Doge-ADMM: A Highly Scalable Parallel Algorithm for Trend Filtering on Graphs

Yinan Bu* Dongrun Wu* Yihong Zuo*

Abstract

Graph trend filtering is a powerful nonparametric method based on minimizing the ℓ_1 norm of discrete graph differences for analyzing data with arbitrary graph structures, aiming to extract piecewise smooth graph signals. However, traditional ADMM algorithms face challenges in balancing the trade-off between convergence rate and the tractability of subproblems, often resulting in computational inefficiency. In this paper, We propose a novel **D**ifferential **O**perator **G**rouping-bas**e**d ADMM algorithm (Doge-ADMM), which leverages differential operator grouping techniques to achieve scalable parallel computation with closed-form solutions for sub-problems while maintaining a fast convergence rate. Experiments on 2-D grid graphs demonstrate that Doge-ADMM achieves up to 30x acceleration over state-of-the-art methods in second-order trend filtering, making it highly suitable for large-scale problems.

1 Introduction

Graph trend filtering (Wang et al., 2016) is a typical non-parametric method based on minimizing the ℓ_1 norm of discrete graph differences dealing with data with arbitrary graph structure to extract piecewise smooth graph signals. Such methods and their variants are widely used in image denoising (Figure 9), collaborative filtering, and recommendation systems (Fan et al., 2022). Many previous works of trend filtering can be viewed as a special case of graph trend filtering with specific graph structures, such as univariate trend filtering (Kim et al., 2009; Steidl et al., 2006) with chain graphs and bivariate trend filtering (Wang et al., 2016) with grid graphs.

Recent works also study the behavior of graph trend filtering with different penalties since the ℓ_1 norm will lead to biased estimation for large coefficients. Varma et al. (2019) consider non-convex penalties like smoothly clipped absolute deviation (SCAD) penalty (Fan and Li, 2001) and

*The author names are listed in alphabetical order.

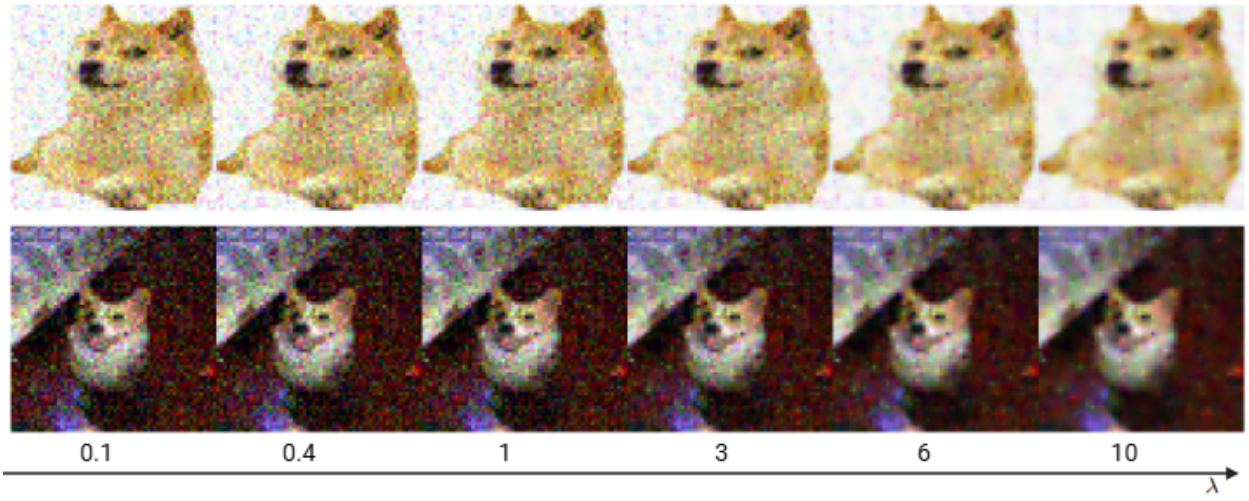


Figure 1: Image denoising using Doge-ADMM graph trend filtering

the minimax concave penalty (MCP) (Zhang, 2010), which alleviate this bias and improve the estimation of large coefficients while maintaining sparsity. Additionally, Fan et al. (2022) explore adaptive penalties that adjust locally based on the graph structure, leading to better preservation of spatially varying trends and improved accuracy in heterogeneous regions of the graph.

All the graph trend filtering methods depend on solving the following optimization problem:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\beta}\|_2^2 + \lambda \,\mathrm{Penalty}(\Delta^{(k+1)}\boldsymbol{\beta}), \tag{1}$$

where $\Delta^{(k+1)}$ is a k-th order differential operator for a graph, and its definition can be found in Section 2.1. A standard way to solve this problem is by using the alternating direction method of multipliers (ADMM) (Boyd et al., 2011). However, traditional ADMM algorithms struggle to balance the trade-off between convergence rate and the tractability of subproblems. In (Wang et al., 2016), when the sub-problems have closed-form solutions, the computation is efficient but convergence can be slow. Methods improving convergence, such as those in Ramdas and Tibshirani (2016), address this by solving more complex sub-problems, but this introduces computational overhead and limits flexibility for broader cases.

In this paper, we present a general framework to solve graph trend filtering that unifies the previous ADMM approaches. We also propose a Differential Operator Grouping-based ADMM algorithm (Doge-ADMM), a novel, highly scalable parallel algorithm for trend filtering on graphs. For simplicity, we only consider the ℓ_1 penalty in this paper, but our methods can extend to arbitrary penalties for problem (1). Our contributions can be summarized as follows:

1. To leverage the trade-off between the algorithm's convergence rate and the complexity of each sub-problem, we propose a differential operator grouping technique to ensure each

sub-problem has a closed-form solution and can be computed in parallel while maintaining the convergence rate.

2. We implement Doge-ADMM for 2-D grid graphs and compare it with other state-of-the-art algorithms for graph trend filtering. Our experiments show that Doge-ADMM outperforms its counterparts, delivering significant acceleration in trend filtering of all orders. For example, in second-order trend filtering, Doge-ADMM achieves 30 times more acceleration than GTF-ADMM (Wang et al., 2016).

Notations. Denote by $[n] = \{1, \ldots, n\}$ for a positive integer n. Denote the cardinality of a set S by $|S|$. We use bold uppercase and lowercase letters to represent matrices and vectors, respectively. For a vector $\boldsymbol{a}$, denote the ℓ_p norm of $\boldsymbol{a} = (a_1, \ldots, a_n)^{\mathrm{T}}$ as $\|\boldsymbol{a}\|_p = (\sum_{i=1}^n a_i^p)^{\frac{1}{p}}$, where $0 \leq p \leq \infty$.

2 Methodology

2.1 Trend Filtering on Graphs

Trend filter is a classical non-parametric method used for identifying and isolating trends within a dataset. It works by smoothing out short-term noise to highlight the underlying long-term trends. The original trend filtering has the form

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\beta}\|_2^2 + \lambda \|D^{(k+1)}\beta\|_1$$

Where $\lambda \geq 0$ is a tuning parameter and $D^{(k+1)}$ is the discrete Differential operator of order k + 1.

$$D^{(1)} = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 & 1 \end{pmatrix}$$

$$D^{(k+1)} = D^{(1)} D^{(k)}$$

The formulation of trend filtering on graphs is a generalization of univariate trend filtering and is first introduced by Wang et al. (2016). For a given graph $G = \{V, E\}$, where $V = [n]$ denotes the vertices set and $E = \{e_1, \cdots, e_m\}$ denotes the edges set, suppose we observe $\boldsymbol{y} = (y_1, \cdots, y_n)^{\mathrm{T}} \in \mathbb{R}^n$ over the nodes. The k-th order trend filtering on graph G can be formulated as the following ℓ_1-penalized optimization problem

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\beta}\|_2^2 + \lambda \|\Delta^{(k+1)}\boldsymbol{\beta}\|_1,$$

where the k-th order differential operator $\Delta^{(k)}$ is defined by recursion

$$\Delta^{(k+1)} = \begin{cases} (\Delta^{(1)})^\top \Delta^{(k)} & \text{for odd } k \\ \Delta^{(1)} \Delta^{(k)} & \text{for even } k. \end{cases} \tag{2}$$

$\Delta^{(1)} \in \mathbb{R}^{m \times n}$ is the first order differential operator whose ℓ-th row is $(0, \cdots, 1, \cdots, -1, \cdots, 0)$, where 1 is at the i-th position and -1 is at the j-th position for $e_\ell = (i, j)$. Then the first order ℓ_1 penalty is equivalent to a penalty on local differences $\sum_{(i,j)\in E} |\beta_i - \beta_j|$. According to Wang et al. (2016), the matrix $\Delta^{(1)}$ can be generated from the degree matrix D and adjacency matrix A: denote Graph Laplacian Matrix $L = D - A$, then we have $\Delta^{(1)} = DL^{\frac{1}{2}}$.

The standard ADMM approach for solving this problem is that by introducing an auxiliary variable $\boldsymbol{\alpha} = \Delta^{(k+1-B)}\boldsymbol{\beta}$, we can rewrite the optimization as

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\beta}\|_2^2 + \lambda \|\Psi^{(B)}\boldsymbol{\alpha}\|_1 \quad subject\ to\ \boldsymbol{\alpha} = \Delta^{(k+1-B)}\boldsymbol{\beta},$$

where

$$\Psi^{(B)} = \begin{cases} \Delta^{(B)} & \text{for even } k+1-B \\ \Delta^{(B-1)} \left(\Delta^{(1)}\right)^T & \text{for odd } k+1-B \end{cases} \tag{3}$$

represents the Differential operator that remains in the original problem. Then augmented Lagrangian can be written as

$$L_\rho(\boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\beta}\|_2^2 + \lambda \|\Psi^{(B)}\boldsymbol{\alpha}\|_1 + \frac{\rho}{2}\|\boldsymbol{\alpha} - \left(\Delta^{(k+1-B)}\boldsymbol{\beta} - \boldsymbol{\theta}\right)\|_2^2.$$

Hence we can derive the ADMM iteration

$$\boldsymbol{\alpha}^{t+1} \leftarrow \arg\min_{\boldsymbol{\alpha}} \lambda \|\Psi^{(B)}\boldsymbol{\alpha}\|_1 + \frac{\rho}{2}\|\boldsymbol{\alpha} - \left(\Delta^{(k+1-B)}\boldsymbol{\beta}^t - \boldsymbol{\theta}^t\right)\|_2^2 \tag{4}$$

$$\boldsymbol{\beta}^{t+1} \leftarrow \left(I + b\rho\left(\Delta^{(k+1-B)}\right)^T \Delta^{(k+1-B)}\right)^{-1} \left(y + \rho\left(\Delta^{(k+1-B)}\right)^T \left(\boldsymbol{\theta}^t + \boldsymbol{\alpha}^{t+1}\right)\right) \tag{5}$$

$$\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t + \left(\boldsymbol{\alpha}^{t+1} - \Delta^{(k+1-B)}\boldsymbol{\beta}^{t+1}\right) \tag{6}$$

When $B = 0$, sub-problem (4) has a closed-form solution $\boldsymbol{\alpha}^{t+1} \leftarrow S_{\lambda/\rho}(\Delta^{(k+1-B)}\boldsymbol{\beta}^t - \boldsymbol{\theta}^t)$, where $S_{\lambda/\rho}$ is a soft-thresholding operator at level λ/ρ. This leads to efficient computing in each iteration. However, this setting will bring a slow convergence rate in practice, hence Ramdas and Tibshirani (2016) then proposed by setting $B = 1$, the convergence rate can be improved significantly since it solves a "harder" problem (4) in each iteration. But this method also has two disadvantages. On the one hand, the sub-problem (4) can not be solved explicitly so an external optimizer will be introduced, which could bring computational inefficiency and error accumulation. On the other hand, this method is difficult to generalize to the general case of B.

2.2 Differential Operator Grouping based ADMM

To address the computational challenge and scalability issue, we propose a Differential Operator Grouping-based ADMM algorithm (Doge-ADMM) for trend filtering on graphs. This method can deal with a general choice of B, and ensure each subproblem has a closed-form solution and can be computed parallel. We begin by defining a *parallelizable differential operator grouping*.

Intuitively, the difficulty of solving sub-problem (4) is caused by the term $\|\Psi^{(B)}\boldsymbol{\alpha}\|_1$. Hence our idea is to decompose $\Psi^{(B)}$ to $\sum_{i=1}^{b}\Psi_i^{(B)}$, where $\Psi_i^{(B)}$ is composed of some rows of $\Psi^{(B)}$ and some zero-vectors. We call $\{\Psi_i^{(B)}, i\in[b]\}$ a parallelizable differential operator grouping when it satisfies for each $i\in[b]$, each column of $\Psi_i^{(B)}$ has at most one non-zero element. The existence of the parallelizable differential operator grouping is trivial since we can let i-th row of $\Psi_i^{(B)}$ be equal to the i-th row of $\Psi^{(B)}$, and the other rows be zero.

Suppose we have a parallelizable differential operator grouping $\{\Psi_i^{(B)}, i\in[b]\}$ already, we then derive the Doge-ADMM algorithm. By introducing auxiliary variables $\boldsymbol{\alpha}_i=\Delta^{(k+1-B)}\boldsymbol{\beta}$, we can rewrite the optimization as

$$\arg\min_{\boldsymbol{\beta}}\frac{1}{2}\|\boldsymbol{y}-\boldsymbol{\beta}\|_2^2+\lambda\sum_{i=1}^{b}\|\Psi_i^{(B)}\boldsymbol{\alpha}_i\|_1 \tag{7}$$

$$subject\ to\ \boldsymbol{\alpha}_i=\Delta^{(k+1-B)}\boldsymbol{\beta},\ i=1,2,\ldots,b. \tag{8}$$

Then the augmented Lagrangian is :

$$\begin{aligned}L_\rho(\boldsymbol{\beta},\boldsymbol{\alpha},\boldsymbol{u}) &= \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{\beta}\|_2^2+\lambda\sum_{i=1}^{b}\|\Psi_i^{(B)}\boldsymbol{\alpha}_i\|_1+\sum_{i=1}^{b}\boldsymbol{u}_i^T\left(\boldsymbol{\alpha}_i-\Delta^{(k+1-B)}\boldsymbol{\beta}\right)+\frac{\rho}{2}\sum_{i=1}^{b}\|\boldsymbol{\alpha}_i-\Delta^{(k+1-B)}\boldsymbol{\beta}\|_2^2\\ &= \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{\beta}\|_2^2+\lambda\sum_{i=1}^{b}\|\Psi_i^{(B)}\boldsymbol{\alpha}_i\|_1+\frac{\rho}{2}\sum_{i=1}^{b}\|\boldsymbol{\alpha}_i-\left(\Delta^{(k+1-B)}\boldsymbol{\beta}-\boldsymbol{\theta}_i\right)\|_2^2\quad\left(\boldsymbol{\theta}_i=\frac{\boldsymbol{u}_i}{\rho}\right),\end{aligned}$$

from which we can derive the Doge-ADMM updates:

$$\boldsymbol{\alpha}_i^{t+1}\leftarrow\arg\min_{\boldsymbol{\alpha}_i}\frac{1}{2}\|\boldsymbol{\alpha}_i-\left(\Delta^{(k+1-B)}\boldsymbol{\beta}^t-\boldsymbol{\theta}_i^t\right)\|_2^2+\frac{\lambda}{\rho}\|\Psi_i^{(B)}\boldsymbol{\alpha}_i\|_1 \tag{9}$$

$$\boldsymbol{\beta}^{t+1}\leftarrow\left(I+b\rho\left(\Delta^{(k+1-B)}\right)^T\Delta^{(k+1-B)}\right)^{-1}\left(y+\rho\sum_{i=1}^{b}\left(\Delta^{(k+1-B)}\right)^T\left(\boldsymbol{\theta}_i^t+\boldsymbol{\alpha}_i^{t+1}\right)\right) \tag{10}$$

$$\boldsymbol{\theta}_i^{t+1}\leftarrow\boldsymbol{\theta}_i^t+\left(\boldsymbol{\alpha}_i^{t+1}-\Delta^{(k+1-B)}\boldsymbol{\beta}^{t+1}\right) \tag{11}$$

Algorithm 1 presents the detailed process of Doge-ADMM algorithm.

Algorithm 1 Doge-ADMM Algorithm

Input: Observed data $\boldsymbol{y}$, regularization parameter λ, block size B, and order k.

Generate: $\Delta^{(k+1-B)}$, $\Psi^{(B)}$ according to equations equation 2 and equation 3.

Grouping: Decompose $\Psi^{(B)}$ as $\Psi^{(B)} = \sum_{i=1}^{b} \Psi_i^{(B)}$.

Initialize: $\boldsymbol{\beta}^0 = \boldsymbol{y}$, $\boldsymbol{\alpha}_i^0 = \Delta^{(k+1-B)}\boldsymbol{y}$, $\boldsymbol{\theta}_i^0 = \mathbf{0}$ for $i = 1, 2, \ldots, b$.

Repeat until convergence:

Update $\boldsymbol{\alpha}_i^{t+1}$ **for** $i = 1, \ldots, b$

$\boldsymbol{\alpha}_i^{t+1} \leftarrow \arg\min_{\boldsymbol{\alpha}_i} \frac{1}{2}\|\boldsymbol{\alpha}_i - \left(\Delta^{(k+1-B)}\boldsymbol{\beta}^t - \boldsymbol{\theta}_i^t\right)\|_2^2 + \frac{\lambda}{\rho}\|\Psi_i^{(B)}\boldsymbol{\alpha}_i\|_1$

Update $\boldsymbol{\beta}^{t+1}$

$\boldsymbol{\beta}^{t+1} \leftarrow \left(I + b\rho\left(\Delta^{(k+1-B)}\right)^T \Delta^{(k+1-B)}\right)^{-1}\left(y + \rho\sum_{i=1}^{b}\left(\Delta^{(k+1-B)}\right)^T\left(\boldsymbol{\theta}_i^t + \boldsymbol{\alpha}_i^{t+1}\right)\right)$

Update $\boldsymbol{\theta}_i^{t+1}$ **for** $i = 1, \ldots, b$

$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^t + \left(\boldsymbol{\alpha}_i^{t+1} - \Delta^{(k+1-B)}\boldsymbol{\beta}^{t+1}\right)$

Output: Final solution $\boldsymbol{\beta}$

We point out that since we assume the parallelizable differential operator grouping, sub-problem (9) can be decomposed into small parallelizable and tractable problems. Specifically speaking, each row of $\Psi_i^{(B)}\boldsymbol{\alpha}_i$ contains different components of $\boldsymbol{\alpha}_i$, so solving sub-problem (9) can be simplified as optimize on each row of $\Psi_i^{(B)}\boldsymbol{\alpha}_i$ separately and all the optimizations can be parallelizable computed. Each small optimization has the same form of the problem

$$\min_{\boldsymbol{x}} \quad \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 + |\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}|$$

and thus has a closed-form solution

$$\begin{cases} \boldsymbol{x} = \boldsymbol{y} - \boldsymbol{a}, & \text{if} \boldsymbol{a}^{\mathrm{T}}\boldsymbol{y} > \|\boldsymbol{a}\|_2^2 \\ \boldsymbol{x} = \boldsymbol{y} + \boldsymbol{a}, & \text{if} \boldsymbol{a}^{\mathrm{T}}\boldsymbol{y} < -\|\boldsymbol{a}\|_2^2 \\ \boldsymbol{x} = \boldsymbol{y} - \frac{\boldsymbol{a}^{\mathrm{T}}\boldsymbol{y}}{\|\boldsymbol{a}\|_2^2}\boldsymbol{a}, & \text{otherwise} \end{cases}$$

where $\boldsymbol{a}$ is the coefficient in rows of $\Psi_i^{(B)}\boldsymbol{\alpha}_i$.

2.3 Special Case: 2D-Grid Graph

In practical implementation, how to find a parallelizable differential operator grouping with a small number of groups is not trivial and depends on the specific graph structure and the order B. We use a special case of 2D-grid graph structure, which is commonly used in image denoising(Pang and Cheung (2017),Faisal et al. (2012)), to show the practical guideline of parallelizable differential operator grouping. We consider the case of $B = 1, 2, 3$.

For $B = 1$, i.e., the matrix Δ implies the first-order difference of $\boldsymbol{\beta}$. The rows of the matrix correspond to the edges of the graph and the columns of the matrix correspond to the nodes. To

give the decomposition, we need to augment the image coordinates by adding a point in the middle of each edge and extending it into the coordinates of this image. Then the rows of Δ can be divided into four groups.

$$\begin{cases} \text{Group 1:}\{\text{edges corresponding to } (i,j) \mid ((i+j)\%4, (i-j)\%4) = (1,1)\} \\ \text{Group 2:}\{\text{edges corresponding to } (i,j) \mid ((i+j)\%4, (i-j)\%4) = (1,3)\} \\ \text{Group 3:}\{\text{edges corresponding to } (i,j) \mid ((i+j)\%4, (i-j)\%4) = (3,1)\} \\ \text{Group 4:}\{\text{edges corresponding to } (i,j) \mid ((i+j)\%4, (i-j)\%4) = (3,3)\} \end{cases}$$

By selecting the remainders of the modulus, the chosen points are positioned at the center of each edge and ensures that the nodes undergoing difference operations within each group do not intersect.

The grouping outcome is shown as below:

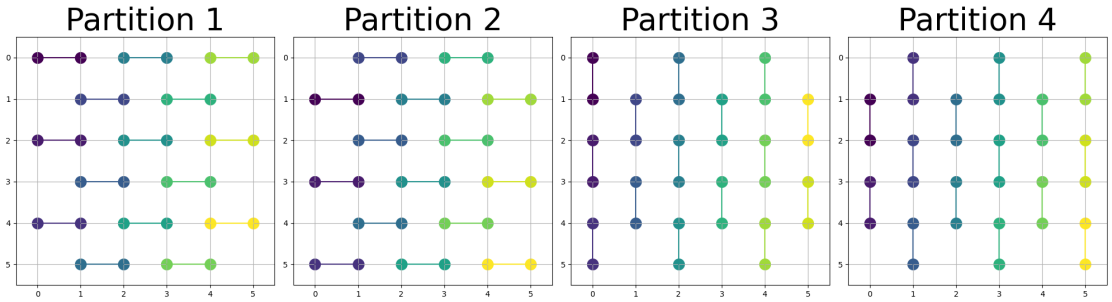


Figure 2: Grouping outcome under the condition B=1

For $B = 2$, the situation is slightly different with $B = 1$. The rows of the matrix also correspond one-to-one with the nodes of the graph. Therefore, we do not need to augment the coordinates of the graph, instead, we directly use the original coordinates. The decomposition can be illustrated as follows:

$$\begin{cases} \text{Group 1:}\{(i,j) \mid ((2i+j)\%5, (i-2j)\%5) = (0,0)\} \\ \text{Group 2:}\{(i,j) \mid ((2i+j)\%5, (i-2j)\%5) = (2,1)\} \\ \text{Group 3:}\{(i,j) \mid ((2i+j)\%5, (i-2j)\%5) = (1,3)\} \\ \text{Group 4:}\{(i,j) \mid ((2i+j)\%5, (i-2j)\%5) = (4,2)\} \\ \text{Group 5:}\{(i,j) \mid ((2i+j)\%5, (i-2j)\%5) = (3,4)\} \end{cases}$$

We use a 6×6 graph as an example, and the grouping outcome can be illustrated as follows:

For $B = 3$, similarly to $B = 1$, we need to augment the coordinates. Then the decomposition

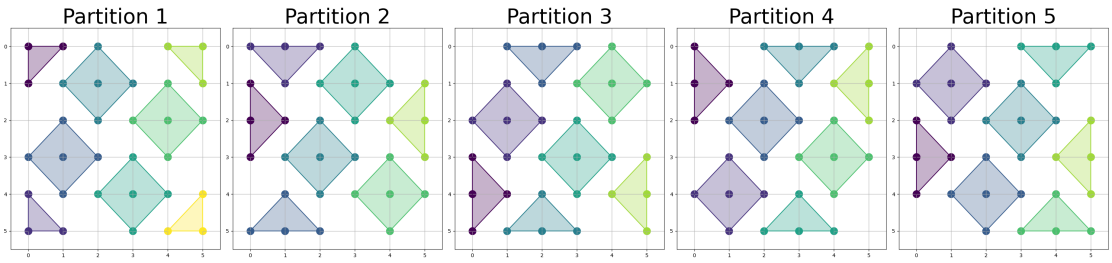


Figure 3: Grouping outcome under the condition B=2

can be given as follows:

$$\begin{cases} \text{Group 1:}\{\text{edges corresponding to } (i,j) \mid ((i+j)\%8, (i-j)\%8) = (1,1)\} \\ \text{Group 2:}\{\text{edges corresponding to } (i,j) \mid ((i+j)\%8, (i-j)\%8) = (3,3)\} \\ \text{Group 3:}\{\text{edges corresponding to } (i,j) \mid ((i+j)\%8, (i-j)\%8) = (5,5)\} \\ \text{Group 4:}\{\text{edges corresponding to } (i,j) \mid ((i+j)\%8, (i-j)\%8) = (7,7)\} \\ \text{Group 5:}\{\text{edges corresponding to } (i,j) \mid ((i+j)\%8, (i-j)\%8) = (3,7)\} \\ \text{Group 6:}\{\text{edges corresponding to } (i,j) \mid ((i+j)\%8, (i-j)\%8) = (5,1)\} \\ \text{Group 7:}\{\text{edges corresponding to } (i,j) \mid ((i+j)\%8, (i-j)\%8) = (7,3)\} \\ \text{Group 8:}\{\text{edges corresponding to } (i,j) \mid ((i+j)\%8, (i-j)\%8) = (1,5)\} \end{cases}$$

Note that, there are actually 16 groups in total, but we only listed 8 of them. This is because with the condition $B = 3$, the differential relationships between nodes center around an edge, forming a hexagonal shape. Therefore, the orientation of the edges (horizontal, vertical) also needs to be considered in the grouping. Considering the symmetry in horizontal and vertical directions, the additional central points can be obtained by performing a symmetric transformation of the original central points, i.e., we only need to switch (i, j) into (j, i), and there come the other 8 groups.

Similar to above, we give the decomposition of a 6×6 graph:

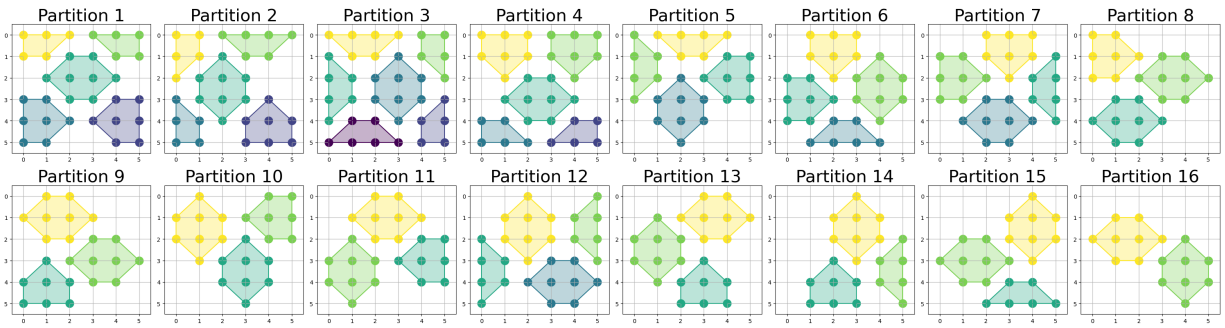


Figure 4: Grouping outcome under the condition B=3

3 Numerical Simulations

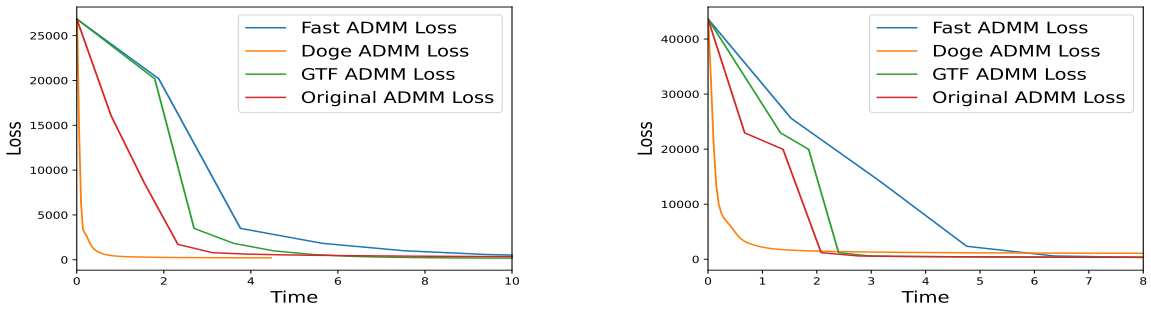
In this section, we present the results of our numerical simulations to demonstrate the effectiveness and efficiency of the proposed method. The experiments focus on evaluating both the computational efficiency and the image-denoising performance of our approach. We compare our method against existing techniques to highlight its advantages in terms of both speed and accuracy. The code is available at github.com/byn1002/Trend-Filtering-on-Graphs.

3.1 Computational Efficiency

In this section, we use image de-noising experiments to show the efficiency of our methods. We compare Doge-ADMM with previous trend filtering graph methods, including the original ADMM method, GTF ADMM method (Wang et al., 2016) and a Fast ADMM method(Ramdas and Tibshirani, 2016).

Figure 5 shows the relationship between the loss value and runtime for denoising the image "doge" with manually added noise. The results show that Doge-ADMM outperforms all other methods in the settings of $k = 0$ and $k = 1$.

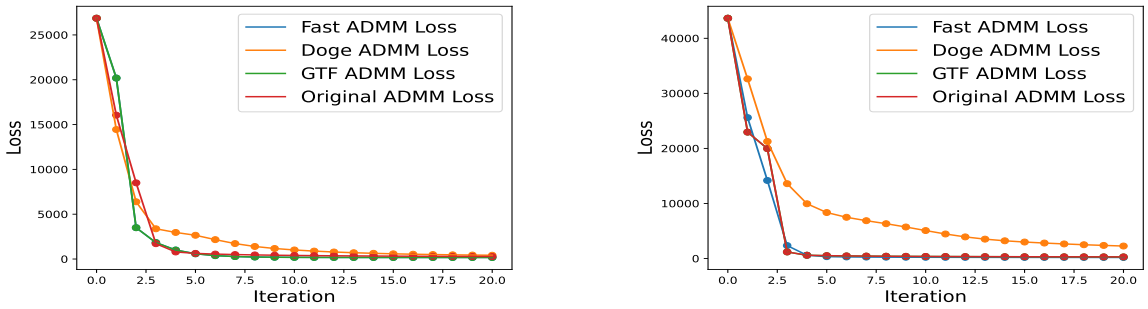
Figure 6 shows the relationship between the loss value and iterations for denoising. It turns out that with the same iteration number, the loss value does not seem to vary much. Therefore, with the same iterations, our Doge-ADMM method can achieve remarkable speed-up.



(a) k=0 (b) k=1

Figure 5: Loss-Time curve

Next, we conduct a Monte Carlo experiment to demonstrate the significant speed improvement of our algorithm. For each algorithm, we add random Gaussian noise with a variance of 20^2 to the Doge image, repeat the experiment 50 times, and record the average time. In each algorithm, we used the following parameter settings: $k = 2, B = 1, \rho = 1,$ and $\lambda = 10$. Table 3.1 reveals that Doge-ADMM achieves at least 30 times more acceleration than all other methods. This also shows that the acceleration of Doge-ADMM is mainly achieved by the extremely low computational cost



(a) k=0 (b) k=1

Figure 6: Loss-Iteration curve

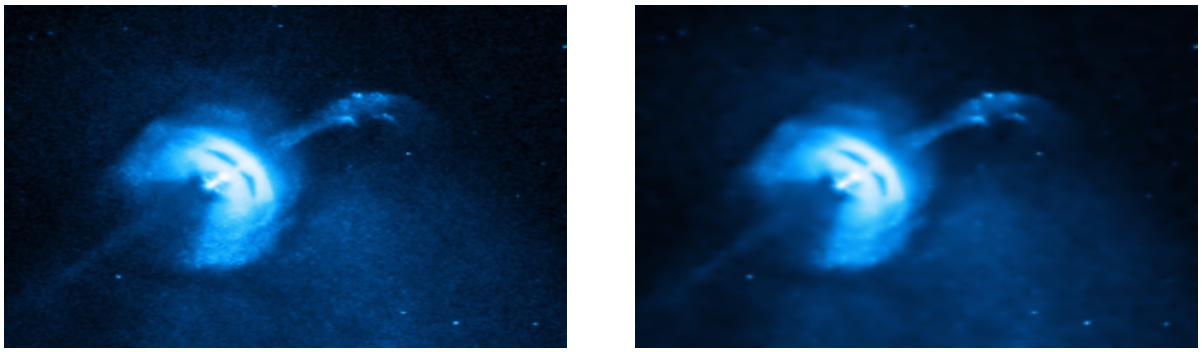
in one iteration. This is mainly because when $B = k + 1$, sub-problem (10) will no longer require inversion operations, which can significantly reduce computational overhead.

Table 1: Computational time (in seconds) and iteration time for image denoising

Method	Original ADMM	GTF ADMM	Fast ADMM	Doge-ADMM
Time	22.046($\pm$1.051)	29.894($\pm$1.683)	31.432($\pm$ 0.889)	0.493 ($\pm$ 0.011)
Loss($\times 10^5$)	8.255($\pm$0.138)	8.246($\pm$0.141)	8.176($\pm$ 0.145)	8.214($\pm$ 0.162)
Iteration	16.36($\pm$0.59)	23.16($\pm$0.36)	8.06($\pm$ 0.24)	47.25($\pm$0.43)
Iteration/Time	0.7420($\pm$0.044)	0.7747($\pm$0.045)	0.2564($\pm$0.011)	95.841($\pm$2.309)

3.2 Image Denoising

In this section, we apply our proposed method to the denoising of images in two critical domains: astronomy and medicine. Specifically, we focus on the denoising of astronomical images, where noise can significantly impact the accuracy of star positioning and identification. Figure 7 demonstrates the effectiveness of our method in denoising the neutron star image. After applying the filtering process, the resulting image successfully highlights the critical information, particularly the positions of the stars, while effectively mitigating the noise present in the background.



(a) Original Image (b) Filtered Image

Figure 7: Neutron star image denoising

Functional Magnetic Resonance Imaging (fMRI) is a neuroimaging technique that measures brain activity by detecting changes in blood flow. It captures dynamic brain activity by detecting the hemodynamic response to neural activation, providing high spatial and temporal resolution images. However, fMRI data are often corrupted by noise, including physiological and scanner-related artifacts, which can hinder accurate analysis and interpretation. By leveraging the power of the Doge-ADMM method, we effectively reduce noise while preserving the underlying brain activity patterns. Figure 8 shows the denoised fMRI data, where the algorithm has successfully filtered out the noise, enhancing the visibility of brain activation areas. The filtered results allow for clearer and more reliable analysis of brain function and connectivity, facilitating better interpretation of neural responses during specific tasks or stimuli.

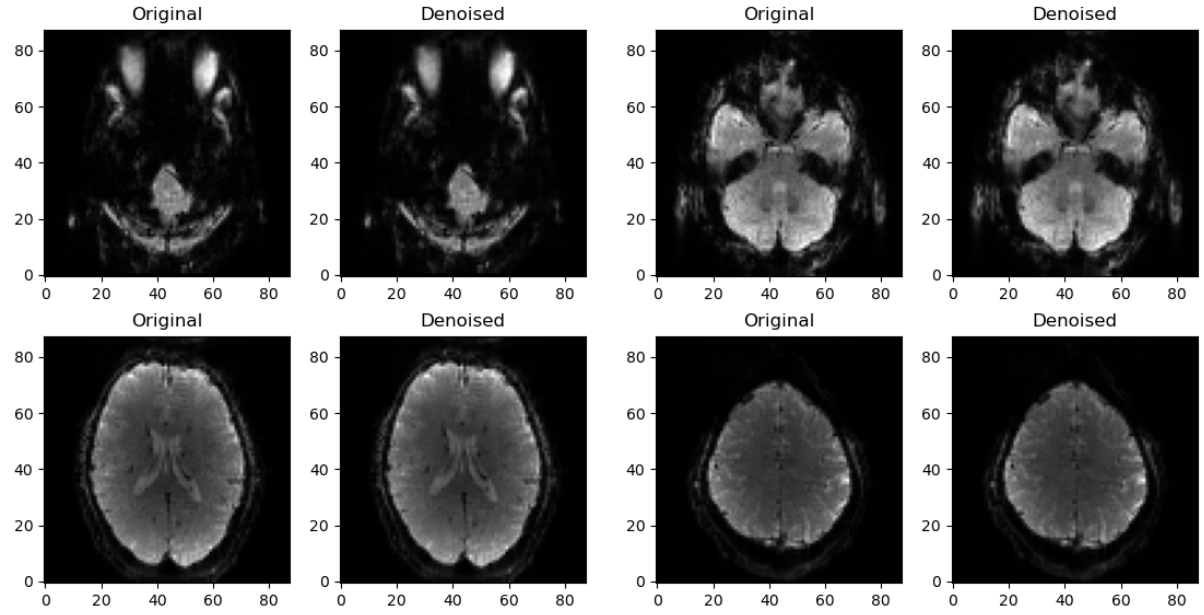


Figure 8: Denoising for fMRI

4 Conclusion

In this paper, we introduced the Doge-ADMM, an efficient method for trend filtering on graphs that combines computational efficiency with high accuracy. Our method leverages the structure of the ADMM to provide scalable method for large-scale graph data, enabling effective trend detection and smoothing with minimal computational overhead. Experimental results demonstrate its superiority over existing methods in terms of both performance and adaptability to different graph structures.

Future work will focus on extending Doge-ADMM to accommodate a broader range of graph structures and various penalties. These enhancements aim to make the approach applicable to a wider variety of real-world scenarios, further advancing its utility in graph-based data analysis.

References

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122.

Faisal, A., Parveen, S., Badsha, S., and Sarwar, H. (2012). An improved image denoising and segmentation approach for detecting tumor from 2-d mri brain images. In *2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*, pages 452–457.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360.

Fan, W., Liu, X., Jin, W., Zhao, X., Tang, J., and Li, Q. (2022). Graph trend filtering networks for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pages 112–121.

Kim, S.-J., Koh, K., Boyd, S., and Gorinevsky, D. (2009). \ell_1 trend filtering. *SIAM review*, 51(2):339–360.

Pang, J. and Cheung, G. (2017). Graph laplacian regularization for image denoising: Analysis in the continuous domain. *IEEE Transactions on Image Processing*, 26(4):1770–1785.

Ramdas, A. and Tibshirani, R. J. (2016). Fast and flexible admm algorithms for trend filtering. *Journal of Computational and Graphical Statistics*, 25(3):839–858.

Steidl, G., Didas, S., and Neumann, J. (2006). Splines in higher order tv regularization. *International journal of computer vision*, 70:241–255.

Varma, R., Lee, H., Chi, Y., and Kovacevic, J. (2019). Improving graph trend filtering with non-convex penalties. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5391–5395.

Wang, Y.-X., Sharpnack, J., Smola, A. J., and Tibshirani, R. J. (2016). Trend filtering on graphs. *Journal of Machine Learning Research*, 17(105):1–41.

Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty.

A Experimental Details

Original images

Figure 9: Doge

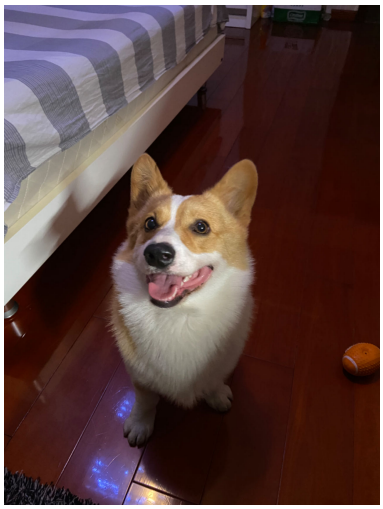


Figure 10: Dog

Codes

The code is available at github.com/byn1002/Trend-Filtering-on-Graphs.