SMOOTH SIGNAL RECOVERY ON PRODUCT GRAPHS

Rohan Varma[†], *Jelena Kovačević*[*]

[†]Dept. of Electrical and Computer Engineering, Carnegie Mellon University
[*]Tandon School of Engineering, New York University

ABSTRACT

Product graphs are a useful way to model richer forms of graph-structured data that can be multi-modal in nature. In this work, we study the reconstruction or estimation of smooth signals on product graphs from noisy measurements. We motivate and present representations and algorithms that exploit the inherent structure in product graphs for better and more computationally efficient recovery. These contributions stem from the key insight that smooth graph signals on product graphs can be structured as low-rank tensors. We develop and present algorithms primarily based on two approaches, the first of which is the Tucker decomposition for tensors, while the second is a flexible convex optimization formulation. We further present numerical experiments that exhibit the superior performance of these methods with respect to existing methods for smooth signal recovery on graphs.

***Index Terms*—** graph signal, smooth, recovery, reconstruction, tensor decomposition, low-rank

1. INTRODUCTION

Signal estimation from noisy observations is a well-studied problem in signal processing and has applications for signal inpainting, collaborative filtering, recommender systems and other large-scale data completion problems. Since noise can have deleterious, cascading effects in many downstream tasks, being able to efficiently and accurately reconstruct a signal is of significant importance.

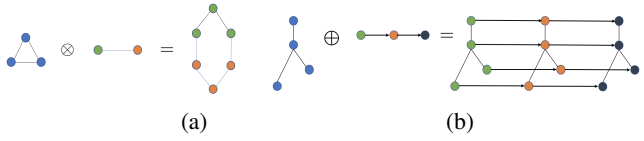
With the explosive growth of information and communication, signals are being generated at an unprecedented rate from various sources, including social networks, citation networks, biological networks, and physical infrastructure [1]. Unlike time-series signals or images, these signals possess complex, irregular structure, which requires novel processing techniques leading to the emerging field of signal processing on graphs [2]. Product graphs are graphs that are composed of smaller *graph atoms*; we motivate how this model is a flexible and useful way to model richer data that may be multi-modal in nature [3–5]. For example, product graph *composition* using a product operator is a natural way to model time-varying signals on a sensor network as shown in Figure 1(b). The graph signal formed by the measurements of all the sensors at all the time steps is supported by the graph that is the product of the sensor network graph and the time series graph. The k^{th} measurement of the n^{th} sensor is indexed by the n^{th} node of the k^{th} copy of the sensor network graph. Multiple types of graph products exist, that is, we can enforce connections across modes in different ways [6]. In the case of the Cartesian product as in Figure 1(b), the measurement of the n^{th} sensor at the k^{th} time step is related to not only to its neighboring sensors at the k^{th} time step but also to its measurements at the $(k-1)^{th}$ and $(k+1)^{th}$ time steps respectively. A social network with multiple communities can also be represented by the Kronecker graph product of the graph that represents a community structure and the graph that captures the interaction between neighbors. In the context of recommender engines where we have user ratings for different entities at different times, we can view this as a signal lying on the Kronecker product of three graphs, the graph relating the different users, the graph relating the different entities, and the time graph.

In graph signal processing, a canonical assumption is that the graph signal is *smooth* with respect to the underlying graph structure, that is, the signal coefficients vary slowly over local neighborhoods of the graph. Hence, constructing a framework for the efficient reconstruction of smooth signals on such product graphs is an important step for tasks such as graph signal recovery, compression, and semi-supervised learning on large-scale and multi-modal graphs. While it is straightforward to apply existing methods and algorithms for smooth graph signal recovery that treat the product graph as a holistic entity, our motivations for this work and consequently our contributions are two-fold. We aim to develop a framework for the reconstruction of smooth signals on product graphs that not only (1) *exploits the inherent structure in product graphs* for better performance, but also affords us (2) *savings in computational complexity*.

2. GRAPH SIGNAL PROCESSING

2.1. Graphs, Graph Signals and Product Graphs

We consider a graph $G = (\mathcal{V}, \mathbf{A})$, where $\mathcal{V} = \{v_0, \dots, v_{N-1}\}$ is the set of nodes and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the graph shift, or a weighted adjacency matrix. $\mathbf{A}$ Represents the connections of the graph G, which can be either directed or undirected. The edge weight $w(n \to m) = \mathbf{A}_{n,m}$ between nodes v_n and v_m is a quantitative expression of the underlying relation between the n^{th} and the m^{th} node, such as a similarity, a dependency, or a communication pattern. If a non-zero edge weight between v_n and v_m exists, we write $v_n \sim v_m$. In this work, we only consider undirected graphs with positive edge weights. We can define the graph Laplacian as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the diagonal degree matrix. Once the node order is fixed, the



(a) (b)

Fig. 1*: (a) Under the Kronecker product, $(u_1, u_2) \sim (v_1, v_2)$ in the product graph if $u_1 \sim v_1$ and $u_2 \sim v_2$. (b) Under the Cartesian product, $(u_1, u_2) \sim (v_1, v_2)$ in the product graph if $u_1 = v_1$ and $u_2 \sim v_2$ or $u_1 \sim v_1$ and $u_2 = v_2$*

Emails: rohanv@andrew.cmu.edu, jelenak@nyu.edu
This work is supported by NSF under grant CCF-1563918

graph signal is written as a vector

$$\mathbf{x} = \left[x_0, x_1, \ldots, x_{N-1}\right]^T \in \mathbb{R}^N.$$

Product graphs are graphs whose adjacency matrices are composed using the *product* (represented by the square symbol $\square$) of the adjacency matrices of smaller *graph atoms*. Consider two graphs $G_1 = (\mathcal{V}_1, \mathbf{A}_1)$ and $G_2 = (\mathcal{V}_2, \mathbf{A}_2)$. The graph product of G_1 and G_2 is the graph $G = G_1 \square G_2 = (\mathcal{V}, \mathbf{A}_1 \square \mathbf{A}_2)$ where $|\mathcal{V}| = |\mathcal{V}_1| \cdot |\mathcal{V}_2|$. The set of nodes $\mathcal{V}$ is the Cartesian product of the sets $\mathcal{V}_1$ and $\mathcal{V}_2$. That is, a node (u_1, u_2) is created for every $u_1 \in \mathcal{V}_1$ and $u_2 \in \mathcal{V}_2$.

Typically, we use one of the Kronecker graph product ($\otimes$, Figure 1(a)), the Cartesian graph product ($\oplus$, Figure 1(b) or the strong graph product ($\boxtimes$) which is a combination of both the Kronecker and Cartesian product to compose a product graphs. Since the product is associative, one can extend the above formulation to define product graphs constructed from multiple graph-atoms. In the following exposition, for clarity and brevity, we only consider the Kronecker product. However, the frameworks and algorithms either hold or can easily be extended to both Cartesian and strong products. We consider a product graph $G = (\mathcal{V}, \mathbf{A}), |\mathcal{V}| = N$, that is constructed from J graph atoms $G_1, G_2, \cdots G_J$, where $G_j = (\mathcal{V}_j, \mathbf{A}^j), |\mathcal{V}_j| = N_j$, using the Kronecker product where $\prod_{j=1}^{J} N_j = N$. We can write the resulting graph shift matrix of the product graph as

$$\mathbf{A} = \mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)} \otimes \cdots \otimes \mathbf{A}^{(J)} = \otimes_{j=1}^{J} \mathbf{A}^{(j)} \tag{1}$$

2.2. Bandlimited Signals and Smoothness

A natural proxy for the smoothness of a signal on a graph is its variation over the graph which can be defined with respect to the graph adjacency matrix such that $\mathrm{S}_{\mathbf{A}}(\mathbf{x}) = \|\mathbf{x} - \mathbf{A}\,\mathbf{x}\|_2^2$ or with respect to the graph Laplacian $\mathrm{S}_{\mathbf{L}}(\mathbf{x}) = \mathbf{x}^T\,\mathbf{L}\,\mathbf{x}$. We note that both these characterizations are similar, and can be used interchangeably in our algorithms. With a slight abuse of notation, we use the latter characterization with respect to the graph Laplacian and denote it by $S(\mathbf{x})$ in the following exposition.

Definition 1. We can define a graph signal $\mathbf{x} \in \mathbb{R}^N$ as being *smooth* on a graph with parameter $\eta \geq 0$, when $\mathrm{S}(\mathbf{x}) \leq \eta \|\mathbf{x}\|_2^2$.

The spectral decomposition of $\mathbf{A}$ is $\mathbf{A} = \mathbf{V}\,\Lambda\,\mathbf{U}$ where the eigenvectors of $\mathbf{A}$ form the columns of the graph Fourier basis $\mathbf{V}$ [7]. We can show that smooth signals are *approximately bandlimited*, that is the majority of their energy lies in the low-frequency subspace spanned by the top K vectors of the graph Fourier basis $\mathbf{V}$ [8–10].

3. RECONSTRUCTION OF SMOOTH SIGNALS

In the basic setting, we assume we measure a noisy or corrupted signal $\mathbf{y}$ and seek to reconstruct $\mathbf{x}$ from $\mathbf{y}$ where $\epsilon \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$.

$$\mathbf{y} = \mathbf{x} + \epsilon$$

Broadly, there are two frameworks for signal reconstruction or estimation. The first, which we refer to as the *synthesis* framework, generally consists of constructing an appropriate basis or dictionary over the graph for the class of signals, and then regressing the observed signal $\mathbf{y}$ over this basis. The second, which we can refer to as the *analysis* framework, typically formulates an optimization problem with a regularizer that penalizes some criterion.

The graph Fourier basis promotes smoothness in the sense that smooth signals are approximately bandlimited with respect to the graph Fourier basis. A straightforward way to recover the signal under the synthesis framework would then be to choose the best K such that we project the signal onto the corresponding bandlimited space. We refer to this method as **GFProj**. Rather than explicitly enforcing bandlimitedness, under the analysis framework, we can formulate a convex optimization problem that minimizes the graph variation of the graph signal by solving the following problem which we refer to as **GTV**: $\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{x}\|_2^2 + \lambda\,\mathrm{S}(\mathbf{x})$. While this is the basic formulation, [11] presents a more flexible framework that can deal with multiple signals as well as outliers. The approaches described in [11] however, do not exploit the inherent structure of product graphs, and instead treat the graph holistically.

3.1. Smooth Graph Signals as Low-Dimensional Tensors

It is straightforward to see that we can leverage the formulations presented above directly on product graphs without incorporating the structure of the product graph. However, in the following discussion, we study the reconstruction of smooth signals on product graphs by exploiting the low-dimensional structure of the signal on the product graph.

A natural way to organize signals on product graphs is by using tensors that can be thought of as generalizations of a matrix to higher dimensions. [12, 13]. In the case of a signal lying on a product of three graphs ,we represent the data by a 3rd order tensor $\boldsymbol{\mathcal{X}}$ where the (i_1, i_2, i_3)-th entry in the tensor $\boldsymbol{\mathcal{X}}$ indicates the signal value corresponding to the node in the product graph corresponding to the tuple of the i_1-th node in $\mathbf{A}_1$, the i_2-th node in $\mathbf{A}_2$, and the i_3-th node in $\mathbf{A}_3$. In the context of recommender engines for example, this would in turn correspond to the i-th user's rating of the j-th entity at the k-th time instant. Hence, a signal $\mathbf{x} \in \mathbb{R}^N$ associated with the product graph defined in 1 can be organized as a J-th order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N_1 \times N_2 \cdots \times N_J}$.

We largely adopt the nomenclature and notation in [12] for tensors. Fibers are the higher order analog of rows and columns in matrix and are obtained by fixing every index but one. The mode-j unfolding of the tensor $\boldsymbol{\mathcal{X}}$, $\boldsymbol{X}_{(j)} \in \mathbb{R}^{N_j \times (N/N_j)}$ arranges the mode-j fibers as the columns of the matrix. The j-mode (matrix) product of a tensor $\boldsymbol{\mathcal{X}}$ with a matrix Φ denoted by $\boldsymbol{\mathcal{X}} \times_j \Phi$ corresponds to multiplying each mode-j fibre by Φ. Let $\bigtimes_{j=1}^{J} F_j$ be short-form for multiplying a tensor along each mode by $\boldsymbol{F}_j$. That is, $\mathcal{G}(\bigtimes_{j=1}^{J} F_j) = \mathcal{G} \times_1 \boldsymbol{F}_1 \times_2 \boldsymbol{F}_2 \cdots \times_J \boldsymbol{F}_J$. The CP-rank of a tensor is defined as the minimum number of rank-one tensors that generate $\boldsymbol{\mathcal{X}}$ as their sum. In general, computing the CP-rank of a tensor is difficult, in fact, it is an NP-hard problem. An alternative notion of the rank of a tensor, the *n-rank*,is the tuple of the ranks of the mode-n unfoldings which is easy to compute and yields a greater degree of flexibility. As a result, in this work, we only consider the n-rank of a tensor to quantify the low dimensional structure of the tensor. Any signal $\mathbf{x}$ on a product graph $\mathbf{A}$ can be decomposed as $\mathbf{x} = \sum_{i=1}^{R} \mathbf{x}_i^{(1)} \otimes \mathbf{x}_i^{(2)} \cdots \otimes \mathbf{x}_i^{(J)}$ such that each $\mathbf{x}^{(j)}$ lies on the respective graph atom $\mathbf{A}_j$. We can study the decomposition of smooth signals on product graphs and show how an extension of the smoothness assumption to signals on product graphs leads to the corresponding tensor $\boldsymbol{\mathcal{X}}$ possessing a low-dimensional structure. We can then also show that the j-mode fibers of the tensor $\boldsymbol{\mathcal{X}}$ corresponding to $\mathbf{x}$ are smooth with respect to the j-th graph atom $\mathbf{A}_j$. As a result, in the following algorithms and frameworks, we exploit this low-dimensional structure of the graph signal tensor for smooth signal recovery.

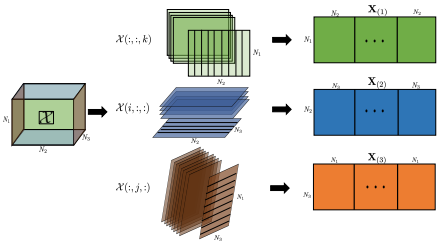


Fig. 2*: Mode unfoldings of a third-order tensor along each of its three modes.*

3.2. Reconstruction via Smooth Tucker Decomposition

Similarly to matrix factorization, PCA, and in graph signal processing, transforms such as spectral graph wavelets and the graph Fourier transform, tensor decomposition allows us to detect latent structure in graph data. The Tucker decomposition decomposes a tensor into a *core tensor* and multiple matrices which correspond to different core scalings along each mode. Therefore, the Tucker decomposition can be seen as a higher-order PCA [12–14]. For a J-th order tensor, the Tucker decomposition approximates a tensor $\boldsymbol{\mathcal{X}}$ with a core-tensor $\boldsymbol{\mathcal{G}}$ and J column-wise orthonormal factor matrices $\boldsymbol{F}_j \in \mathbb{R}^{N_j x R_j}$, $P_j \leq N_j$ $j = \{1, \cdots, J\}$ such that $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{G}}(\bigtimes_{j=1}^{J} F_j)$. Under such a decomposition, the n-rank of $\boldsymbol{\mathcal{X}}$ is simply the tuple of the ranks of the mode-j unfoldings, $(R_1, R_2 \cdots R_J)$. We note however that, in general, we need to estimate or fix the n-rank beforehand which is often difficult or unwieldy.

3.2.1. Synthesis

We now formulate the direct analog of the synthesis approach on a single graph in **GFProj**, where we project the signal onto the low-frequency bandlimited subspace spanned by the graph Fourier basis vectors, to the Tucker decomposition and product graphs. We note that by setting each of the factor matrices equal to the leading R_j columns of the GFT basis $\mathbf{V}^{(j)}$ of the graph atoms $\mathbf{A}_j$, we can enforce bandlimitedness of the product signal on the graph. That is, by setting $\boldsymbol{F}_j = \mathbf{V}^{(j)}_{R_j}$ we can explicitly enforce smoothness of the graph signal on the product graph. We call this algorithm **TD-S**.

3.2.2. Analysis

Under the analysis framework, we now formulate an optimization problem that enforces in addition to the Tucker decomposition structure described above, a smoothness regularizer. Particularly, given a set of graph signals on a product graph, we aim to find a low-rank decomposition that explicitly enforces smoothness not only across edges within the same mode but also across modes of the tensor or product graph. We can define the optimization problem as follows:

$$\begin{aligned} \underset{\boldsymbol{\mathcal{X}}}{\arg\min} \quad & \|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{G}}(\bigtimes_{j=1}^{J} F_j)\|_F^2 + \lambda g(\boldsymbol{F}_i) + \gamma h(\boldsymbol{F}_i) \\ \text{subject to} \quad & g(\boldsymbol{F}_i) = \sum_{j=1}^{J} tr(\boldsymbol{F}_j^T \, \mathbf{L}_j \, \boldsymbol{F}_j), \\ & h(\boldsymbol{F}_i) = tr((\otimes_{j=1}^{J} F_j)^T \, \mathbf{L} (\otimes_{j=1}^{J} F_j)), \\ & \boldsymbol{F}_i^T \boldsymbol{F}_i = \boldsymbol{I}, \forall i \end{aligned} \tag{2}$$

$g(\cdot)$ enforces smoothness within modes while $h(\cdot)$ enforces smoothness across modes of the tensor or product graph. Since the above formulation is convex over each of the variables we are optimizing over, we can solve it in an alternating fashion by solving for each of $\boldsymbol{F}_j$ while leaving the other factor matrices $\boldsymbol{F}_{(-j)}$ fixed. Due to lack of space, we omit detailed derivations and only present the framework in Algorithm 1 which we refer to as **TD-A**. It is closely related to and is a generalization of previous work in [15].

Algorithm 1 (TD-A): Tucker Decomposition via Alternating Least Squares

Inputs: $\boldsymbol{\mathcal{Y}}$, $R_j, \forall j \in \{1 \cdots J\}$ and parameters λ, γ
Initialize:
  $\boldsymbol{F}_j^{(0)} = I, \forall j$
repeat
  for `j ← 1 to J` **do**
    $\boldsymbol{M}_j^{(k+1)} \leftarrow Y_{(j)} \bigotimes_{j=1, j \neq i}^{J} \boldsymbol{F}_j^{(k)}$
    $u_j \leftarrow \prod_{j=1, j \neq i}^{J} \text{tr}(\boldsymbol{F}_j^{(k)T} \, \mathbf{D}_j \, \boldsymbol{F}_j^{(k)})$
    $v_j = \prod_{j=1, j \neq i}^{J} \text{tr}(\boldsymbol{F}_j^{(k)T} \, \mathbf{A}_j \, \boldsymbol{F}_j^{(k)})$
    $\boldsymbol{H}_j^{(k+1)} \leftarrow \boldsymbol{M}_j^{(k+1)} \boldsymbol{M}_j^{(k+1)T} - (\lambda + \gamma u_j) \, \mathbf{D}_j + (\lambda + \gamma v_j) \, \mathbf{A}_j$
    $\boldsymbol{F}_j^{(k+1)} \leftarrow$ top R_j eigenvectors of $\boldsymbol{H}_j^{(k+1)}$
  end for
  $k \leftarrow k + 1$
until convergence
$\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{Y}}(\bigtimes_{j=1}^{J} \boldsymbol{F}_j^{(k)T})$

3.3. Reconstruction via the Nuclear Norm of Unfoldings

In the algorithms presented in Section 3.2, we saw that we needed to fix or estimate the n-rank beforehand which can often be inconvenient. In this section, we alleviate this inflexibility by presenting a more direct optimization formulation. The sum of each of the ranks of the mode-j unfoldings in the n-rank tuple $\sum_{j=1}^{J} R_j$ has been proposed as a proxy for the n-rank [16–18]. We can then use the nuclear norm as a convex surrogate for the rank as is done in many matrix completion problems [19]. We also enforce smoothness of each of the mode-j fibers with respect to the j-th graph atom and define the following convex optimization problem:

$$\underset{\boldsymbol{\mathcal{X}}}{\text{minimize}} \quad \|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2 + \sum_{j=1}^{J} [\alpha \, \text{tr}(\boldsymbol{X}_{(j)}^T \, \mathbf{L}_j \, \boldsymbol{X}_{(j)}) + \beta \|\boldsymbol{X}_{(j)}\|_*] \tag{3}$$

We solve this via the alternating direction method of multipliers (ADMM) framework for separable optimization problems [16, 20]. Towards this, we introduce J tensor variables $\boldsymbol{\mathcal{Z}}_1, \cdots \boldsymbol{\mathcal{Z}}_J$ which represent the J different mode-j unfoldings $\boldsymbol{X}_{(1)}, \cdots, \boldsymbol{X}_{(J)}$ of the tensor $\boldsymbol{\mathcal{X}}$ such that the mode-j unfolding of $\boldsymbol{\mathcal{Z}}_j$, $\mathbf{Z}_{j,(j)} = \boldsymbol{X}_{(j)}, \forall j \in \{1, 2, \cdots J\}$. We can rewrite (3) in the form $f(\boldsymbol{\mathcal{X}}) + \sum_{j=1}^{J} g_j(\boldsymbol{\mathcal{Z}}_j)$:

$$\begin{aligned} \underset{\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}}_j}{\text{minimize}} \quad & \|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2 + \sum_{j=1}^{J} [\alpha \, \text{tr}(\mathbf{Z}_{(j)}^T \, \mathbf{L}_j \, \mathbf{Z}_{(j)}) + \beta \| \, \mathbf{Z}_{(j)} \, \|_*] \\ \text{subject to} \quad & \boldsymbol{\mathcal{Z}}_j = \boldsymbol{\mathcal{X}}, \forall j \end{aligned} \tag{4}$$

We can then write the augmented Lagrangian where $\boldsymbol{\mathcal{W}}_j$ are the

Lagrange variables and μ is the penalty parameter as:

$$\underset{\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}}_{\{j\}}, \boldsymbol{\mathcal{W}}_{\{j\}}}{\text{minimize}} \quad \|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2 + \sum_{j=1}^{J} [\alpha \operatorname{tr}(\mathbf{Z}_{j,(j)}^T \mathbf{L}_j \mathbf{Z}_{j,(j)}) + \beta \| \mathbf{Z}_{j,(j)} \|_* - < \boldsymbol{\mathcal{W}}_j, \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Z}}_j > + \frac{\mu}{2} \|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Z}}_j\|_F^2] \tag{5}$$

Due to limitations on space, we do not present detailed derivations when we minimize 5 over $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Z}}_j$ respectively. However, we note that the subproblem when solving for $\boldsymbol{\mathcal{Z}}_j$ is

$$\underset{\mathbf{Z}_{j,(j)}}{\text{minimize}} \quad \alpha \operatorname{tr}(\mathbf{Z}_{j,(j)}^T \mathbf{L}_j \mathbf{Z}_{(j)}) + \beta \| \mathbf{Z}_{j,(j)} \|_* + < \mathbf{W}_{j,(j)}, \mathbf{Z}_{j,(j)} > + \frac{\mu}{2} \|\boldsymbol{X}_{(j)} - \mathbf{Z}_{j,(j)}\|_F^2 \tag{6}$$

We can solve this convex problem by generalized gradient descent [21] by solving a proximity function in each step with respect to the nuclear norm. We define $\mathcal{D}_\tau(\boldsymbol{C})$ to be the operator that shrinks the singular values of $\boldsymbol{C}$ by soft-thresholding the singular values of $\boldsymbol{C}$ by τ. We call the resulting sub-algorithm **GD-Z** and give an overview in Algorithm 2. We refer to the overall algorithm as **NNFold**, the pseudocode for which is presented in Algorithm 3.

Algorithm 2 (**GD-Z**): Gradient Descent Algorithm for 6

Inputs: $\mathbf{W}_{j,(j)}, \boldsymbol{X}_{(j)}$, and parameters α, β, μ
Initialize:
  $\mathbf{Z}_{j,(j)} = 0$
repeat until convergence
  Choose step size t by backtracking line search
  $\mathbf{Z}_{j,(j)} \leftarrow \mathcal{D}_{t\beta}(\mathbf{Z}_{j,(j)} - t[2\mu(\mathbf{Z}_{j,(j)} - \boldsymbol{X}_{(j)}) + \mathbf{W}_{j,(j)} + 2\alpha \mathbf{L} \mathbf{Z}_{j,(j)}])$
until termination
return $\mathbf{Z}_{j,(j)}$

Algorithm 3 (**NNFold**): ADMM algorithm for 3

Inputs: $\boldsymbol{\mathcal{Y}}$, and parameters α, β, μ
Initialize:
  $\boldsymbol{X}_{(j)}^{(0)}, \mathbf{Z}_{(j)}^{(0)}, \mathbf{W}_{(j)}^{(0)} = 0, \forall j$
repeat until convergence
  $\boldsymbol{\mathcal{X}}^{(k+1)} \leftarrow \frac{1}{J\mu - 2} \sum_{j=1}^{J} (\boldsymbol{\mathcal{W}}_j^{(k)} + \mu \boldsymbol{\mathcal{Z}}_j^{(k)}) - 2\boldsymbol{\mathcal{Y}}$
  for `j ← 1 to J` **do**
    $\mathbf{Z}_{j,(j)}^{(k+1)} \leftarrow$ **GD-Z**$(\mathbf{W}_{j,(j)}^{(k)}, \boldsymbol{X}_{(j)}^{(k)}, \alpha, \beta, \mu)$
    $\boldsymbol{\mathcal{W}}_j^{(k+1)} \leftarrow \boldsymbol{\mathcal{W}}_j^{(k)} - \mu(\boldsymbol{\mathcal{X}}^{(k+1)} - \boldsymbol{\mathcal{Z}}_j^{(k+1)})$
  end for
  $k \leftarrow k + 1$
until termination
return $\boldsymbol{\mathcal{X}}^{(k)}$

Theorem 1. Algorithm 3 **NNFold** converges if the optimal solution set is nonempty such that every limit point of the sequence $\{\boldsymbol{\mathcal{X}}^{(k)}\}$ is an optimal solution.

Proof. Proof omitted due to lack of space □

4. NUMERICAL EXPERIMENTS

We construct a synthetic ground truth smooth signal on a product graph $\mathbf{A}$ composed using the Kronecker product of a random geometric graph, star graph and a chain graph each of which has 25 nodes. We use a heat diffusion model over the product graph such that the graph signal tensor has varying CP-ranks, $r = 2, 4, 6$. We add noise so the signal we denoise over $\boldsymbol{\mathcal{Y}}$ has an SNR of 5dB. We compare the algorithms **GFProj** (Section 3), **GTV** (Section 3), **TD-S**(Section 3.2.1), **TD-A**(Section 3.2.2), and **NNFold**(Section 3.3). The results are shown in the Table 1. We see that **TD-A**, **NNfold** that exploit the product structure consistently outperform **GFProj** and **GTV** while **NNfold** tends to outperform the Tucker decomposition based methods especially for more complex signals.

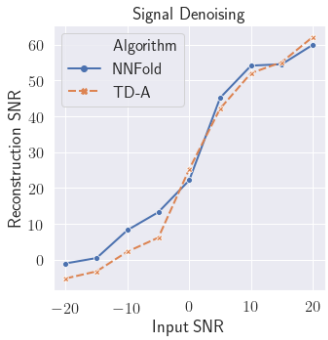


Fig. 3*: Reconstructed signal SNR for varying levels of noise*

	$r = 2$	$r = 4$	$r = 6$
GFProj	1.85e-3	4.6e-2	4.9e-2
GTV	7.82e-4	8.12e-3	8.88e-3
TD-S	1.22e-3	9.14e-3	1.23e-2
TD-A	9.21e-5	6.42e-4	3.2e-3
NNFold	2.02e-4	5.98e-4	9.69e-4

Table 1*: MSE for denoising smooth signal on product graph using each of the 5 discussed algorithms*

For the same synthetic smooth signal, we compare Algorithms **TD-A** and **NNfold** for denoising for different levels of noise. The results are shown in Figure 3. While at high SNR levels, they are both very similar, at low SNR levels or in noisy settings, **NNfold** performs significantly better.

Computational Complexity: The graph atoms $\mathbf{A}^{(j)}$ the product graph is composed of are of size $O(poly(N^{\frac{1}{J}}))$. Since we only perform computationally heavy operations like matrix inversion and singular value decomposition ($O(N^3)$) on structures derived from the graph atoms, the algorithms yield significant computational gains over algorithms that do not exploit the structure in product graphs.

5. CONCLUSIONS

Product graphs are a pragmatic and flexible framework for modeling many kinds of multi-modal real-world graph structured data. In this work, we studied the reconstruction of noisy smooth signals on product graphs. We exploited the low-dimensionality of these signals on product graphs by modeling them as low-rank tensors. Our motivations in this work are two-fold in that in addition to better reconstruction performance, we can also gain computational savings. We presented two main algorithms the first of which is based on the Tucker decomposition while the second is based on the the nuclear norm of the mode-j unfoldings of the tensor. Further, we presented numerical experiments that showcase the superior performance of these algorithms with respect to algorithms that do not exploit the structure of product graphs.

6. REFERENCES

[1] M. Newman, *Networks*. Oxford University Press, 2018.

[2] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[3] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 80–90, 2014.

[4] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 985–1042, 2010.

[5] R. Varma and J. Kovačević, "Sampling Theory for Graph Signals on Product Graphs," *arXiv preprint arXiv:1809.10049*, 2018.

[6] P. M. Weichsel, "The Kronecker product of graphs," *Proceedings of the American mathematical society*, vol. 13, no. 1, pp. 47–52, 1962.

[7] M. Vetterli, J. Kovačević, and V. K. Goyal, *Foundations of signal processing*. Cambridge University Press, 2014.

[8] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete Signal Processing on Graphs: Sampling Theory," *IEEE transactions on signal processing*, vol. 63, no. 24, pp. 6510–6523, 2015.

[9] S. Chen, R. Varma, A. Singh, and J. Kovačević, "Signal representations on graphs: Tools and applications," *arXiv preprint arXiv:1512.05406*, 2015.

[10] ——, "Signal recovery on graphs: Fundamental limits of sampling strategies," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 539–554, 2016.

[11] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovacevic, "Signal Recovery on Graphs: Variation Minimization." *IEEE Trans. Signal Processing*, vol. 63, no. 17, pp. 4609–4624, 2015.

[12] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[13] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[14] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, 2015.

[15] A. Narita, K. Hayashi, R. Tomioka, and H. Kashima, "Tensor factorization using auxiliary information," *Data Mining and Knowledge Discovery*, vol. 25, no. 2, pp. 298–324, 2012.

[16] S. Gandy, B. Recht, and I. Yamada, "Tensor completion and low-n-rank tensor recovery via convex optimization," *Inverse Problems*, vol. 27, no. 2, p. 025010, 2011.

[17] M. Yuan and C.-H. Zhang, "On tensor completion via nuclear norm minimization," *Foundations of Computational Mathematics*, vol. 16, no. 4, pp. 1031–1068, 2016.

[18] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 208–220, 2013.

[19] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, p. 717, 2009.

[20] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[21] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.