Temporal Graph Signal Decomposition

Maxwell McNeil*
mmcneil2@albany.edu
University at Albany—SUNY, USA

Lin Zhang
lzhang22@albany.edu
University at Albany—SUNY, USA

Petko Bogdanov
pbogdanov@albany.edu
University at Albany—SUNY, USA

ABSTRACT

Temporal graph signals are multivariate time series with individual components associated with nodes of a fixed graph structure. Data of this kind arises in many domains including activity of social network users, sensor network readings over time, and time course gene expression within the interaction network of a model organism. Traditional matrix decomposition methods applied to such data fall short of exploiting structural regularities encoded in the underlying graph and also in the temporal patterns of the signal. How can we take into account such structure to obtain a succinct and interpretable representation of temporal graph signals?

We propose a general, dictionary-based framework for temporal graph signal decomposition (TGSD). The key idea is to learn a low-rank, joint encoding of the data via a combination of graph and time dictionaries. We propose a highly scalable decomposition algorithm for both complete and incomplete data, and demonstrate its advantage for matrix decomposition, imputation of missing values, temporal interpolation, clustering, period estimation, and rank estimation in synthetic and real-world data ranging from traffic patterns to social media activity. Our framework achieves 28% reduction in RMSE compared to baselines for temporal interpolation when as many as 75% of the observations are missing. It scales best among baselines taking under 20 seconds on 3.5 million data points and produces the most parsimonious models. To the best of our knowledge, TGSD is the first framework to jointly model graph signals by temporal and graph dictionaries.

CCS CONCEPTS

• **Information systems** → **Data mining**.

KEYWORDS

graph mining, signal processing, time series, dictionary coding, sparse decomposition, interpolation, periodicity detection

ACM Reference Format:
Maxwell McNeil, Lin Zhang, and Petko Bogdanov. 2021. Temporal Graph Signal Decomposition. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467379

*Corresponding Author

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467379

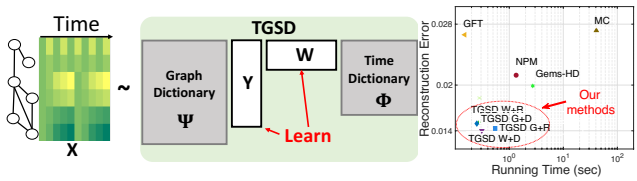


Figure 1: (Left panel:) TGSD decomposes a temporal graph signal as a product of two fixed dictionaries and two corresponding sparse encoding matrices. (Right panel:) Variants of TGSD (lower left corner) are both more accurate in missing value reconstruction and faster than most baselines from the literature. Details of competing techniques are available in Sec. 6.

1 INTRODUCTION

Multivariate time series often feature temporal and "spatial" structure inherent to the domain in which they are collected. Incorporating this structure when mining such data is essential to obtaining parsimonious, robust, and interpretable models. In this paper we focus specifically on temporal data associated with the nodes of a fixed graph and refer to such data as *temporal graph signals (TGS)*. Example settings abound: traffic over time in transportation networks [22], temporal readings in sensor networks [48], gene expression time series overlaid on protein interactions networks [4], and user opinion dynamics in social networks [3]. Beyond structural coupling on the graph, temporal graph signals may exhibit temporal structures such as periodicity, smoothness, trends, and others. Our goal in this work is to obtain a low-dimensional embedding of a TGS which jointly exploits such graph and temporal properties. It is important to note that the TGS setting is fundamentally different from dynamic graph mining where the structure (sets of nodes and edges) evolves as opposed to signals over the nodes [6]. The benefit of employing knowledge of the underlying graph structure has been demonstrated in graph signal processing [13, 37, 41], where node values are treated as a signal over the graph and the spectrum of the graph Laplacian is employed as basis for reconstructing the signal. Temporal extensions have also been recently proposed [50]. Graph structure modeling has found valuable applications in compression/summarization of node attributes [42], in-network regression and optimization [18], missing value imputation [19], community detection [17] and anomaly detection [2].

Similarly, temporal structures such as periodicity [45], trends [53] and smoothness [10] have a long history in research on time series. Most existing methodology focuses on modeling either the graph structure [2, 18, 19, 42] or the structure of the temporal signal [45, 53]. The interplay between temporal and structural graph properties gives rise to important behaviors, however, no frameworks currently exist to facilitate a joint representation. For example, traffic levels in a transportation network are shaped by both network locality and the time of the day [12].

Missing values [9] and irregular sampling in the time and/or the graph domain [31] also pose important challenges in TGS data analysis. Missing values may arise due to sensor malfunction or

other kinds of data corruption; sub-sampling in the graph domain may arise in social media data due to privacy constraints, while temporal sub-sampling may arise due to limitations of how fast time snapshots can be collected. A joint structural modeling of the graph and temporal patterns can be especially advantageous in such settings of missing or incomplete data. Hence, the key question we seek to address in this paper is: *How to efficiently learn a succinct, robust and interpretable representation for temporal graph signals, which can (i) jointly exploit temporal and graph structural regularities and (ii) handle missing values?*

We propose a general framework for *temporal graph signal decomposition (TGSD)* based on joint dictionary encoding in the graph and time domain. TGSD decomposes the signal as a product of four matrices: two fixed dictionaries and two corresponding sparse encoding coefficient matrices (Fig. 1, left pane). Our framework flexibly incorporates widely adopted temporal and graph dictionaries from the literature and can also be employed for data with missing values. We develop a general and efficient solver for TGSD and demonstrate its utility in datasets from different domains. The framework achieves significant performance gains over baselines for matrix decomposition, missing value reconstruction, temporal interpolation, node clustering, and periodicity detection, while scaling significantly better than most of them (Fig. 1, right pane).

Our contributions in this paper are as follows:

• **Generality and novelty:** We propose a general dictionary-based decomposition framework for temporal graph signals. To the best of our knowledge TGSD is the first method to unify graph signal processing and time series analysis by incorporating existing as well as custom dictionaries from both domains.

• **Scalability and parsimony:** TGSD scales to large instances on par or better than high-accuracy baselines. It produces low complexity and interpretable representations of the input data.

• **Applicability and accuracy:** We demonstrate TGSD's utility for data decomposition, missing value imputation, interpolation, clustering, period detection, and rank estimation. Its quality dominates that of baselines across applications.

2 RELATED WORK

Sparse dictionary coding aims to represent data via a sparse combination of dictionary basis. It is widely used in signal processing [55], image analysis [15], and computer vision [47]. Dictionaries are designed to capture the underlying structure in the data, e.g. DFT [40] and DWT [50]. Some methods learn the dictionary from data [56]. Our work focuses specifically on signals over graphs and can accommodate arbitrary time and graph dictionaries designed to match expected structures in the data, making the literature on dictionary encoding complementary to our approach.

Graph signal processing (GSP) specifically models signals over graph nodes and is a popular emerging research area in signal processing [41]. A central premise is that a graph signal can be represented as a linear combination of graph dictionary basis. The eigenvectors of the graph Laplacian are often adopted as basis in this domain [13]. Proposals to learn the basis from data also exist in the literature [43]. Beyond a univariate signal over nodes, a recent approach considers also multivariate graph signals [50]. Our work focuses on evolving graph signals, and thus, can be viewed as a generalization of static GSP methods.

A recent clustering approach, called CCTN, focuses on temporal evolution of signals over graphs to clusters nodes [30]. CCTN computes a sparse data representation and employs the network structure as a regularizer. Different from CCTN, our work employs the graph structure for dictionary encoding and can be applied to many downstream tasks in addition to clustering. We also outperform CCTN for clustering both in terms of scalability and accuracy.

Mining and optimization for network samples is another relevant area which differs from classical machine learning in that features are associated with network nodes [39, 54]. Common to this setting is that network samples share a common structure but are modeled as independent, while in our setting they are ordered in time and this temporal order is crucial. An optimization framework for graph node values was recently proposed with a key premise of local smoothness [18]. This work, however, does not consider temporal evolution of the node values and uses the graph structure for regularization rather than encoding.

Matrix completion techniques estimate missing values in incomplete matrices [36]. Some employ nuclear norm minimization [20] and others low-rank semi-definite programming [33]. These methods are developed for general matrices and are agnostic to existing column/row side information (e.g., time and graph structures) which we consider in TGSD. Some recent methods incorporate side information as regularizers to improve the completion quality [23, 38] or employ known temporal patterns such as periodicity [44]. Different from the above, we perform joint row and column structured encoding, which, as we demonstrate experimentally, outperforms general matrix completion methods.

Dynamic graph mining methods seek to discover patterns from the evolving graph structure as opposed to node signals that evolve over a fixed structure [6, 29]. Classic problems in this domain area include link prediction and recommendation [25] community detection [29] and frequent pattern mining [6]. All these methods focus on structural dynamics, while our work focuses on node signal evolution within a fixed structure.

Graph neural networks are popular deep architectures shown to achieve state-of-the-art performance in text classification [51], recommender systems [34] and other tasks [24]. GNNs employ an input graph structure to learn representations of node values based on their network context. Some methods in this domain also consider time, but are primarily designed for forecasting [52], action recognition [49], or dynamic graph structure [32], as opposed to evolving signals over a graph. In addition, this group of methods are often designed for specific tasks (e.g. recommendation and node classification), while our framework is general as it can be employed for an array of tasks and can incorporate various dictionaries.

3 PROBLEM FORMULATION

A graph $G = (\mathcal{M}, H)$ is a set of nodes $\mathcal{M}$, $|\mathcal{M}| = n$, and a weighted adjacency matrix H whose non-zero entries encode the strength of the edges among corresponding nodes. The combinatorial Laplacian matrix L associated with a graph G is defined as $L = F - H$, where F is is a diagonal matrix of weighted node degrees $F_{ii} = \sum_q H_{ij}$.

A temporal graph signal (TGS) is a matrix $X \in \mathbb{R}^{n \times t}$ with t time snapshots. Our goal is to succinctly encode the signal via graph and time dictionaries, resulting in a decomposition of the form:

$$X \approx \Psi Y W \Phi, \tag{1}$$

where $\Psi \in \mathbb{R}^{n\times m}$ is a fixed graph dictionary, $\Phi \in \mathbb{R}^{s\times t}$ is a fixed temporal dictionary and $Y \in \mathbb{R}^{m\times k}$ and $W \in \mathbb{R}^{k\times s}$ are the corresponding encoding coefficient matrices we have to estimate. We choose to use two encoding matrices Y and W, similar to many sparse coding and non-negative matrix factorization methods, in order to learn effectively both the graph and temporal structures and maintain separable interpretable embeddings for nodes and time snapshots. The internal dimension of the encoding matrices k is a parameter which controls the rank of the encoding, akin to the number of components in dimensionality reduction techniques. The internal sizes of the dictionaries m, t depend on the type of dictionaries selected discussed in the following section.

If unconstrained, the encoding matrices Y and W from Eq. 1 will overfit noise in the input. This may be especially exacerbated when the adopted dictionaries Ψ and Φ are large and exhaustive. To alleviate this limitation, we impose sparsity on the encoding matrices akin to sparse coding techniques. In addition, the basic encoding from from Eq. 1 cannot handle missing values in the input signal X or irregular temporal sub-sampling. To this end, we include a mask matrix Ω ensuring a fit to observed values.

Definition 3.1. Sparse temporal graph signal decomposition with missing values: *Given a temporal graph signal X with missing values, a binary mask Ω of the same size, graph Ψ and time Φ dictionary matrices and rank k, find the encoding matrices Y and W which minimize the following objective:*

$$\underset{Y,W}{\operatorname{argmin}} \ \|\Omega \odot (X - \Psi Y W \Phi)\|_F^2 + \lambda_1 \|Y\|_1 + \lambda_2 \|W\|_1 ,$$

where $\odot$ denotes the element-wise product and λ_1 and λ_2 are sparsity regularization parameters.

In the absence of missing values, we can omit the mask matrix Ω in the fit term, however, we will discuss solutions for this more general version of the objective. The sparse TGSD problem can be viewed as a generalization of matrix completion when missing values are present, and sparse coding when the input is complete. In particular, the missing value objective is a generalization of low-rank matrix completion [8, 57], since choosing trivial identity matrix dictionaries of appropriate sizes reduces our objective to that in matrix completion $\|\Omega \odot (D - AB)\|_F^2$. Different from the above, TGSD can harness the representative power of various dictionaries to capture structures in both rows and columns. Similarly, selecting identity matrix dictionaries in the absence of missing values reduces our objective to matrix factorization with sparsity regularization. It is important to note that our optimization solutions discussed next are applicable to any dictionaries, making TGSD general and flexible to both existing as well as "custom" new dictionaries.

4 OPTIMIZATION SOLUTIONS FOR TGSD

We derive an optimization technique for the missing values objective from Def 3.1 and discuss how it can be customized for decomposition without missing values. Since the objective from Def. 3.1 is jointly convex, we employ Alternating Direction Method of Multipliers (ADMM) [7] to solve it. We first introduce intermediate variables $D = X$, $Z = Y$ and $V = W$ which help ensure that all subproblems have a closed-form solution, and rewrite the objective as:

$$\underset{D,Y,W,Z,V}{\operatorname{argmin}} \ \|D - \Psi Y W \Phi\|_F^2 + \lambda_1 \|Z\|_1 + \lambda_2 \|V\|_1 + \lambda_3 \|\Omega \odot (D - X)\|_F^2 \quad s.t.\ Y = Z, W = V, D = X \tag{2}$$

With some algebraic transformations, the corresponding Lagrangian function is as follows:

$$\mathcal{L} = \|D - \Psi Y W \Phi\|_F^2 + \lambda_1 \|Z\|_1 + \lambda_2 \|V\|_1 + \lambda_3 \|\Omega \odot (D - X)\|_2 + \frac{\rho_1}{2}\left\|Z - Y + \frac{\Gamma_1}{\rho_1}\right\|_F^2 + \frac{\rho_2}{2}\left\|V - W + \frac{\Gamma_2}{\rho_2}\right\|_F^2, \tag{3}$$

where Γ_1 and Γ_2 are the Lagrangian multipliers and ρ_1 and ρ_2 are penalty parameters. Next we derive the individual variable updates for ADMM.

Update D**:** Let $P = \Psi Y W \Phi$, we have the optimization problem for Y as follows:

$$\underset{D}{\operatorname{argmin}} \ \|D - P\|_F^2 + \lambda_3 \|\Omega \odot (D - X)\|_F^2 \tag{4}$$

By taking the gradient and equating it to zero, we have $D = (P + \lambda_3 \Omega \odot X) \oslash (I + \lambda_3 \Omega)$, where $\oslash$ is element-wise division.

Update Y**:** Let $B = W\Phi$, we then have the following optimization problem for Y:

$$\underset{Y}{\operatorname{argmin}} \ \|D - \Psi Y B\|_F^2 + \frac{\rho_1}{2}\left\|Z - Y + \frac{\Gamma_1}{\rho_1}\right\|_F^2 \tag{5}$$

Setting the gradient with respect to Y to zero, we get:

$$2\Psi^T \Psi Y B B^T + \rho_1 Y = 2\Psi^T D B^T + \rho_1 Z + \Gamma_1 . \tag{6}$$

• *Case 1:* If Ψ is a dictionary of orthogonal atoms, we can simplify the above as follows:

$$Y = (2\Psi^T D B^T + \rho_1 Z + \Gamma_1)(2BB^T + \rho_1 I)^{-1} . \tag{7}$$

• *Case 2:* If Ψ is not orthogonal, we cannot solve the problem as outlined above, and thus, develop a more general solution. Note that both BB^T and $\Psi^T\Psi$ are positive semi-definite and symmetric. Let their eigenvalue decomposition be as follows: $\Psi^T\Psi = Q_1\Lambda_1 Q_1^T$ and $BB^T = Q_2\Lambda_2 Q_2^T$, where Q_1, Q_2 are orthonormal eigenvector matrices and Λ_1, Λ_2 are diagonal non-negative eigenvalue matrices. Let Π_1 be the quantity on the right side of Eq. 6, and let us multiply the equation on both sides by the eigenvector matrices as follows:

$$\begin{aligned} &\Pi_1 = 2\Psi^T D B^T + \rho_1 Z + \Gamma_1 \\ &\Rightarrow \Pi_1 = 2Q_1\Lambda_1 Q_1^T Y Q_2 \Lambda_2 Q_2^T + \rho_1 Y \\ &\Rightarrow Q_1^T \Pi_1 Q_2 = 2\Lambda_1 Q_1^T Y Q_2 \Lambda_2 + \rho_1 Q_1^T Y Q_2 \end{aligned} \tag{8}$$

Substituting $E_1 = Q_1^T Y Q_2$ in Eq. 8 we obtain $Q_1^T \Pi_1 Q_2 = 2\Lambda_1 E_1 \Lambda_2 + \rho_1 E_1$, and an element-wise solution for E_1 as follows:

$$[E_1]_{(i,j)} = [Q_1^T \Pi_1 Q_2]_{(i,j)} / 2[\Lambda_1]_{(ii)}[\Lambda_2]_{(jj)} + \rho_1 \tag{9}$$

Finally, we update Y based on E_1: $Y = Q_1 E_1 Q_2^T$.

Update W**:** When we fix other variables and set $A = \Psi Y$, the problem w.r.t. W is reduced to:

$$\underset{W}{\operatorname{argmin}} \ \|D - AW\Phi\|_F^2 + \frac{\rho_2}{2}\left\|V - W + \frac{\Gamma_2}{\rho_2}\right\|_F^2 \tag{10}$$

• *Case 1:* For orthogonal Φ we can set the gradient w.r.t. W to zero, obtaining:

$W = (2A^T A + I\rho_2)^{-1}(2A^T X \Phi^T + \rho_2 V + \Gamma_2)$.

• *Case 2:* For non-orthogonal Φ, we get $W = Q_3 E_2 Q_4^T$, where $E_2(i,j) = [Q_3^T \Pi_2 Q_4]_{i,j} / 2[\Lambda_4]_{ii}[\Lambda_3]_{jj} + \rho_2$ and (Q_3, Λ_3 and (Q_4, Λ_4) are the (eigenvector, eigenvalue) matrices of $A^T A$ and $\Phi\Phi^T$, respectively.

Algorithm 1 TGSD (with missing values)

Input: Input X, mask Ω, dictionaries $\{\Psi, \Phi\}$, k, λ_1, λ_2

1: Initialize $Y = Z = \mathbf{1}$, $W = V = \mathbf{1}$
2: **while** not converged **do**
3: $P = \Psi Y W \Phi$
4: $D = (P + \lambda_3 \Omega \odot X) \oslash (I + \lambda_3 \Omega)$
5: $B = W\Phi$
6: $Y = (2\Psi^T D B^T + \rho_1 Z + \Gamma_1)(2BB^T + \rho_1 I)^{-1}$
7: $A = \Psi Y$
8: $W = (2A^T A + I\rho_2)^{-1}(2A^T X \Phi^T + \rho_2 V + \Gamma_2)$
9: $V_{ij} = sign\left(H_{ij}\right) \times max\left(\left|H_{ij}\right| - \frac{\lambda_1}{\rho_2}, 0\right)$
10: $Z_{ij} = sign\left(H_{ij}\right) \times max\left(\left|H_{ij}\right| - \frac{\lambda_1}{\rho_1}, 0\right)$
11: $\Gamma_1^{i+1} = \Gamma_1^i + \rho_1 (Z - Y)$
12: $\Gamma_2^{i+1} = \Gamma_2^i + \rho_2 (V - W)$
13: $i \leftarrow i + 1$
14: Convergence condition: $\left|f^{i+1} - f^i\right| \leq \varepsilon$, where f^{i+1} and f^i are the objective values of Eq. 4 at iterations $i+1$ and i.
15: **end while**

Update Z and V: The problems w.r.t Z and V are:

$$\begin{cases} \underset{Z}{\text{argmin}}\ \lambda_1 \|Z\|_1 + \frac{\rho_1}{2} \left\|Z - Y + \frac{\Gamma_1}{\rho_1}\right\|_F^2 \\ \underset{V}{\text{argmin}}\ \lambda_2 \|V\|_1 + \frac{\rho_2}{2} \left\|W - V + \frac{\Gamma_2}{\rho_2}\right\|_F^2 \end{cases} \tag{11}$$

Closed-form solutions are available due to [28]:

$$\begin{cases} Z_{ij} = sign\left(H_{ij}^{(1)}\right) \times max\left(\left|H_{ij}^{(1)}\right| - \frac{\lambda_1}{\rho_1}, 0\right) \\ V_{ij} = sign\left(H_{ij}^{(2)}\right) \times max\left(\left|H_{ij}^{(2)}\right| - \frac{\lambda_2}{\rho_2}, 0\right), \end{cases} \tag{12}$$

where $H^{(1)} = Y - \frac{\Gamma_1}{\rho_1}$ and $H^{(2)} = W - \frac{\Gamma_2}{\rho_2}$.

The overall TGSD algorithm. We show all updates within the overall optimization procedure in Alg. 1. We repeat updates from Step 3 to Step 12 until convergence. We demonstrate experimentally that key hyper-parameters like the number of components k can be in a supervised manner by cross-validation. A similar approach can be employed for the sparsity regularizers λ_1 and λ_2.

The complexity of TGSD is dominated by the matrix inversions in Step 6 and 8. Although the complexity of a quadratic matrix inversion is in general cubic, in practice our overall running time is practical due to fast convergence and the ability to work with reduced dictionaries as demonstrated in scalability experiments in Fig. 6. The optimization procedure can be minimally altered for the case without missing values by setting Ω to an all-ones matrix and removing the optimization of D from step 4, in which the overall complexity remains the same. For orthogonal dictionaries we can use the more efficient updates from step 8 and step 6 for W and Y, respectively. When either of the dictionaries (graph or time) is non-orthogonal, we need to work with the general solutions employing eigenvalue decomposition (Case 2 in the updates of Y and W). Non-orthogonal versions come with extra cost due to the eigendecompositions, however the overall complexity remains unchanged since the inversions remain the most costly steps.

5 DICTIONARIES FOR TGSD

Our framework can flexibly accommodate many graph and time dictionaries. For the purposes of evaluation we employ several popular alternatives listed in Tbl. 1. We experiment with multiple versions of TGSD employing different combinations of graph and time dictionaries. Our naming convention specifies them in order. For example, when employing GFT for Ψ and DFT for Φ, we denote

	Graph dictionaries		Temporal dictionaries		
	GFT (G)	Wavelet (W)	DFT (D)	Ramanujan (R)	Spline (S)
Orthogonal	✓	✓	✓		
Parameter-free	✓	✓	✓		

Table 1: Summary of dictionaries we experiment with.

our method: TGSD G+D. We next provide a brief definition of the dictionaries and refer to relevant work for more details.

The **Graph Fourier Transform (GFT) [13]** basis consists of the eigenvectors U of the graph Laplacian matrix L, where $L = U\Lambda U^T$. Graph signal processing draws a parallel between GFT and the discrete Fourier transform (DFT) where small eigenvalues in Λ correspond to "low-frequency" components as they tend to identify larger regions in the graph structure. These "low-frequency" can also be used capture higher order dependencies between nodes while the "high-frequencies" capture more local dependencies [37]. GFT is orthonormal since U is an eigenvector matrix.

Graph-Haar Wavelets [11] have been adopted for many graph data analytics tasks [11] and are central to one of our baselines Gems-HD [50]. An orthonormal basis is computed by thresholding the Fiedler vector, obtaining recursive bisections of the graph. Let V' be a subset of nodes obtained in the recursive partitioning tree and V_1' and V_2' the two subsets obtained by thresholding the Fiedler vector at 0 for the subgraph induced by V'. The basis function $\phi'(v)$ for V' is defined as:

$$\phi'(v) = \begin{cases} \frac{\sqrt{|V_2'|}}{\sqrt{|V_1'|}\sqrt{|V_1'|+|V_2'|}} & \text{if } v \in V_1', \\ -\frac{\sqrt{|V_1'|}}{\sqrt{|V_2'|}\sqrt{|V_1'|+|V_2'|}} & \text{if } v \in V_2', \\ 0 & \text{if } v \notin V \end{cases} \tag{13}$$

The **Discrete Fourier Transform (DFT) [45]** dictionary W for temporal signals of length N is defined as:

$$W = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{(N-1)} & \omega^{2(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{bmatrix}, \tag{14}$$

where $\omega = e^{\frac{-2\pi i}{N}}$ and i is the imaginary unit. This basis is unitary.

The **Ramanujan periodic dictionary [45]**, similar to DFT, is applicable to periodic signals and is constructed by stacking period-specific sub-matrices of varying width $R = \left[\Phi_1, .., \Phi_{g_{max}}\right]$, where g_{max} is the a maximum modeled period and Φ_i is the periodic basis of period g_i. Period-specific matrices $\Phi_g = \left[D_{d_1}, D_{d_2}, ...D_{d_K}\right]$ have columns determined by the divisors $\{d_1, d_2, ...d_K\}$ of g. $D_{d_i} \in \mathbb{R}^{g \times \phi(d_i)}$ is a periodic basis for period d_i of the following circulant matrix form:

$$D_{d_i} = \begin{bmatrix} C_{d_i}(0) & C_{d_i}(g-1) & \dots & C_{d_i}(1) \\ C_{d_i}(1) & C_{d_i}(0) & \dots & C_{d_i}(2) \\ \dots & \dots & \dots & \dots \\ C_{d_i}(g-1) & C_{d_i}(g-2) & \dots & C_{d_i}(0) \end{bmatrix}, \tag{15}$$

where the number of columns, $\phi(d_i)$ denotes the Euler totient function. Elements $C_{d_i}(g)$ are computed as the Ramanujan sum:

$$C_{d_i}(g) = \sum_{k=1, gcd(k,d_i)=1}^{d_i} e^{j2\pi kg/d_i}, \tag{16}$$

where $gcd(k, d_i)$ is the greatest common divisor of k and d_i. This dictionary is not orthogonal.

The **Spline dictionary [16]** is applicable for encoding smoothly-evolving time series and can be constructed by employing B-splines

Task	Baselines	Quality metric
1. Decomposition	MCG, LRDS, GEMS-HD, SVD	RMSE v.s. model size
2. Imputation	MCG, LRDS, GEMS-HD, BRITS	RMSE v.s. % missing
3. Interpolation	MCG, LRDS, GEMS-HD, BRITS	RMSE v.s. % missing
4. Node clustering	CCTN, PCA	Accuracy
5. Period detection	NPM, FFT, AUTO	Accuracy

Table 2: Summary of evaluation tasks, baselines and metrics.

$B_{i,d}(u)$, defined by the Cox-de-Boor formula:

$$B_{i,p} = \frac{u - u_i}{u_{i+p} - u_i} B_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} B_{i+1,p-1}(u),$$

where $B_{i,0} = 1$ if $u_i \leq u < u_{i+1}$, and 0 otherwise. $B_{i,d}(u)$ is non-zero in the range of $[u_i, u_{i+d+1})$. This dictionary is non-orthogonal.

6 EXPERIMENTAL EVALUATION

We evaluate TGSD on five tasks listed in Tbl. 2 and compare against a total of 10 baselines across tasks (detailed in Sec. 6.2) on 6 datasets (Tbl. 3). Our goal is to test the accuracy, scalabilty and conciseness of our model. We quantify model conciseness in terms of number of non-zero values (nnz) in its representation and measure scalability in terms of running time of single-core MATLAB implementations.

6.1 Datasets.

Table 3 shows the statistics of our real-world and synthetic datasets. The graph structure in our **Synthetic** data consists of 7 overlapping groups (on average 10% of nodes in each group belong to other groups). We generate independently periodic time series for each group similar to the protocol in [45]. We scale node signals randomly (uniform in [1, 10]) and add Gaussian noise at $SNR = 10$.

We also employ 2 real-world data-sets for reconstruction, imputation and interpolation experiments; and 3 more real-world data sets with ground-truth communities to evaluate clustering. The **Bike** [1] dataset contains daily bike check-out counts at rental stations in Boston. Pairs of stations are connected by an edge if within approximately 2.22 km. The graph in the **Traffic** [5] dataset corresponds to a highway network where nodes are locations of inductive loop sensors. We use the average speed (at a resolution of 3 hours) at sensors as our evolving graph signal. We normalize both the Bike and Traffic datasets by using the MATLAB's function *normr* which scales each to a norm of 1: $\|X_i\|_2 = 1, \forall i \in [1, n]$.

The **Reality Mining** [14] tracks the number of hourly interactions of 142 people at MIT where an edge between two individuals exists if they interacted at least 50 times. We employ lab group membership (provided in the dataset) as community ground truth. The two **Reddit** datasets [29] are derived from public reddit comments between 2008 and 2015. Undirected user-user edges exist if a user replied at least once to another user's top-level post. **Reddit-epi** consists of 242 users who posted in one of 25 subredits dedicated to popular shows. **Reddit-sp** involves 625 users in 6 sports-related subreddits. Resolution of the datasets is hourly in both cases. The subreddit in which a user participates the most is treated as the ground truth community assignment for that user.

6.2 Baselines

In the decomposition, imputation and interpolation tasks, we employ three baselines: **MCG** [23], **LRDS** [38] and **Gems-HD** [50] (Tbl. 2). **MCG** imputes matrix missing values based on rank minimization and by incorporating a row and a column similarity graph.

Dataset	Nodes	Edges	t	k	Resolution
Synthetic	175-50k	2k-500k	200-50k	7	-
Bike [1]	142	3446	628	NA	1 day
Traffic [5]	1938	5318	720	NA	3 hour
Reality Mining [14]	94	1546	8636	5	1 hour
Reddit-epi [29]	242	1220	3728	25	1 hour
Reddit-sp [29]	625	2872	4325	6	1 hour

Table 3: Summary of evaluation datasets.

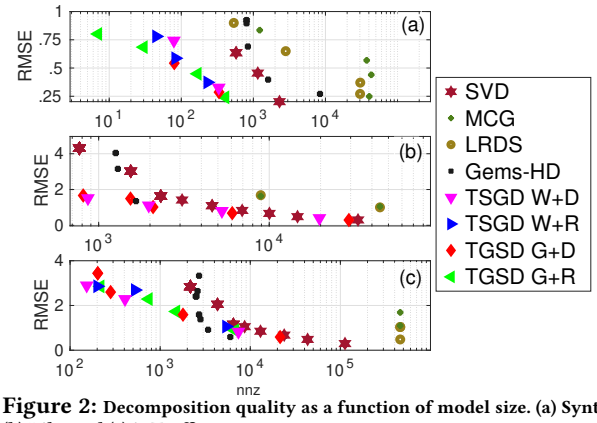


Figure 2: Decomposition quality as a function of model size. (a) Synthetic, (b) Bike and (c) is Traffic.

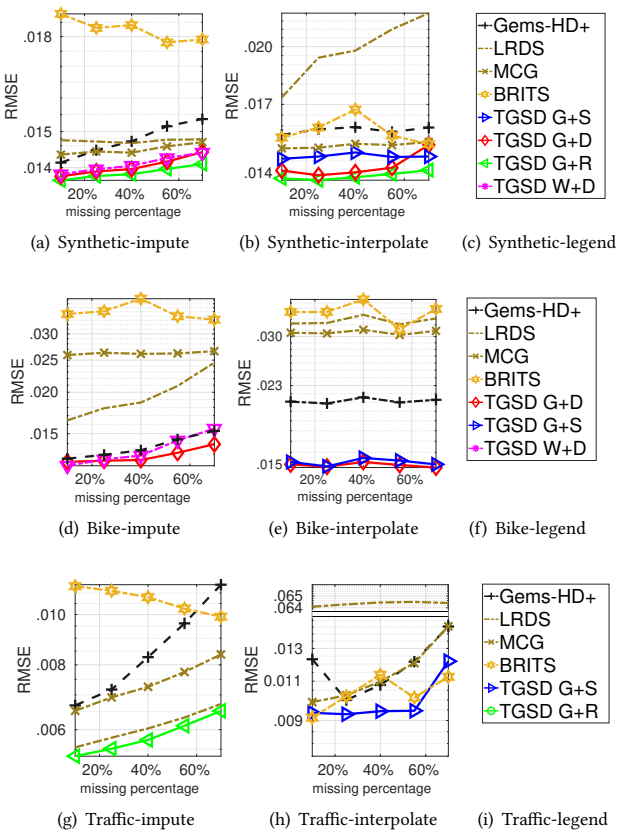
We define the column graph by connecting neighboring time-steps, thus enforcing smoothness in time. **LRDS** also employs rank minimization within a regularizer that combines graph and temporal smoothness. **Gems-HD** is a state-of-the-art graph signal processing method for evolving graph signals. As it does not handle missing values, we impute them as a pre-processing step employing [35] and denote the resulting 2-step method **Gems-HD+**. We also compare to **BRITS** [9] which interpolates missing values in times series by employing a bi-directional recurrent neural network. It also takes advantage of the correlation structure among uni-variates. *We perform exhaustive hyper-parameter search for all baselines (details in the supplement) and report results for the best parameter settings.*

For clustering we compare against the state-of-the-art for graph time series clustering **CCTN** [30], and to **PCA** [21]. We employ k-means to cluster the low-dimensional representations of competing methods, following the protocol from **CCTN** [30]. We employ three baselines for period detection: the state-of-art period learning method **NPM** [45], **FFT** [26], and a method combining auto-correlation and Fourier transform **AUTO** [27]. When the input contains missing values, we first impute using splines [35] resulting in baselines: **NPM+**, **FFT+** and **AUTO+**.

6.3 Graph signal decomposition

We first evaluate the ability of our model to succinctly reconstruct evolving graph signals. We vary the parameters of all competing methods and report the RMSE of their reconstruction as a function of the number of non-zero model coefficients (NNZ) in Fig. 2. We plot only Pareto-optimal points for all methods, i.e. parameter settings resulting in dominated points are omitted for clarity of the figures. We add SVD in the comparison as a "strawman" baseline.

The variants of TGSD dominate all baselines in the small nnz range, since the dual encoding requires fewer coefficients to represents trends in the signal. Since our synthetic datasets have a clear community structure with periodic and synchronous within-community signals, TGSD is able to represent the data using a



(a) Synthetic-impute (b) Synthetic-interpolate (c) Synthetic-legend

(d) Bike-impute (e) Bike-interpolate (f) Bike-legend

(g) Traffic-impute (h) Traffic-interpolate (i) Traffic-legend

Figure 3: Comparison of quality for missing value imputation (a), (d), and (g); and for interpolation (b), (e), and (h)

small number of coefficients (Fig. 2(a)). TGSD's quality similarly dominates alternative in real-world datasets for small nnz.

Gems-HD achieves a good-quality reconstruction when larger models are allowed in the Traffic dataset (Fig. 2(c)) as it employs a small number of the atoms to express a signal on a graph. These atoms select the communities in the graph and a coefficient matrix is used to reconstruct community signals. With sufficient number of coefficients Gems-HD is able to express a large portion of the communities and their conserved temporal patterns in the Traffic dataset. LRDS and MCG require significantly more coefficients on all datasets (the x-axis is logarithmic), as their primary goal is data imputation and their models are not explicitly sparsified. TGSD outperforms SVD following a similar trend with increasing models size. The quality advantage of TGSD is thanks to the structural knowledge encoded in the dictionaries enabling sparser encodings.

6.4 Missing value imputation.

Next we evaluate TGSD's accuracy in predicting missing values in temporal graph signals. We vary the ratio of missing values by randomly removing observations and quantify the accuracy (as RMSE) of competing techniques to predict them. We perform a dataset-specific parameter search for each method (details in the supplement) and report average accuracy of 5 sets of removed values for each missing percentage level in Figs. 3(a), 3(d), 3(g).

A version of TGSD outperforms all baselines across datasets. Different dictionaries have advantages in specific settings, however. On *Synthetic* data (Fig. 3(a)) all dictionaries for TGSD outperform baselines, but the GFT + Ramanujan combination dominates since signals have strong periodicity and the Ramanujan encoding is known to be advantageous in such scenarios [29]. MCG and LRDS are the next best methods and follow similar trends. They are both designed for missing value imputation based on low rank and graph smoothness regularizers. Gems-HD+ performs well for small number of missing values but degrades rapidly as the spline-based imputation treats time series independently and does not take advantage of the graph structure which is critical when many values are missing. BRITS's average behavior exhibits an unexpected downward trend, however, the variance of this method is very large with the standard deviation being equivalent 24% of average RMSE for all runs and especially so for higher missing percentage (27%). For context TGSD G+R had an average standard devation of 1%. The tens of thousands of parameters in BRIT's model may cause it to overfit noise. In the *Bike* dataset (Fig. 3(d)), the DFT dictionary performs best for TGSD and Gems-HD+ is the second best method. This data is aggregated at daily resolution hiding finer temporal patterns and shifting more weight on to the importance of the graph which may explain Gems-HD+ good performance. LRDS, BRITS and MCG perform significantly worse in this setting.

In *Traffic* (Fig. 3(g)), similar to synthetic, the Ramanujan dictionary is preferable for TGSD and LRDS is the best performing baseline. Traffic time series are periodic and smooth along both the graph and time, giving TGSD with Ramanujan dictionary an advantage as it can encode both short- and long-term patterns. LRDS and MCG both employ temporal smoothness regularization rendering them the next best methods.

6.5 Temporal interpolation

We also consider a scenario in which whole temporal slices are missing and quantify the ability of TGSD to interpolate in time. To this end we remove random slices and present average interpolation RMSE. We again perform a dataset-specific grid search for each methods (detailed in the supplement). Similar to value imputation, TGSD outperforms all baseline across datasets. The spline temporal dictionary performs on par with the Ramanujan and DFT and better on the Traffic dataset. This is expected as temporal smoothness becomes important when entire snapshots are missing. In *Synthetic* (Fig. 3(b)) all temporal dictionaries perform well, with the Ramanujan having some advantage. The DFT dictionary performance degrades for large number of missing snapshots, while that of Spline and Ramanujan remains stable. BRITS, MCG, and Gems-HD+ also preform well, while the quality of LRDS is the outlier. LRDS smooths out based on values in the same row and column, however, since here whole columns (slices) are missing, its column smoothing is rendered ineffective.

TGSD has a significant advantage over baselines in the *Bike* dataset (Fig. 3(e)). The use of spline interpolation as a preprocessing step in Gems-HD+ gives it advantage over other baselines, however, since this preprocessing is graph-agnostic, its performance is inferior compared to TGSD. While MCG enforces local temporal smoothness, it ignores long-term dependencies such as periodicity, hence its inferior quality on this daily resolution dataset. BRITS performs well at both low and high fractions of missing slices in *Traffic* (Fig. 3(h)), even outperforming TGSD in those extreme settings by

	TGSD		CCTN [30]		PCA	
Dataset	ACC	Time	ACC	Time	ACC	Time
Synthetic	**85%**	0.6s	14% (20%*)	20s	57%	.02s
Reality Mining	**63%**	80s	55% (50%*)	38m	54%	25s
Reddit-epi	**45%**	11s	35% (32%*)	2.2h	44%	4s
Reddit-sp	**38%**	37s	36%(36%*)	4.7h	35%	6s

Table 4: Comparison of TGSD (G+R in Synthetic, G+D otherwise) and baselines on node clustering accuracy (ACC) and running time (Time).

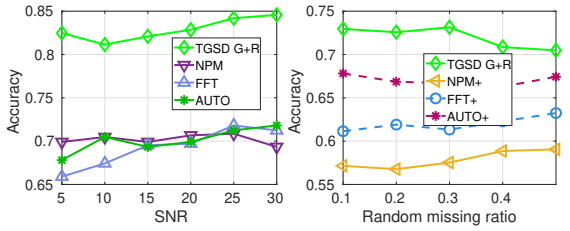
small margins. BRITS' RMSE has a very large standard deviation (12% of average) across runs indicating potential overfitting to the observed values leading also to a non-smooth average RMSE trend. Among the variants of TGSD, the Spline dictionary performs best in this dataset. Gems-HD+ and MCG have similar performance while LRDS performs an order of magnitude worse than all other methods (note that the y axis is discontinuous).

6.6 Clustering node time series

Since our proposed model computes a representation of nodes in ΨY, we next seek to evaluate the utility of the latter for node time series clustering on four datasets with ground truth communities: *Synthetic*, *Reality Mining* [14], *Reddit-epi* [29] and *Reddit-sp*[29]. We compare the performance of TGSD, CCTN and PCA in Tbl. 4 in terms of clustering accuracy and the running time. All methods uses Kmeans with K equal to the number of clusters to obtain cluster labels. TGSD exhibits better clustering quality in all datasets and scales as well as PCA. The periodic dictionaries enable TGSD to capture the accurate patterns in time series, resulting in effective features for clustering. We show two sets of results for CCTN for two values for embedding dimensionality (parameter d in CCTN). The quality value in brackets is for the default $d = 3$, while that outside the brackets is for d set to the ground truth number of clusters in each dataset. The performance of CCTN is second best in the *Reality mining* and *Reddit-sp* and it is the slowest alternative among the 3 alternatives. Different from TGSD, CCTN aims to learn a low-dimensional dictionary to represent all time series. The learned dictionary does not capture interpretable temporal structures such as seasonality, but instead seeks to minimize the reconstruction error in a data-driven manner which may explain its relatively lower performance in the periodic Synthetic dataset. The performance of CCTN is also sensitive to the number of embedding dimensions d. PCA is the second best method on the other two datasets and is the fastest among the three. It does not explicitly consider temporal or graph structures in the data and instead seeks to maximally represent the variance within a pre-specified number of components (set to the true number of communities). While PCA is general and fast, its generality becomes a limitation in the presence of rich graph and temporal structures exploited by TGSD.

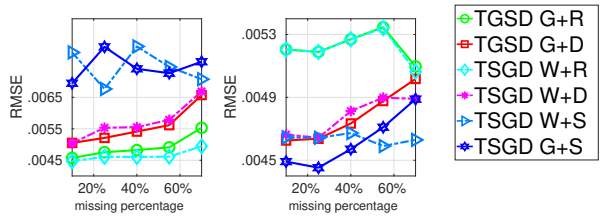
6.7 Period detection

We also evaluate the performance of TGSD for period detection. When using Ramanujan dictionary, we can extract a periodic coefficient matrix $A = \Psi Y W$ and then predict leading periods similar to the approach by Tenneti et Al. [45]. We report the accuracy of the top-k predicted periods in Synthetic data (average of 5 runs), where k is selected based on the actual number of ground truth (known) periods used to generate the node time series. We perform two experiments by varying (i) the SNR and (2) the percentage of



(a) Varying noise (b) Varying missing values

Figure 4: Period learning in synthetic data.



(a) Periodic Signal (b) Smooth Signal (c) Legend

Figure 5: Comparison of dictionary combinations.

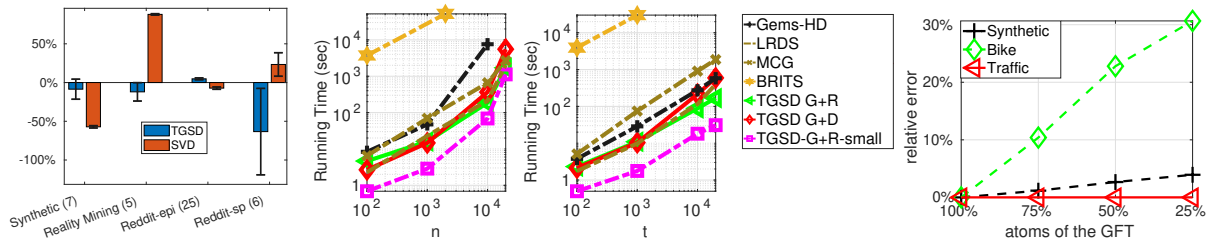
random missing values and report the results in Fig. 4. In both experiments, TGSD exhibits superior quality: 15% improvement over the best baselines and across SNRs, and up to 10% improvement over the best baseline for varying levels of missing values. Unlike all baselines which are designed for time series, TGSD employs jointly the graph an temporal dictionaries making it less sensitive to noise and more robust in the presence of missing values. For the case of missing values, the graph dictionary employed by TGSD allows for better imputation and, in turn, better periodicity estimation. In comparison, baselines employ a two-step approach as they are not designed to handle missing values: they first impute values using spline interpolation in time and then estimate periodicity. Although, baselines employ similar dictionaries for time as those in TGSD (NPM employs the Ramanujan dictionary and DFT and AUTO are based on the Fourier transform), their inability to model relationships among individual time series encoded in the graph structure renders them less accurate than TGSD.

6.8 Dictionary comparison

We compare the dictionary combinations for the missing value imputation task on synthetic data in Fig. 5. In Fig. 5(a) we generate the node signals in the same manner as in the *Synthetic* datasets employed in previous experiments, namely we allow nodes within clusters to share periodic trends. As a result, the Ramanujan dictionary coupled with both GFT and Wavelets in the graph domain performs the best. In contrast, in Fig. 5(b) we generate node signals using random (non-periodic) and smooth trends. As expected, for such locally smooth but non-periodic data, the Spline dictionary works best for encoding the temporal domain. The DFT dictionary provides a middle ground in terms of quality in both cases.

6.9 Estimation of the number of components k

Another important parameter for TGSD is the number of components k for the decomposition. Setting this parameters is a similar problem to determining an optimal rank for other decomposition approaches (e.g., PCA, SVD, NMF), for which a cross-validation



(a) Rank Estimation (b) Nodes (c) Timestamps (d) Legend Scalab. (e) Effect of partial dictionary

Figure 6: Rank detection relative error and variance (TGSD G+R for Synthetic TGSD G+D for all others), scalability comparison as a function of (b) n and (c) t; and (d) quality of TGSD value imputation for partial GFT dictionary.

was shown to perform best among alternatives based on statistical tests and heuristics [46]. In the element-wise k-fold (ekf) cross-validation folds are values are randomly created and removed from a matrix, then imputed and predicted by a reconstruction method with different number of components k [46]. The k leading in the lowest average SSE is predicted as the optimal rank. We extend this rank estimation protocol to TGSD and compare the quality of estimating k using TGSD and SVD [46] employing spline-imputation on *Synthetic*, *Reality Mining* [14], *Reddit-epi, Reddit-sp* [29].

We compare the quality of the two estimates w.r.t. number of ground truth communities employing 5-fold cross validation and present results in 6(a). TGSD outperforms SVD in all but the *Reddit-sp* datasets. TGSD is able to handle missing values directly meaning that is not affected by the quality of a reprocessing imputation scheme used in SVD. It is also able to utilize the graph and temporal dictionaries to further guide its missing value imputation. A similar strategy can be adopted for TGSD's other parameters λ_1 and λ_2.

6.10 Scalability and partial graph dictionaries.

We next evaluate the scalability of competing techniques with the number of nodes and time steps. We add a partial dictionary variant of TGSD, TGSD-G+R-small, in the scalability comparison to quantify the scalability improvement it may offer. Namely, TGSD-G+R-small employs 10% of the lowest-frequency GFT atoms, i.e. the Laplacian eigenvectors corresponding to the smallest eigenvalues.

TGSD-G+R-small scales best among alternatives, while TGSD with the full GFT and Ramanujan dictionaries are the second fastest approach (Fig. 6). TGSD with GFT and DFT is the slowest among TGSD's variants due to the relatively large dictionaries (the full DFT is $\Phi \in \mathcal{R}^{(t\times t)}$ and the full GFT: $\Psi \in \mathcal{R}^{(n\times n)}$). Among the baselines, **MCG** scales significantly worse than **LRDS** with both t and n since the time and graph regularizers employed in the latter retain convexity of the overall objective and allow for a more efficient solver. In contrast, **MCG** employs a nuclear norm to enforce a low rank, resulting in a $O(nt^2 + t^3)$ worst-case complexity, where t is the number of columns. **Gems-HD**'s running time grows quickly with the number of nodes as it employs the graph-Haar wavelets for basis, resulting in a cost of $O(n^2)$ per atom (atom encodings are fit one at a time). **BRITS**, which is based on deep learning, scales orders of magnitude worse than all other methods as it needs to learn many more parameters requiring hundreds of epochs. It is important to note that while all other methods are executed on conventional CPU architecture, the running time results we report for **BRITS** are for execution on a dedicated GPU server with state-of-the-art NVIDIA Tesla V100 GPU card.

While a partial graph dictionary offers scalability improvement (TGSD-G+R-small in Figs. 6(b), 6(c)), it is natural to question if these savings come at the expense of quality. To investigate this trade-off, we compare TGSD's quality on missing value imputation with decreasing subset of the GFT dictionary atoms.

The GFT has ordered columns of frequencies with the first column corresponding to the lowest "frequency" and following columns corresponding to finer partitions, i.e. higher frequencies. We leverage this information by only employing a fixed percentage of the lowest frequency atoms for encoding. We report results for TGSD in Fig. 6(e) for random value imputation at 25% missing values in the *Synthetic*, *Bike* and *Traffic* datasets. The only data set that is significantly impacted by using a reduced GFT is *Bike* while the quality on Synthetic and Traffic is practically unaffected when employing as few as 25% of the atoms. The *Bike* has the smallest graph making every column in the GFT important. In contrast the *Traffic* dataset has by far the largest graph and signals on it can be encoded using very few atoms due to strong local similarity of node behavior (neighboring nodes are consecutive sensors on the same highway and they observe similar speed).

The promising results on scalability and retained quality for partial dictionaries suggest that there exists a significant opportunity for further time savings for TGSD by carefully selecting partial dictionaries for both time and the graph structure. Beyond partial dictionaries and how to sub-select them, automatic dictionary type selection and optimal selection of the number of components k also promise further reduction in the running time and quality improvement for our method. Such considerations is beyond the scope of this paper, though a promising direction for future investigation.

7 CONCLUSION

In this paper we proposed a general framework for dictionary-based decomposition of temporal graph signals, called TGSD. Our algorithm employs an ADMM optimization procedure and can take advantage of multiple existing dictionaries to jointly encode the time and graph extents of temporal graph signals. We performed an exhaustive evaluation of TGSD on five application tasks and demonstrated its effectiveness in both synthetic and real-world datasets. TGSD dominated baselines across tasks. In particular, TGSD achieved 28% reduction in RMSE compared to baselines for temporal interpolation of graph signals when as many as 75% of the observed snapshots were missing. At the same time, TGSD scaled

best with the size of the input with the fastest variation processing 3.5 million data points in under 20 seconds while producing the most parsimonious and accurate decomposition models.

ACKNOWLEDGMENTS

This research is funded by an academic grant from the National Geospatial-Intelligence Agency (Award No. # HM0476-20-1-0011, Project Title: Optimizing the Temporal Resolution in Dynamic Graph Mining). Approved for public release, 21-302. The work is also partially supported by the NSF Smart and Connected Communities (SC&C) grant CMMI-1831547.

REFERENCES

[1] Hubway data visualization challenge:. http://hubwaydatachallenge.org.
[2] L. Akoglu, H. Tong, and D. Koutra. Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery*, 29(3):626–688, 2015.
[3] V. Amelkin, P. Bogdanov, and A. K. Singh. A distance measure for the analysis of polar opinion dynamics in social networks. In *Proc. of the International Conference on Data Engineering (ICDE)*, pages 159–162. IEEE, 2017.
[4] Z. Bar-Joseph. Analyzing time series gene expression data. *Bioinformatics*, 20(16):2493–2503, 2004.
[5] P. Bickel, C. Chen, J. Kwon, J. Rice, P. Varaiya, and E. van Zwet. Traffic flow on a freeway network. In *Nonlinear Estimation and Classification*, pages 63–81. Springer, 2003.
[6] K. M. Borgwardt, H.-P. Kriegel, and P. Wackersreuther. Pattern mining in frequent dynamic subgraphs. In *Proc. of Intl. Conference on Data Mining (ICDM)*, 2006.
[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, Jan. 2011.
[8] R. Cabral, F. De la Torre, J. P. Costeira, and A. Bernardino. Unifying nuclear norm and bilinear factorization approaches for low-rank matrix decomposition. In *2013 IEEE International Conference on Computer Vision*, pages 2488–2495, 2013.
[9] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li. Brits: Bidirectional recurrent imputation for time series. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6775–6785. Curran Associates, Inc., 2018.
[10] D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. In *12th ACM international conference on Knowledge discovery and data mining*, pages 554–560, 2006.
[11] M. Crovella and E. Kolaczyk. Graph wavelets for spatial traffic analysis. In *Proc. of the joint IEEE Conference on Computer and Communications Societies (INFOCOM)*, volume 3, pages 1848–1857. IEEE, 2003.
[12] W. Dong and A. Pentland. A network analysis of road traffic with vehicle tracking data. In *AAAI Spring Symposium: Human Behavior Modeling*, pages 7–12, 2009.
[13] X. Dong, D. Thanou, M. Rabbat, and P. Frossard. Learning graphs from data: A signal representation perspective. *IEEE Signal Proc. Magazine*, 36(3):44–63, 2019.
[14] N. Eagle and A. S. Pentland. Reality mining: sensing complex social systems. *Personal and ubiquitous computing*, 10(4):255–268, 2006.
[15] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE TIP*, 15(12):3736–3745, 2006.
[16] V. Goepp, O. Bouaziz, and G. Nuel. Spline regression with automatic knot selection. *arXiv preprint arXiv:1808.01770*, 2018.
[17] A. Gorovits, E. Gurjal, V. Papalexakis, and P. Bogdanov. Larc: Learning activity-regularized overlapping communities across time. In *ACM International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD 2018)*, 2018.
[18] D. Hallac, J. Leskovec, and S. Boyd. Network lasso: Clustering and optimization in large graphs. In *21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 387–396, 2015.
[19] M. Huisman. Imputation of missing network data: Some simple procedures. *Journal of Social Structure*, 10:1–29, 2009.
[20] P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. In *Forty-Fifth Annual ACM Symposium on Theory of Computing*, page 665–674, 2013.
[21] I. T. Jolliffe and J. Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.
[22] M. Joshi and T. H. Hadi. A review of network traffic analysis and prediction techniques. *arXiv preprint arXiv:1507.05722*, 2015.
[23] V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst. Matrix completion on graphs. *arXiv preprint arXiv:1408.1717*, 2014.
[24] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.
[25] X. Li, M. Zhang, S. Wu, Z. Liu, L. Wang, and P. S. Yu. Dynamic graph collaborative filtering. *arXiv preprint arXiv:2101.02844*, 2021.
[26] Z. Li, B. Ding, J. Han, R. Kays, and P. Nye. Mining periodic behaviors for moving objects. In *16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1099–1108. ACM, 2010.
[27] Z. Li, J. Wang, and J. Han. eperiodicity: Mining event periodicity from incomplete observations. *IEEE Trans. Knowl. Data Eng.*, 27(5):1219–1232, 2015.
[28] Z. Lin, M. Chen, and Y. Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *ArXiv*, abs/1009.5055, 2013.
[29] A. G. Lin Zhang and P. Bogdanov. Perceids: Periodic community detection. In *IEEE ICDM (ICDM)*, 2019.
[30] Y. Liu, L. Zhu, P. Szekely, A. Galstyan, and D. Koutra. Coupled clustering of time-series and networks. In *2019 SIAM ICDM*, pages 531–539. SIAM, 2019.
[31] P. Lorenzo, S. Barbarossa, and P. Banelli. Sampling and recovery of graph signals. In *Cooperative and Graph Signal Processing*, pages 261–282. Elsevier, 2018.
[32] F. Manessi, A. Rozza, and M. Manzo. Dynamic graph convolutional networks. *Pattern Recognition*, 97:107000, 2020.
[33] K. Mitra, S. Sheorey, and R. Chellappa. Large-scale matrix factorization with missing data under additional constraints. In *NIPS*, pages 1651–1659, 2010.
[34] F. Monti, M. M. Bronstein, and X. Bresson. Geometric matrix completion with recurrent multi-graph neural networks. *CoRR*, abs/1704.06803, 2017.
[35] S. Moritz and T. Bartz-Beielstein. imputets: Time series missing value imputation in r. *R Journal*, 9(1), 2017.
[36] L. T. Nguyen, J. Kim, and B. Shim. Low-rank matrix completion: A contemporary survey. *IEEE Access*, 7:94215–94237, 2019.
[37] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst. Graph signal processing: Overview, challenges, and applications. *IEEE*, 106(5):808–828, 2018.
[38] K. Qiu, X. Mao, X. Shen, X. Wang, T. Li, and Y. Gu. Time-varying graph signal reconstruction. *IEEE J. of Selected Topics in Signal Processing*, 11(6):870–883, 2017.
[39] S. Ranu, M. Hoang, and A. Singh. Mining discriminative subgraphs from global-state networks. In *Proceedings of the 19th International Conference on Knowledge Discovery and Data Mining*, pages 509–517, New York, NY, USA, 2013. ACM.
[40] K. R. Rao, D. N. Kim, and J.-J. Hwang. *Fast Fourier Transform - Algorithms and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2010.
[41] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, May 2013.
[42] A. Silva, P. Bogdanov, and A. K. Singh. Hierarchical in-network attribute compression via importance sampling. In *2015 IEEE 31st International Conference on Data Engineering*, pages 951–962. IEEE, 2015.
[43] A. Silva, A. Singh, and A. Swami. Spectral algorithms for temporal graph cuts. In *Proceedings of the 2018 World Wide Web Conference*, pages 519–528, 2018.
[44] H. Tan, Y. Wu, B. Shen, P. J. Jin, and B. Ran. Short-term traffic prediction based on dynamic tensor completion. *IEEE Transactions on ITS*, 17(8):2123–2133, 2016.
[45] S. V. Tenneti and P. P. Vaidyanathan. Nested periodic matrices and dictionaries: New signal representations for period estimation. *IEEE Trans. Signal Processing*, 63(14):3736–3750, 2015.
[46] S. Wold. Cross-validatory estimation of the number of components in factor and principal components models. *Technometrics*, 20(4):397–405, 1978.
[47] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE TPAMI*, 31(2):210–227, 2008.
[48] C. Xiaming, J. Yaohui, Q. Siwei, H. Weisheng, and J. Kaida. Analyzing and modeling spatio-temporal dependence of cellular traffic at city scale. In *Communications (ICC), 2015 IEEE International Conference on*, 2015.
[49] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. *AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018.
[50] Y. Yankelevsky and M. Elad. Finding gems: Multi-scale dictionaries for high-dimensional graph signals. *IEEE Transactions on Signal Processing*, 67(7):1889–1901, 2019.
[51] L. Yao, C. Mao, and Y. Luo. Graph convolutional networks for text classification. *AAAI Conference on Artificial Intelligence*, 33(01):7370–7377, Jul. 2019.
[52] B. Yu, H. Yin, and Z. Zhu. Spatio-temporal graph convolutional neural network: A deep learning framework for traffic forecasting. *CoRR*, abs/1709.04875, 2017.
[53] G. P. Zhang and M. Qi. Neural network forecasting for seasonal and trend time series. *European journal of operational research*, 160(2):501–514, 2005.
[54] L. Zhang and P. Bogdanov. Dsl: Discriminative subgraph learning via sparse self-representation. In *Proceedings of SIAM International Conference on Data Mining (SDM)*, 2019.
[55] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang. A survey of sparse representation: algorithms and applications. *IEEE access*, 3:490–530, 2015.
[56] G. Zheng, Y. Yang, and J. Carbonell. Efficient shift-invariant dictionary learning. In *22nd ACM SIGKDD*, pages 2095–2104, 2016.
[57] Y. Zheng, G. Liu, S. Sugimoto, S. Yan, and M. Okutomi. Practical low-rank matrix approximation under robust l1-norm. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1410–1417, 2012.

SUPPLEMENT

In what follows we discuss how parameters were set in each experiment in the following order: imputation and interpolation, graph signal decomposition, clustering, then period detection. Our goal in providing these experimental details is to facilitate reproduciblity. Tbl.5 contains the search grid for all methods and their parameters used in graph signal decomposition (Sec 6.3), imputation (Sec 6.4) and interpolation (Sec 6.5) experiments. Tbl. 7 shows the estimated optimal parameters for baselines while Tbl. 6 shows the parameters we used in TGSD for all experiments.

BRITS

Data/Task	*Syn. and Bike / imputation and interpolation*
batch size	{54,108,216}
epoch	{500,1000,1500}
hidden layer size	{34,64,128}
Data/Task	*Traffic / imputation*
batch size	{54}
epoch	{500,1000,1500}
hidden layer size	{34,64,128}
Data/Task	*Traffic / interpolation*
batch size	{54}
epoch	{500,1000}
hidden layer size	{34,64,128}

LRDS

Data/Task	*All / decomposition, imputation, interpolation*
Nuclear norm weight	{.1,.5,1}
Smoothness weight	{.1,.5,1}
Convergence weight	{.1,.5,1}
Data/Task	*All / only decomposition*
"zero" threshold	{.0001,.001,.01,.1}

MCG

Data/Task	*All / decomposition, imputation, interpolation*
Nuclear norm weight	{.1,.5,1}
Column graph weight	{.01,.1,1 }
Row graph weight	{.01,.1,1}
Data/Task	*All / only decomposition*
"zero" threshold	{.0001,.001,.01,.1}

GEMS-HD

Data/Task	*Synthetic / decomposition*
Target atom sparsity	[1:10:100]
Target signal sparsity	[1:10:100]
Dictionary size	[1:10:100]
Data/Task	*Bike / decomposition*
Target atom sparsity	[1:10:100],[100:20:200]
Target signal sparsity	[1:10:100],[100:20:200]
Dictionary size	[1:10:100],[100:20:200]
Data/Task	*Traffic / decomposition*
Target atom sparsity	[1:10:100],[100:20:200],[100:20:220]
Target signal sparsity	[1:10:100],[100:20:200],[100:20:220]
Dictionary size	[1:10:100],[100:20:200],[100:20:220]

GEMS-HD+

Data/Task	*All / imputation, interpolation*
Target atom sparsity	{5,10,15,20,25,30}
Target signal sparsity	{5,10,15,20,25,30}
Dictionary size	{5,10,15,20,25,30}

TGSD

Data/Dictionaries/Task	*all/ all/ decomposition*
λ_1	{.001,.01,.1,1}
λ_2	{.001,.01,.1,1},
k	[1:10],[10:10:100]
% of Ψ atoms used	[10%:10%:100%]
Data/Dictionaries/Task	*all/ all/ imputation, interpolation*
λ_1	{.01,.1,1,10}
λ_2	{.01,.1,1,10},
λ_3	{.1,1,10}
k	{5,10,15,20,25,30}

Table 5: Parameter search space for each data set and competing methods in decomposition, imputation and interpolation experiments. Curly braces (e.g. $\{1, 2, 3\}$) indicate a set of values we tested and brackets (e.g. $[1:2:9]$) indicate that we iterate in an interval from the first to the third values by a step specified in the middle value ($[1:2:9]$ represents the values $\{1, 3, 5, 7, 9\}$)

Data-Task	Variation	λ_1	λ_2	λ_3	k
Synthetic-imputation	TGSD W+D	0.01	1	1	5
	TGSD G+D	0.01	0.1	1	5
	TGSD G+R	0.1	0.1	1	5
Synthetic-interpolation	TGSD G+S	1	1	1	5
	TGSD G+D	1	1	1	5
	TGSD G+R	0.01	1	1	30
Synthetic-clustering	TGSD G+R	10	.01	NA	7
Synthetic-period detection	TGSD G+R	.1	.1	10	1
Synthetic-k estimation	TGSD G+R	.01	.001	10	Vary
Bike-impute	TGSD W+D	0.01	0.1	10	25
	TGSD G+D	0.01	0.1	10	30
Bike-interpolate	TGSD G+D	0.01	0.01	10	25
	TGSD G+S	0.01	0.01	10	30
Traffic-impute	TGSD G+R	0.1	0.1	10	5
Traffic-interpolate	TGSD G+S	1	1	10	5
Reddit-sp-clustering	TGSD G+D	1	1	NA	6
Reddit-sp-k estimation	TGSD G+D	1	1	10	Vary
Reddit-epi-clustering	TGSD G+D	2	0.01	NA	25
Reddit-epi-k estimation	TGSD G+D	2	0.01	10	Vary
Reality Mining-clustering	TGSD G+D	5	3	NA	5
Reality Mining-k estimation	TGSD G+D	5	3	10	Vary

Table 6: Parameters for TGSD

• **Imputation (Sec 6.4) and Interpolation (Sec 6.5).** For all method used in imputation and interpolation experiments we preform a task- and data-specific grid search at 25% missing values and use the best parameters for all missing value levels. The specific values searched are shown in Tbl. 5. For BRITS's and LRDS's parameter search spaces we check values both higher and lower than the default parameters set by the authors in the codes they kindly have made publicly available. Due to scalability issues the searched ranges for BRITS are coarser in larger datasets (Tbl. 5) as finer searches did not complete within weeks on state-of-the-art GPU servers. For MCG we explore parameters similar to those for LRDS as they are both low rank methods with smoothing. The authors of GEMS-HD+ list optimal parameters in their publication and we tests alternatives "around" these prescribed values. We report the best found parameters for each experiment in Tbl. 7.

• **Data decomposition (Sec 6.3)**. The parameter settings used in graph signal decomposition can also be found in Tbl. 5. For MCG and LRDS we search the same parameter spaces and add a "zero" threshold parameter which allows us to control the size of the models. For GEMS-HD we expand the search space to obtain a clearer trend for the number of nonzero coefficients necessary for a wide range of RMSE. We test a larger range of values for larger datasets as they require a larger model sizes (nonzero coefficients). We do not include an optimal value table for this search space as the optimal parameters will vary with RMSE and NNZ.

• **Clustering (Sec 6.6).** For CCTN we use the default parameter λ value of 2 provided by the authors and for the dimensionality of the embeddings we report results for both the default parameter 3 and when the dimensionality is set to the ground truth number of clusters in each dataset (Tbl. 4 in the main paper). CCTN's limited scalability made grid search not feasible on our large datasets (it requires close to 5 hours for a single run on the Reddit-sp). For PCA we set the number of dimensions equal to the ground truth number of clusters.

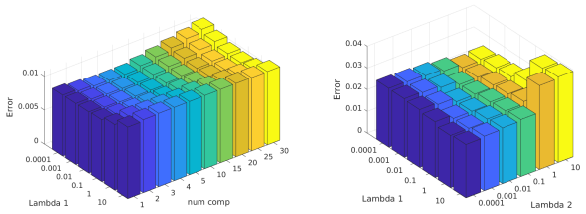
• **Period detection (Sec 6.7).** We set NPM's only parameter—the maximum period in the Ramanujan dictionary—to 50. AUTO's maximum period is set to 100. FFT is parameter-free.

	BRITS			MCG		
	batch size	epoch	hidden layer size	Nuclear norm weight	Column graph weight	Row graph weight
Synthetic-impute	54	1000	128	0.1	1	0.01
Synthetic-interpolate	108	500	128	0.1	0.01	0.1
Bike-impute	216	500	64	1	0.01	0.01
Bike-interpolate	216	500	128	0.5	1	0.01
Traffic-impute	54	1000	34	0.1	0.01	0.01
Traffic-interpolate	54	500	128	0.1	0.01	0.01
	GEMS-HD+			LRDS		
	Dictionary Size	Target Atom sparsity	Target Signal Sparsity	Nuclear norm weight	Smoothness weight	Convergence weight
Synthetic-impute	5	30	15	0.1	0.1	0.1
Synthetic-interpolate	5	5	5	0.1	1	0.1
Bike-impute	20	30	30	1	0.1	1
Bike-interpolate	5	5	5	1	0.1	0.1
Traffic-impute	5	15	30	0.1	0.5	1
Traffic-interpolate	5	5	5	0.1	1	0.1

Table 7: Optimal parameters found by grid search for imputation and interpolation tasks.

	BRITS			MCG			LRDS			GEMS-HD+		
	Avg ± STD	Min	Max	Avg ± STD	Min	Max	Avg ± STD	Min	Max	Avg ± STD	Min	Max
Synthetic-impute	0.0205 ± 0.0029	0.0193	0.0394	0.0154 ± 0.0006	0.0144	0.0164	0.0152 ± 0.0008	0.0145	0.0162	0.0165 ± 0.0008	0.0152	0.0182
Synthetic-interpolate	0.0167 ± 0.0054	0.0152	0.0554	0.0173 ± 0.0015	0.0155	0.0220	0.0212 ± 0.0013	0.0193	0.0229	0.0167 ± 0.0000	0.0167	0.0167
Bike-impute	0.0346 ± 0.0017	0.0324	0.0422	0.0307 ± 0.0026	0.0265	0.0347	0.0162 ± 0.0001	0.0160	0.0164	0.0174 ± 0.0013	0.0160	0.0218
Bike-interpolate	0.0352 ± 0.0021	0.0329	0.0449	0.0320 ± 0.0016	0.0297	0.0347	0.0335 ± 0.0002	0.0331	0.0337	0.0203 ± 0.0000	0.0203	0.0203
Traffic-impute	0.0110 ± 0.00005	0.0110	0.0112	0.0025 ± 0.0000	0.0025	0.0025	0.0112 ± 0.0047	0.0053	0.0167	0.0083 ± 0.0004	0.0076	0.0091
Traffic-interpolate	0.0138 ± 0.0089	0.0097	0.0340	0.0160 ± 0.0000	0.0160	0.0160	0.0702 ± 0.0004	0.0694	0.0706	0.0111 ± 0.000	0.0111	0.0111

Table 8: Average, standard deviation, minimum, and maximum RMSE found in grid search for each data set and competing method in imputation and interpolation experiments.



(a) λ_1 v.s. k (b) λ_1 v.s. λ_2

Figure 7: Parameter sensitivity analysis for 25% random missing imputation in the *Synthetic*

• **Parameter search and optimal parameters for TGSD.** We performed grid search for all variations of TGSD for the graph signal decomposition, imputation, and interpolation experiments (detailed in Tbl.5). For period detection we set K =1, λ_1 = .1, and λ_2 = 0.1. For the estimation of k we set $\lambda_1 = 0.01$, $\lambda_2 = 0.001$. The settings for TGSD in all experiments except graph signal decomposition can be found in Tbl. 6.

While we can employ cross-validation to learn good regularization parameters values by occluding values from X (akin to our approach for k estimation), we are also interested in characterizing the sensitivity of TGSD to these parameters. In Fig. 7, we present the sensitivity of our model to pairs of its key hyperparameters $\{\lambda_1, \lambda_2, k\}$. We fix one parameter and vary the other two, and present the random value imputation error at 25% missing values for the *Synthetic* data set and TGSD G+D. Our model's quality deviates from optimal for high λ_1 and λ_2. Pushing these values too high forces TGSD to overly sparse encodings which "underfit" the data. Reasonably small values in the 0.1 range produce optimal performance and the quality is not sensitive to variations in the vicinity of this value.

• **Spline imputation.** For baselines which do not handle missing values directly we utilized spline imputation as a preprocessing step. To this end, we employ Matlab's "interp1" function. We apply this imputation method to each node timeseries with missing values to reconstruct the latter.

• **An additional baseline of inconclusive performance.** We also evaluated the Geometric Matrix Completion with Recurrent Multi-Graph Neural Networks (MCGNN) [34] as a potential baseline for missing value imputation and temporal interpolation. This work can be intuitively considered as a deep learning extension to [23]. Unfortunately, we were not able to obtain competitive results from MCGNN (at least an order of magnitude worse SSE than all other competing techniques). Hence we do not report it in the experimental results. We tested MCGNN with the same row and column graphs that we adopt for MCG [23] on the synthetic 25% random missing task with grid search over the parameters in Tbl. 9. Note, that MCGNN is a deep learning method and the grid search we performed required substantial computational time (over a week).

Order of row Chebyshev polynomial	{4,5,6}
Order of column Chebyshev polynomial	{4,5,6}
Diffusion steps	{24,32,42}
Number of convolution features	{5,10,15}

Table 9: Parameter search for MCGNN[34]

While we were not able to obtain competitive results, it is possible that the method can be competitive if larger grid search is employed. A possible limitation could be that the column graph may be ineffective in capturing temporal patterns in the convolution architecture. We refrain from making any stronger conclusions based on the experiments we have performed.

• **Code.** An implementation of TGSD is available at the authors' website: http://www.cs.albany.edu/~petko/lab/code.html