e.,

$$\widehat{Y}_f = \widehat{E}_{||W_p}\{Y_f|U_f\}) + \widehat{E}_{||U_f}\{Y_f|W_p\}) = \alpha \cdot U_f + \beta \cdot W_p.$$

Since the initial data do not necessarily correspond to the zero-input response, one can apply LQ-decomposition to transform data matrices into block matrices with zeros in the upper-right block

$$\begin{bmatrix} U_f \\ W_p \\ Y_f \end{bmatrix} = \begin{bmatrix} R_{11} & 0 & 0 \\ R_{21} & R_{22} & 0 \\ R_{31} & R_{32} & 0 \end{bmatrix} \begin{bmatrix} Q_1^T \\ Q_2^T \\ Q_3^T \end{bmatrix},$$

which yields to the equation

$$Y_f = \left(R_{31} - R_{32}R_{22}^{\dagger}R_{21}\right) R_{11}^{-1}U_f + R_{32}R_{22}^{\dagger}W_p, \qquad (15)$$

where $R_{22}^{\dagger}$ denotes the pseudo-inverse of R_{22}. Since row spaces of W_p and U_f, X_f and U_f do not intersect and $X_f \in span(W_p)$, equations (3) and (15) yields

$$\widehat{E}_{||U_f}\{Y_f|W_p\}) = R_{32}R_{22}^{\dagger}W_p = \Gamma_s X_f.$$

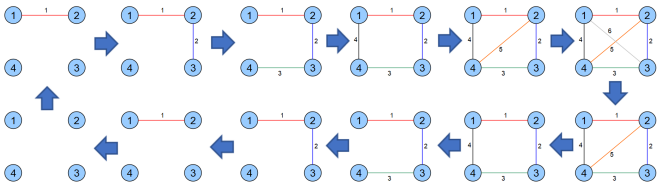

Fig. 12. Graph Dynamic №1 (4 nodes). Link labels correspond to a predefined order.

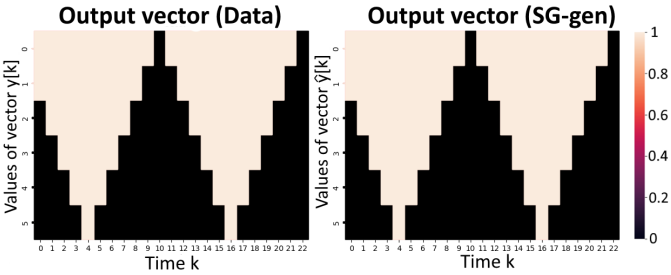

Fig. 13. Comparison of the SG-gen model with the real data for the Graph Dynamic №1 (4 nodes). The rows correspond to coordinates of real and estimated output vectors $y[k]$ (on the left) and $\widehat{y}[k]$ (on the right) while the colour indicates the values of these vectors.

Suppose that the SVD of an oblique projection be given by $\widehat{E}_{||U_f}\{Y_f|W_p\}) = U\Sigma V^T$ with $rank(\Sigma) = n$. Thus, the extended observability matrix and the state vector can be given as $\Gamma_s = U\Sigma^{1/2}$ and $X_f = \Sigma^{1/2}V^T$.

Finally, the N4SID algorithm estimates the system matrix (Q by applying the least-squares method to the equation

$$\begin{bmatrix} X_{s+1,h-s-1} \\ Y_{s,s,h-s-1} \end{bmatrix} = Q \begin{bmatrix} X_{s,h-s-1} \\ U_{s,s,h-s-1} \end{bmatrix}.$$

Appendix C
Experiments on Periodic Data

Graph dynamic №1.

An example of the graph dynamic №1 for a graph with 4 nodes is shown in Fig. 12. We observe that the periodic dynamic №1 can be modelled as a linear process. SG-gen accurately predicts the graphs sequence $G_2, \ldots, G_T$ from graph G_1 for such a simple dynamic. Fig. 13 illustrates the results for a graph dynamic with 4 nodes from Fig. 12.

The experiments indicate that the minimal number of periods to identify a system matrix Q with accurate predictions ($MSE\,(y, \widehat{y}) < 10^{-8}$) does not exceed 2 for networks with less than 14 nodes and does not exceed 3 for networks with up to 25 nodes. We emphasize that 2 periods is sufficient to reconstruct the correct graph sequence for networks with more than 14 nodes, however, the MSE will be higher ($MSE\,(y, \widehat{y}) < 10^{-4}$). Non-zero values of the MSE do not influence the accuracy of predictions, while the presence of errors is due to numerical finite accuracy (rounding) computation. The sequence of state vectors $x[1], \ldots x[T-1]$ has a period p. Fig. 14 illustrates the dynamics of $x[k]$ for a graph with 4 nodes ($p = 12$).

Interestingly, we observe that the minimal order of the system n is identical to the number of links in the graph G if we allow $MSE\,(y, \widehat{y}) < 10^{-2}$ (see Fig. 15). Since the initial dimensionality of the input data is L and the graph dynamic №1 has a period $p = 2L$, SG-gen expands the subspace to $2L$ by adding L linearly independent vectors. Therefore, the system matrix Q is $p \times p$ square matrix.

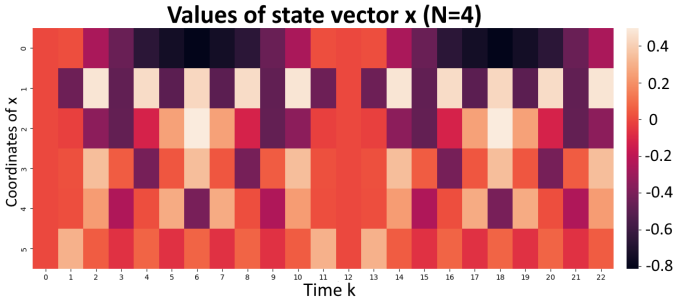

Fig. 14. Dynamic of state vector X for a graph with 4 nodes.

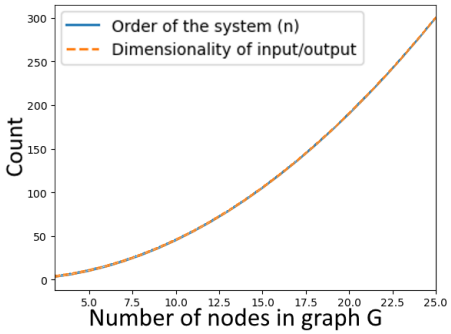

Fig. 15. Dimensionality of input/output vectors and state vector.

Fig. 16 illustrates eigenvalues of Q on a complex plane for a graph with 4 nodes ($p = 2L = 12$). All eigenvalues of the matrix Q are unique and evenly spaced around the unit circle; two eigenvalues are real numbers -1 and 1.

Graph dynamic №2.

We consider the graph dynamic №1 and assume that the graph does not change for 6 time slots if it is empty, for 4 time slots if it is full and for 2 time slots if it contains half the number of possible links. An example of such dynamic for a graph with 4 nodes is shown in Fig. 17.

SG-gen provides an ideal performance and generates the exact dynamic ($MSE\,(y, \widehat{y}) < 10^{-7}$). The minimal number of periods to identify Q accurately by the N4SID algorithm is 3 for graphs with 3 nodes and 9 for graphs with 10 nodes. Such an observation can be explained by the increased complexity of the process and a higher stability of subspace algorithms in case of many observations.

The dimension n of vector state $x[k]$ depends on the period $p$of the process and the dimensionality of the input vector L (see Fig. 18). We observe for the graph dynamic №2 that $n = p - L - 1$. All eigenvalues are distinct and spaced on the unit circle excluding -1.

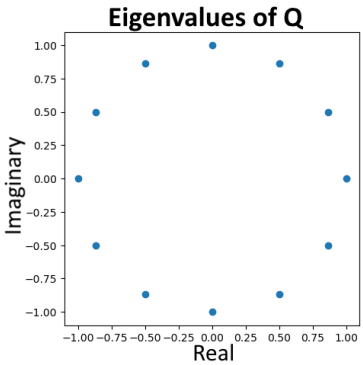

Fig. 16. Eigenvalues of system matrix Q for a graph with 4 nodes.

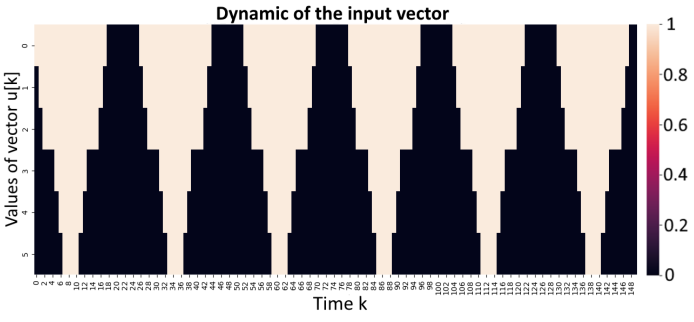

Fig. 17. Graph Dynamic №2 (4 nodes).

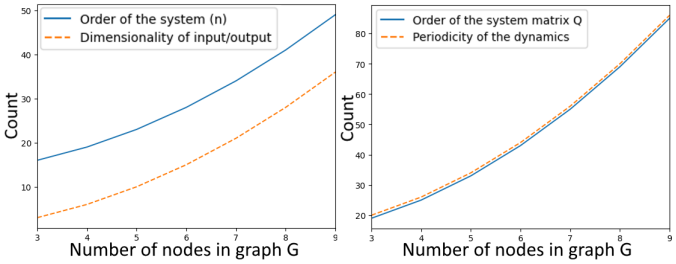

Fig. 18. The difference between the order of the system, dimension of inputs and periodicity.

Graph dynamic №3.

We assume that the probability of changing the graph is 4 times more likely than remaining it unchanged. An example of such dynamic is provided in Fig. 19.

SG-gen correctly generates any given graph sequence with $MSE\,(y, \widehat{y}) < 10^{-8}$. The minimal number of periods to learn the system matrix Q depends on the graph size but, in general, it is less than 7. The order n of the state vector satisfies $|p - L - n| \leq 1$ for all tested graphs (see Fig. 20). Thus, any periodic process which performs up to 1 random change in the graph structure can be modelled by SG-gen.

Graph dynamic №6.

We consider the graph dynamic №2 and enumerate all the periods when the network reaches $L/2$ links. We assume that if the enumerated number is even, the growth/decrease patterns remain the same as for the graph dynamic №2. On the contrary, if the number is odd, the growth (decrease) pattern will transform to the decrease (growth) pattern. The visualization of such process for a graph with 4 nodes is provided in Fig. 21.

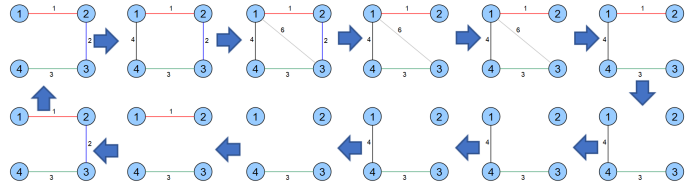

Fig. 19. Graph Dynamic №3 (4 nodes).

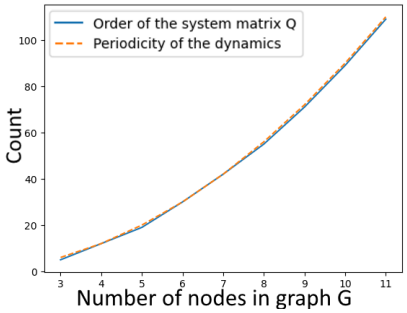

Fig. 20. Connection between the order of Q and the periodicity of the process.

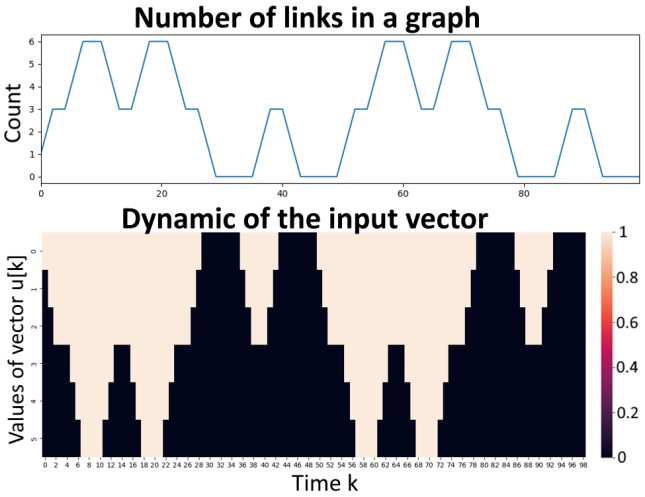

Fig. 21. Graph Dynamic №6 (4 nodes). The upper plot shows how the network changes with respect to the number of links in a graph while the bottom plot illustrates the dynamics of vector $u[k]$.

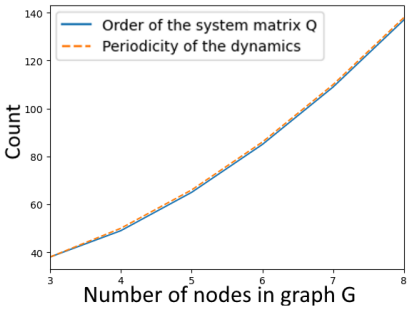

Fig. 22. Connection between the order of Q and the periodicity of the dynamics.

The graph dynamic №6 has been studied for graphs with up to 8 nodes. We observed that the SG-gen model provides an exact graph sequence $G_2, \ldots, G_T$. A graph with 4 nodes requires 6 periods while a graph with 8 nodes requires 32 periods. The dimension n of the state vector satisfies $|p - L - n| \leq 1$ where p is a period of the process and L is the total number of edges in the graph (see Fig. 22). The matrix of the system Q has the same properties as it is for the graph dynamic №2.

Graph dynamic №7.

We consider the graph dynamic №1 and assume that the sequences of link addition and removal are set randomly. An example of such a dynamic for a graph with $N = 4$ nodes is shown in Fig. 23. Since there is only 1 change difference per time slot, the graph sequence has the period $p = N(N-1)$.

The coordinates of the input vector are depicted in Fig. 24. To test the SG-gen model, we have examined graphs

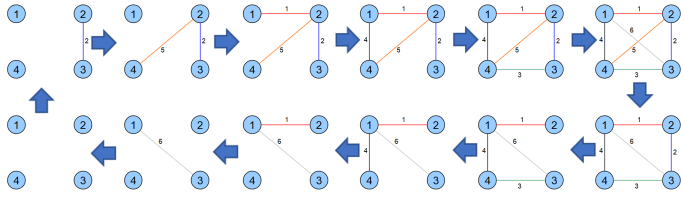

Fig. 23. Graph Dynamic №7 (4 nodes).

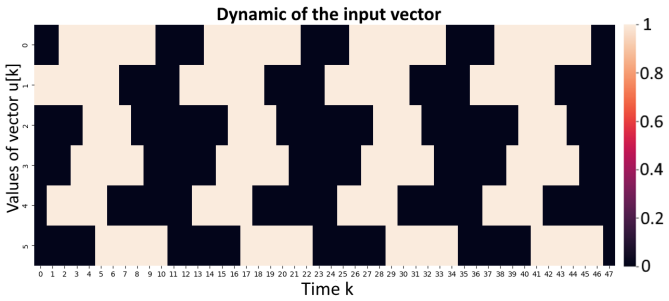

Fig. 24. Graph Dynamic №7 (4 nodes). Link change is performed randomly within a period.

TABLE 4
Percentage of perfectly simulated graph dynamics within 10 periods.

# of changes	Number of nodes in graph N						
	3	4	5	6	7	8	9
$r \leq 1$	100%	100%	100%	100%	100%	100%	100%
$r \leq 2$	99,9%	100%	99,4%	99,6%	99,8%	99,8%	99,8%
$r \leq 3$	99,8%	100%	100%	99,8%	99,9%	100%	99,9%
$r \leq 4$	-	100%	99,9%	100%	99,9%	99,9%	99,9%
$r \leq 5$	-	100%	100%	99,8%	100%	100%	99,9%
$r \leq 6$	-	100%	100%	100%	99,9%	100%	99,8%
$r \leq 7$	-	-	100%	100%	100%	100%	100%
$r \leq 8$	-	-	100%	100%	100%	100%	100%
$r \leq 9$	-	-	100%	100%	100%	100%	100%

with less than 20 nodes and generated 1,000 various graph dynamics. SG-gen perfectly predicts the graphs sequence $G_2, \ldots, G_T$ from graph G_1. Contrary to the graph dynamic №1, SG-gen requires more periods to identify the system matrix Q and make accurate predictions. The number of periods depends on the randomly generated graph sequence but, in general, it does not exceed 6 periods. Third, the dimension n of the state vector $x[k]$ is L. We can conclude that the SG-gen model can ideally generates any graph sequence if its dynamic is periodic, there is only 1 change per time slot and there is a monotonic growth in the number of links which eventually follows a monotonic decrease.

Graph dynamic №8.

We consider a graph dynamic which makes up to r changes per time slot in the graph structure. The period p of the process is $N(N-1)$.

SG-gen has been tested on graphs with less than 10 nodes while the MSE is averaged across 1,000 iterations. The maximal number of changes q per time slot in the graph structure is limited to the maximal possible number of links in a graph, i.e., $q \leq L$. Due to the computational complexity of SG-gen, the maximal possible number of periods to identify the system matrix Q was limited to 10.

Table 4 shows that the SG-gen model provides an exact performance for all the graphs that change by at most 1 link per time slot. The results agree with our previous results for the graph dynamic №3. The performance does not depend on the number of changes in the graph per time slot. Indeed, among 54,000 graph sequences, only 30 of them have not been identified within 10 periods; however, the increase of such parameter has resulted in their correct identification. Therefore, we may infer that any periodic graph sequence can be modelled correctly using the SG-gen model.

APPENDIX D
LINEAR PERIODIC GRAPH GENERATOR (LPG-GEN): PROOF OF THEOREMS IN SECTION 3.4

D.1 Proof of Theorem 3

Proof. Recall that $v[k] = \begin{bmatrix} x[k] \\ u[k] \end{bmatrix}$, where $L \times 1$ vector $u[k]$ is given and $n \times 1$ vector $x[k]$ is unknown. First, we examine how many vectors $v[1], \ldots, v[p]$ should be linearly independent. We consider the system of linear equations:

$$\sum_{i=1}^{p} \alpha_i v[i] = 0, \tag{16}$$

where $\alpha_1, \ldots, \alpha_n$ are unknown coefficients. If the solution is unique ($\alpha_i = 0$ for any $i \in \{1, ..., p\}$), then all the vectors $v[1], \ldots, v[p]$ are linearly independent and $R = p$. Otherwise, the vectors are linearly dependent and $R < p$. Multiplying (16) by Q and invoking (8) yields:

$$\sum_{i=1}^{p} \alpha_i Q v[i] = \sum_{i=1}^{p} \alpha_i v[i+1] = \sum_{i=1}^{p-1} \alpha_i v[i+1] + \alpha_p v[1] = 0,$$

because periodicity implies $v[p+1] = v[1]$. By repeating the same procedure of matrix multiplication, we obtain a system of linear equations

$$\begin{cases} \alpha_1 v[1] + \alpha_2 v[2] + ... + \alpha_p v[p] = 0, \\ \alpha_1 v[2] + \alpha_2 v[3] + ... + \alpha_p v[1] = 0, \\ \qquad\qquad \ldots \\ \alpha_1 v[p] + \alpha_2 v[1] + ... + \alpha_p v[p-1] = 0, \end{cases} \tag{17}$$

which can be written more compactly as (10). The solution of (10) also imposes the restrictions on the construction of the state vectors $x[1], \ldots, x[p]$.

The rank of the matrix $C(V)$ defines the minimal number of linearly independent vectors from the set $v[1], \ldots, v[p]$. Indeed, we can write that

$$rank(C(V)) = rank \begin{bmatrix} C(U) \\ C(X) \end{bmatrix},$$

where $C(X)$ is a $(n \cdot p) \times p$ circulant matrix of $n \times 1$ state vectors $X = \{x[1], \ldots, x[p]\}$. Since the matrix $C(U)$ is fixed and there are no restrictions on the matrix $C(X)$, we conclude that $rank(C(V)) \geq rank(C(U))$. Thus, the minimal number r of linearly independent vectors $v[1], \ldots, v[p]$ is $r = rank(C(U))$. □

D.2 Proof of Theorem 4

Proof. If all vectors $v[1], \ldots, v[p]$ are linearly independent ($r = p$), the proof follows from Lemma 2. Thus, suppose that $r < p$, which implies that vectors $v[1], \ldots, v[p]$ are not independent. For simplicity, we assume that vector $v[p]$ is linearly dependent. Then, there exists $\alpha_j \neq 0$ and $v[p]$ can be expressed from the *(p+1-j)*-equation of (17) as

$$v[p] = -\frac{1}{\alpha_j} \left(\sum_{i=1}^{j-1} \alpha_i v[p-j+i] + \sum_{i=j+1}^{p} \alpha_i v[i-j] \right). \tag{18}$$

If vector $v[p]$ satisfies (18), then the equation $v[1] = Qv[p]$ is redundant. Indeed, the substitution of relation (18) into $v[1] = Qv[p]$ and applying (8) yields

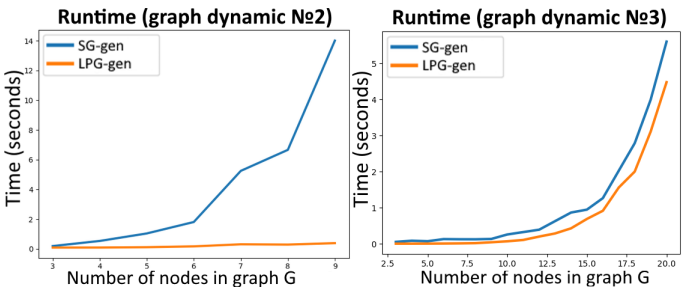

Fig. 25. Runtime of SG-gen and LPG-gen models on graph dynamics 2-3.

$$v[1] = -\frac{1}{\alpha_j}\left(\sum_{i=1}^{j-1}\alpha_i Qv[p-j+i] + \sum_{i=j+1}^{p}\alpha_i Qv[i-j]\right) =$$

$$= -\frac{1}{\alpha_j}\left(\sum_{i=1}^{j-1}\alpha_i v[p-j+i+1] + \sum_{i=j+1}^{p}\alpha_i Qv[i-j+1]\right).$$

Therefore,

$$\alpha_j v[1] + \sum_{i=1}^{j-1}\alpha_i v[p-j+i+1] + \sum_{i=j+1}^{p}\alpha_i Qv[i-j+1] =$$

$$= \sum_{i=1}^{j-1}\alpha_i v[p-j+i+1] + \sum_{i=j}^{p}\alpha_i Qv[i-j+1] = 0. \quad (19)$$

Relation (19) is the *((p+2-j) mod p)*-equation of (17). Since vectors $v[1], \ldots, v[p]$ satisfy (17), relation (19) always holds. Thus, the equation $v[1] = Qv[p]$ is redundant and relation (8) can be rewritten as

$$[v[2]\ \ v[3]\ \ \ldots\ \ v[p]\ ] = Q \cdot [v[1]\ \ v[2]\ \ \ldots\ \ v[p-1]\ ].$$

We have shown that the equation $v[1] = Qv[p]$ can be omitted if $v[p]$ is linearly dependent and satisfies (10). By induction, any equation $v[(k+1)\ mod\ p] = Qv[k]$ can be omitted if $v[k]$ is linearly dependent and satisfies (10). Thus, relation (8) can be reduced to

$$[v[k_1+1]\ \ \ldots\ \ v[k_r+1]] = Q \cdot [v[k_1]\ \ \ldots\ \ v[k_r]],$$

where $v[k_1], \ldots\ v[k_r]$ are linearly independent vectors. □

Appendix E
The Runtime of SG-gen and LPG-gen

We compare the runtime of SG-gen and LPG-gen models on graph dynamics №2 and №3. To reduce the complexity of the N4SID algorithm, which is used by SG-gen, the dimension n of state vectors is set in such a way that the order of the system matrix Q will be $p-1$ or p. Additionally, we averaged the runtime of SG-gen and LPG-gen across 100 iterations for the graph dynamics №3.

Fig. 25 shows that the graph dynamic №2 requires more time compared to the graph dynamic №3, because the the order of Q is larger for the graph dynamic №2. Overall, the LPG-gen model performs better than the SG-gen model. Indeed, the SG-gen model requires parameters s and h (the size of the block Hankel matrix) for the N4SID algorithm which, in general, are defined experimentally, which increases the runtime of SG-gen.

Appendix F
Linear Graph Generator (LG-gen): Proof of Theorems and Properties

F.1 Proof of Theorem 5a in Section 3.6

Proof. Since $\|u\|^2 = \sum_{k=1}^{T}\sum_{i=1}^{L} u_i^2[k]$ and $\|u-\bar{u}\|^2 = \sum_{k=1}^{T}\sum_{i=1}^{L}(u_i[k]-\bar{u}_i[k])^2$, we should check the following inequality:

$$\sum_{k=1}^{T}\sum_{i=1}^{L}(u_i[k]-\bar{u}_i[k])^2 \le \sum_{k=1}^{T}\sum_{i=1}^{L} u_i^2[k],$$

or, equivalently

$$\sum_{k=1}^{T}\sum_{i=1}^{L}\bar{u}_i^2[k] - 2\sum_{k=1}^{T}\sum_{i=1}^{L} u_i[k]\bar{u}_i[k] \le 0.$$

The sequence $\bar{u} = \{\bar{u}[1], \ldots, \bar{u}[T]\}$ is periodic; thus, $\bar{u}_i[k] = \bar{u}_i[k+p]$ for $\forall k = 1, \ldots T-p$ and $\forall i = 1, \ldots, L$. Moreover, the term $\bar{u}_i[k]$ is constructed by averaging the initial sequence u, i.e.,

$$\bar{u}_i[k] = \frac{u_i[k\ mod\ p] + \ldots + u_i[k\ mod\ p + (n_k-1)p]}{n_k},$$

where n_k is the total number of vectors that corresponds to $(k\ mod\ p)$-th element of the periodic graph sequence $\bar{u}$. The parameter n_k satisfies $k\ mod\ p + (n_k-1)p \le T$ and $k\ mod\ p + n_k p > T$. Thus, one can write that

$$\sum_{k=1}^{T}\sum_{i=1}^{L}\bar{u}_i^2[k] = n_1\sum_{i=1}^{L}\bar{u}_i^2[1] + \ldots + n_p\sum_{i=1}^{L}\bar{u}_i^2[p] =$$

$$= \sum_{k=1}^{p}\sum_{i=1}^{L} n_k\bar{u}_i^2[k]$$

and

$$\sum_{k=1}^{T}\sum_{i=1}^{L} u_i[k]\bar{u}_i[k] = \sum_{k=1}^{p}\sum_{i=1}^{L}(u_i[k\ mod\ p] + \ldots +$$

$$+ u_i[k\ mod\ p + (n_k-1)p])\bar{u}_i[k] = \sum_{k=1}^{p}\sum_{i=1}^{L} n_k\bar{u}_i^2[k].$$

Therefore, the initial inequality becomes

$$-\sum_{k=1}^{p}\sum_{i=1}^{L} n_k\bar{u}_i^2[k] \le 0.$$

Since $n_1, \ldots, n_p > 0$, the left part of inequality is zero if and only if all average values $\bar{u}_i[k]$ are zeros. In other words, the periodic sequence $\bar{u} = \{\bar{u}[1], \ldots, \bar{u}[T]\}$ is a set of zero vectors. Otherwise, $\|u-\bar{u}\| < \|u\|$. □

F.2 Proof of Theorem 5b in Section 3.6

Proof. We consider an arbitrary periodic set of vectors $z = \{z[1],\ \ldots, z[T]\}$ in $\mathbb{R}^L$ with period p. To prove the theorem, we need to minimize the function

$$f(z) = \|u-z\|^2 = \sum_{k=1}^{T}\sum_{i=1}^{L}(u_i[k]-z_i[k])^2.$$

Since $z[k] = z[k+p]$ for $\forall k = 1, \ldots, T-p$, we can rewrite the function as

$$f(z) = \sum_{k=1}^{T}\sum_{i=1}^{L}(u_i[k] - z_i[k])^2 = \sum_{k=1}^{p}\sum_{i=1}^{L}(n_k z_i^2[k] -$$

$$- 2(u_i[k] + \ldots + u_i[k + p(n_k - 1)])z_i[k]) + \sum_{k=1}^{T}\sum_{i=1}^{L} u_i^2[k] =$$

$$= \sum_{k=1}^{p}\sum_{i=1}^{L} n_k \left(z_i[k] - \frac{u_i[k] + \ldots + u_i[k + p(n_k - 1)]}{n_k}\right)^2 +$$

$$+ \sum_{k=1}^{T}\sum_{i=1}^{L} u_i^2[k] - \sum_{k=1}^{p}\sum_{i=1}^{L} \frac{(u_i[k] + \ldots + u_i[k + p(n_k - 1)])^2}{n_k},$$

where n_k is the total number of vectors that corresponds to $(k \ mod \ p)$-th element of the periodic graph sequence z. Since $n_1, \ldots, n_p > 0$, the minimum value of f is attained when

$$z_i[k] = \frac{u_i[k] + \ldots + u_i[k + p(n_k - 1)]}{n_k}.$$

Therefore, we observe that $z_i[k]$ is the average value of the i-th components of vectors $u_i[k], \ldots, u_i[k + p(n_k - 1)]$. In other words, $z = \{z[1], \ \ldots, z[T]\}$ is an average periodic sequence, i.e., $z = \bar{u}$. □

F.3 Average Periodic Pattern Decomposition

We have proved in Theorem 5 that the subtraction of the average periodic graph sequence always decreases the mean square error (MSE) if and only if the average sequence of period p is not a zero sequence. Therefore, it is important to understand the following:

1) Is it possible to find a non-zero average periodic sequence at each iteration of the LG-gen model? What is the smallest period p^* for such a sequence?
2) Is it possible to achieve a zero sequence by the sequential subtraction of average periodic sequences?

We denote by $u = (u[1], u[2], \ldots, u[T])$ an arbitrary non-zero vector sequence where $u[k] \in \mathbb{R}^L$ for $\forall k = 1, \ldots, T$. Additionally, let $\bar{u}_p = (\bar{u}_p[1], \bar{u}_p[2], \ldots, \bar{u}_p[p])$ be an average vector sequence, which is defined by relation (14). In fact, since u is arbitrary and the subtraction of average periodic sequences is performed independently for each row of u, we examine a graph with 1 link ($L = 1$) for simplicity.

First, we define the smallest period $p^* \leq T$ such that $\bar{u}_{p^*}$ is a non-zero sequence for any $u \neq 0$. Indeed, we need to find the smallest number p^* such that

$$\forall p \leq p^* \quad \bar{u}_p = 0 \Leftrightarrow u = 0. \tag{20}$$

Since the average sequence $\bar{u}_p$ consists of p elements, the equation $\bar{u}_p = 0$ implies p linear equations. Hence, relation (20) contains a system of $(1 + p^*)p^*/2$ linear equations with T variables $u[1], u[2], \ldots, u[T]$. Thus, we need to identify the smallest number p^* in such a way that relation (20) contains T linearly independent equations and, consequently, has a unique trivial solution $u[1] = u[2] = \ldots = u[T] = 0$.

Fig. 26 illustrates the results for $T \leq 3000$ where the minimal period p^* linearly increases with the number of observations T. Indeed, $p^* = p(T) \approx 0.11T - 10$ for $T \geq 300$. In other words, if $u = (u[1], u[2], \ldots, u[T])$ is a non-zero sequence, then there always exists a non-zero average sequence of period $p \leq 0.11T - 10$ that decreases the MSE.

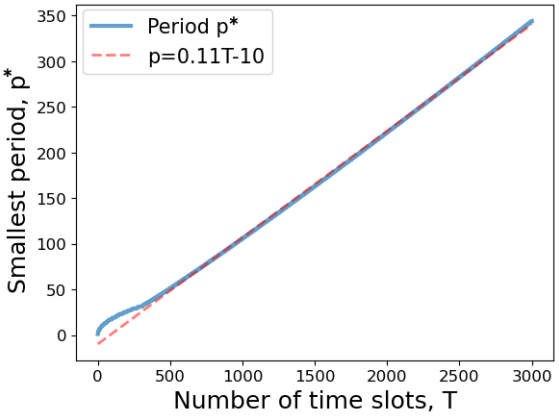

Fig. 26. The smallest period p^* to decompose any arbitrary sequence u of length T.

Therefore, we conclude that LG-gen always decreases the MSE by relatively short periodic sequences.

Next, we determine whether an arbitrary sequence u always converges to zero by LG-gen. In fact, Sethares and Staley [3] have shown that the subtraction of average periodic sequence $\bar{u}_p$ performs an orthogonal projection of u onto vector subspace $\mathcal{P}_p$, which contains all sequences of period p. In other words, the residual $u - \bar{u}_p \notin \mathcal{P}_p$ or, equivalently, $u - \bar{u}_p \in \mathcal{P}_p^{\perp}$ where $\mathcal{P}_p^{\perp}$ is the orthogonal complement of a subspace $\mathcal{P}_p$.

We denote by $r_l \in \mathbb{R}^T$ the residual vector, which is obtained from u by the sequential subtraction of l average sequences with periods $p_1, p_2, \ldots, p_l$. Since the residual vector r_l does not belong to the sum of vector subspaces $\mathcal{P}_{p_1}, \ldots, \mathcal{P}_{p_l}$, r_l is an element of the vector subspace V where the dimension $dim(V)$ of the vector space V is defined by

$$dim(V) = T - dim(\mathcal{P}_{p_1} + \ldots + \mathcal{P}_{p_l}). \tag{21}$$

Relation (21) defines the convergence of the LG-gen model. If the sum of vector subspaces $\mathcal{P}_{p_1}, \ldots, \mathcal{P}_{p_l}$ spans the entire space $\mathbb{R}^T$, the residual vector $r_l = 0$ because $r_l \in V$ and $dim(V) = 0$. An important question is to find the smallest period p^* such that $\mathcal{P}_1 + \mathcal{P}_2 + \ldots + \mathcal{P}_{p^*} = \mathbb{R}^T$. Relation (20) is also the solution to the problem. Thus, we conclude that the LG-gen model accurately describes any sequence u.

F.4 Performance of LocalMin

The LG-gen model is based on the periodicity transform (PT), which identifies periods based on the *LocalMin* algorithm. However, the PT does not in general provide a unique representation and there are other methods to identify periods in the data (see Table 5), which are described in Section F.4.1. Thus, we compare the performance of the *LocalMin* algorithm against other existing methods.

F.4.1 Other periodic transforms

Sethares and Staley [3] introduce various algorithms that define the order in which the projections are applied:

1) *Small2Large:* the algorithm iteratively selects the smallest "significant" period p_i, which satisfies
$$\|u - \bar{u}_p\| < h \cdot \|u\|,$$
where h is a predefined threshold.

TABLE 5
Periodic decomposition techniques.

#	Name	Description
1	*small2large*	the Small2Large algorithm ($h = 0.9$)
2	*l-best*	the l-best algorithm
3	*l-best-gamma*	the l-best-gamma algorithm
4	*Best correlation*	the best correlation algorithm
5	*LocalMin*	the LocalMin algorithm
6	*PT (DFT)*	the best frequency algorithm
7	*PT (Farey)*	the sequence of periods is based on the Farey representation
8	*PT (Ramanujan)*	the sequence of periods is based on the Ramanujan PT
9	*PT (iterative DFT)*	the sequence of periods is iteratively defined by the DFT of the residual
10	*PT (iterative Farey)*	the sequence of periods is iteratively defined by the Farey representation of the residual
11	*PT (iterative Ramanujan)*	the sequence of periods is iteratively defined by Ramanujan PT of the residual

2) *l-best:* the algorithm consists of two steps. First, the l-best algorithm identifies l periods that sequentially provide the largest decrease of the MSE. Second, each average periodic sequence $\bar{u}_{p_i}$ is further decomposed into their constituent periodic elements, which are the factors of a period p_i, to see if these smaller (sub)periods decrease the MSE more than another currently in the list.
3) *l-best$_\gamma$:* a variation of the l-best algorithm where the MSE is normalized by the square root of p_i.
4) *Best Correlation:* the algorithm iteratively selects the period p_i with the highest correlation between u and the p_i-periodic basis vector $x^s_{p_i}$, which is given by

$$x^s_{p_i}[j] = \begin{cases} 1, if\ (j-s)\ mod\ p_i = 0, \\ 0, otherwise, \end{cases}$$

where $s \in \{1, \ldots, p_i\}$.
5) *Best Frequency:* the algorithm performs the Discrete Fourier transform (DFT) of the vector u and then sequentially converts the frequency f_i with the i-th largest magnitude to the closest integer period $p_i = round(1/f_i)$. The average periodic sequence $\bar{u}_{p_i}$, which is subtracted from u, differs from a sinusoidal function with frequency f_i. Therefore, we also consider the *recursive best frequency* algorithm, which recalculates the DFT of u at each step i.

The major disadvantage of the PT is the concatenated structure of periodic subspaces, which leads to the consideration of different projections sequences. An alternate approach was suggested in [4], [5], where the Farey dictionary is introduced. This dictionary is based on the union of non-overlapping columns of several DFT matrices. More precisely, given the maximal period p_{max}, the Farey dictionary $A^{(f)}_{p_{max}}$ is the $T \times \Phi(p_{max})$ block matrix

$$A^{(f)}_{p_{max}} = \begin{bmatrix} V_1 & V_2 & \ldots & V_{p_{max}} \end{bmatrix},$$

where $\Phi(p_{max}) = \sum_{m=1}^{p_{max}} \phi(m)$ and $\phi(m)$ is the Euler totient function (number of integers in $1 \le i \le m$ coprime to m) and the $T \times \phi(m)$ matrix V_m is given by

$$V_m = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ W_m^{k_1} & W_m^{k_2} & \cdots & W_m^{k_{\phi(m)}} \\ W_m^{2k_1} & W_m^{2k_2} & \cdots & W_m^{2k_{\phi(m)}} \\ \vdots & \vdots & \ddots & \vdots \\ W_m^{(T-1)k_1} & W_m^{(T-1)k_2} & \cdots & W_m^{(T-1)k_{\phi(m)}} \end{bmatrix}$$

with $W_m = e^{i2\pi/m}$ and $gcd(m, k_i) = 1$ for $1 \le k_i \le m$. The notation $gcd(m, k_i)$ refers to the greatest common divisor (GCD) while the relation $gcd(m, k_i) = 1$ means that integers m and k_i are coprime. The column space of V_m contains m-periodic sequences. Moreover, any set of T columns in $A^{(f)}_{p_{max}}$ is linearly independent.

The Farey dictionary represents u in the form

$$u = A^{(f)}_{p_{max}} \cdot d \tag{22}$$

where d is an $\Phi(p_{max}) \times 1$ vector, which defines how u can be decomposed as a linear combination of periodic sequences. The solution of d is not unique, therefore, Vaidyanathan and Pal [5] minimizes the l_1 norm of d to obtain a sparse solution for vector d. The Farey Representation of u is computationally expensive as the Farey dictionary $A^{(f)}_{p_{max}}$ has approximately $O(p^2_{max})$ columns.

As an alternative to the Farey representation, Tenneti and Vaidyanathan [4], [6] have introduced the Ramanujan periodicity transform matrix $A^{(R)}_{p_{max}}$ of size $T \times \Phi(p_{max})$ as

$$A^{(R)}_{p_{max}} = \begin{bmatrix} C_1 & C_2 & \ldots & C_{p_{max}}, \end{bmatrix}$$

where $C_m = \begin{bmatrix} c_m & c_m^{(1)} & \cdots & c_m^{(\phi(m)-1)} \end{bmatrix}$ is a $T \times \phi(m)$ matrix, $c_m^{(i)}$ is the circularly downshifted by i version of the $T \times 1$ vector c_m and the k-th component of c_m is the Ramanujan sum

$$(c_m)_k = \sum_{n=1:gcd(n,m)=1}^{m} W_m^{kn}.$$

A distinct feature of the Ramanujan periodicity transform matrix $A^{(R)}_{p_{max}}$ is that the column space of C_m contains m-periodic sequences and the column spaces of C_m and C_q are orthogonal whenever m and q are the divisors of T.

The Ramanujan Periodicity transforms defines u as

$$u = A^{(R)}_{p_{max}} \cdot d \tag{23}$$

where d is an $\Phi(p_{max}) \times 1$ vector, which defines how u is decomposed as a linear combination of periodic sequences. The solution of d is not unique. Therefore, Tenneti and Vaidyanathan formulate the l_1 and l_2 convex programs for d and demonstrate that the l_2 norm solution results in a much faster computation. Again, the Ramanujan Periodicity transforms of u is computationally expensive as the matrix $A^{(R)}_{p_{max}}$ has approximately $O(p^2_{max})$ columns.

Overall, we compare the *LocalMin* algorithm to 10 periodic decomposition techniques from Table 5. The Farey and the Ramanujan representations are only used to identify periods in the data, because there is no guarantee that the subtraction of l periodic sequences from the matrices $A^{(F)}_{p_{max}}$ and $A^{(R)}_{p_{max}}$ decreases the MSE of u.

We attempt to identify a relatively small number of periodic graph sequences that describe the dynamics of

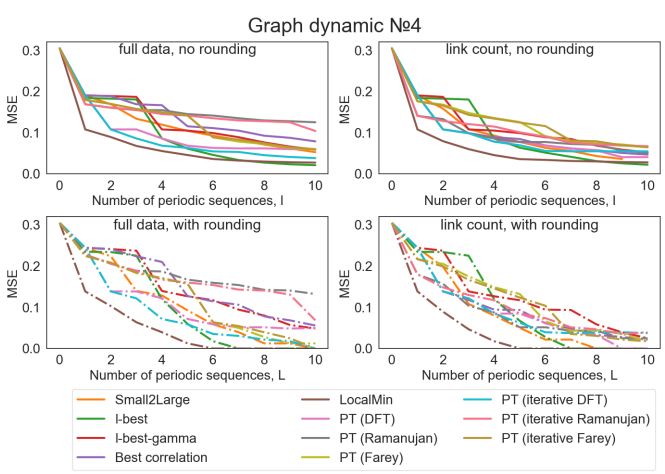

Fig. 27. The comparison of periodic transforms for Graph Dynamic №4.

the graph. In general, the identification of periods can be defined on the $L \times T$ vector sequence $a[1], \ldots, a[T]$ corresponding to the graphs $G_1, \ldots, G_T$ (*full data*). However, due to the high computational complexity, such an approach is impractical for very large graphs. Hence, we also identify periods using a 1-dimensional sequence that describes the dynamics of *link count* in the temporal network.

F.4.2 Experiments on artificial quasi-periodic data

We test two quasi-periodic graph dynamics №4 and №5 from Section 3.7. Fig. 27 illustrates the MSE of the periodicity transforms on the graph dynamics №4 (6 nodes, 15 links, $T = 80$) for $l = 10$ and $p_{max} = T/2$.

First, the rounding at the final step dramatically decreases the number l of periodic sequences required to accurately describe the initial graph dynamics. For instance, if $l = 10$, then the lowest MSE without rounding is approximately 0.02 for the l-best and the LocalMin algorithms. However, the rounding at the final step provides a zero MSE for LocalMin ($l = 6$), l-best ($l = 7$), small2large ($l = 8$) and PT (DFT, $l = 7$). Second, since the link count has the same period as the graph dynamics №4, it is a good measure to define the sequence of projections. For instance, the LocalMin algorithm accurately describes the graph dynamic №4 using 5 periodic sequences if the sequence of periods is identified based on the dynamics of the link count. Overall, we observe that *the LocalMin algorithm outperforms other PT methods* because it accurately describes the initial graph dynamics by a smaller number of periodic sequences.

Next, we compare the PT techniques with respect to the identified periods $p_1, \ldots, p_l$. In general, we aim to find the PT, which accurately describes the initial graph dynamics by relatively short periods p_i as the order of the system matrix Q of LG-gen depends on p_i. Thus, Fig. 28 presents the comparison of the PT techniques with respect to the sum of the periods $\sum_{i=1}^{l} p_i$ and the space complexity, which is the total number of elements in matrices $Q_1, \ldots, Q_l$ of LG-gen. Overall, we observe that the LocalMin algorithm and PT (DFT) provide periodic sequences that have the lowest sum of periods as well as the lowest space complexity.

The results for the graph dynamic №5 are similar to the graph dynamic №4: the dynamics of the graph can be accurately described by a relatively small number of periodic sequences. The LocalMin, the l-best and the small2large

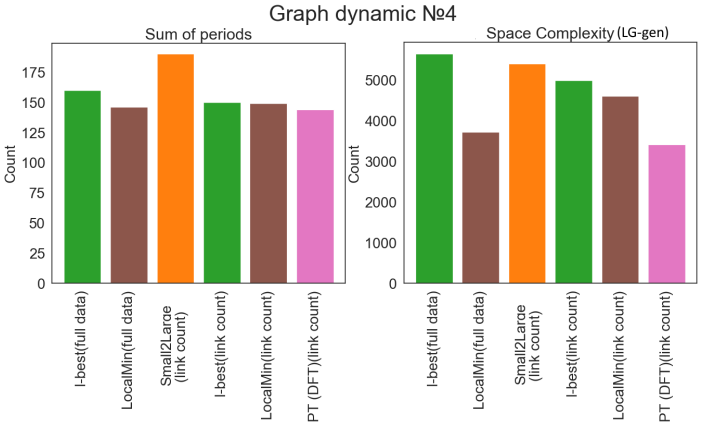

Fig. 28. The sum of periods and the space complexity of the PT for the graph dynamic dynamics №4.

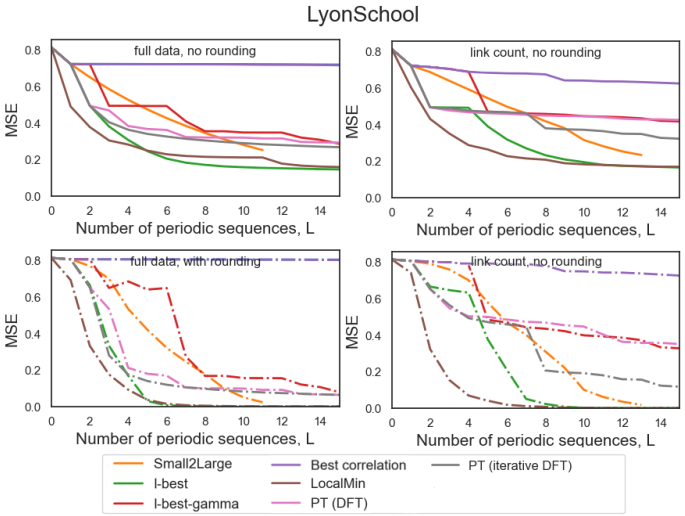

Fig. 29. The PT of the LyonSchool network (MSE).

algorithms with rounding produce an exact periodic decomposition within 10 periods while LocalMin has the lowest space complexity.

F.4.3 Experiments on Real Data

We compare the periodicity transforms on a real network from Section 3.8, collected in the school (LyonScool dataset). Due to the high computational complexity, we have excluded the PT methods №8-11, which are based on Farey or Ramanujan dictionaries.

The performance of the periodicity transforms with respect to different values of $l \leq 15$ is provided in Fig. 29. First, the results on the link count are similar to the results on the full data. For instance, the l-best algorithm, which is performed on the full data and on the link count, requires 9 periodic sequences to describe exactly the dynamics of the graph. Thus, the link count is a good metric to define the sequence of periodic projections in the LyonSchool dataset.

We observe that only l-best and LocalMin algorithms accurately define the graph dynamics within 12 periods. Therefore, we also compare these algorithms with respect to the identified periods $p_1, \ldots, p_l$. Fig. 30 demonstrates that the l-best algorithm (full data) provides the lowest sum of the periods $\sum_{i=1}^{l} p_i$ while the LocalMin algorithm (link count) has the smallest number of elements in matrices $Q_1, \ldots, Q_l$ of LG-gen.

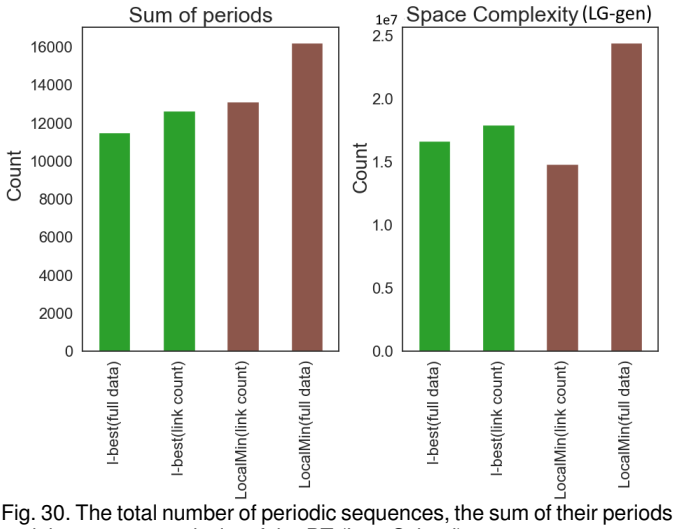

Fig. 30. The total number of periodic sequences, the sum of their periods and the space complexity of the PT (LyonSchool).

REFERENCES

[1] P. Van Overschee and B. De Moor, ""N4SID" subspace algorithms for the identification of combined deterministic stochastic system", *Automatica,* 30(1):75–93, 1994.

[2] T. Katayama, *Subspace Methods for System Identification,* Springer Berlin Heidelberg, New York, 2005.

[3] Sethares, W.A. and Staley, T.W. (1999), "Periodicity transforms," *IEEE Transactions on Signal Processing,* 47(11): 2953-2964.

[4] Tenneti, S. and Vaidyanathan, P. (2014), "Dictionary approaches for identifying periodicities in data," 2014 48th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, pp. 1967-1971.

[5] Vaidyanathan, P. and Pal, P. (2014) "The farey-dictionary for sparse representation of periodic signals," 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, pp. 360-364.

[6] Tenneti, S. and Vaidyanathan, P. (2015), "Nested Periodic Matrices and Dictionaries: New Signal Representations for Period Estimation," in IEEE Transactions on Signal Processing, vol. 63, no. 14, pp. 3736-3750.