Simultaneous Graph Learning and Blind Separation of Graph Signal Sources

Aref Einizade, Sepideh Hajipour Sardouie and Mohammad B. Shamsollahi *Senior Member, IEEE*

***Abstract*—When our sources are graph signals, a more efficient algorithm for Blind Source Separation (BSS) can be provided by using structural graph information along with statistical independence and/or non-Gaussianity. To the best of our knowledge, the GraphJADE and GraDe algorithms are the only BSS methods addressing this issue in the case of known underlying graphs. However, in many real-world applications, these graphs are not necessarily a priori known. In this paper, we propose a method called GraphJADE-GL (GraphJADE with Graph Learning) that jointly separates the graph signal sources and learns the graphs related to them accurately, in an alternating style.**

***Index Terms*—Graph Signal Processing (GSP); Graph Learning (GL); Blind Source Separation (BSS); Numerical Optimization; Erdös-Rényi graph; Machine Learning.**

I. INTRODUCTION

IN many data processing applications, signals are not necessarily on a regular grid. In such cases, considering a flexible grid for signals can help better understand and describe the data structure. These necessities form the foundations of Graph Signal Processing (GSP) [1]. In this field, various tools have been developed for processing signals live on known or defined graphs in a way that express the statistical and structural characteristics of the data well [1, 2]. However, in many applications, there are no such graphs that can accurately describe the graph structure of signals [3, 4]. Therefore, it is very important and helpful to learn such graphs using reasonable assumptions (smoothness [4, 5], sparsity [5], etc.) that are consistent with the statistical nature of the data. Another powerful and widely used tool for understanding the statistical nature of data is Blind Source Separation (BSS). The main purpose of the BSS methods is to retrieve the latent components from a mixture (usually linear) provided to us as observation. To this end, various assumptions have been made about the latent components and the mixing system. One of the most famous and widely used branches of these methods is Independent Component Analysis (ICA). To name a few well-known ICA methods, we can mention FastICA [6, 7], JADE [8], and also recent approaches such as UMME [9], Picard [10], NeoICA [11], and RobustICA [12]. In this family of methods, a criterion for examining independence is considered, and based on it and also the statistics of the input data, an attempt is made to separate and retrieve independent sources. One of the most useful criteria is non-Gaussianity. Therefore, most of these methods have difficulty to separate Gaussian components. In these cases, methods that measure temporal correlation can be used (e.g. SOBI [13], etc.). In the case of graph signal sources, these temporal correlations can be expressed as graph dependencies. Despite the great efforts that have been made in the area of BSS, the development of methods dealing with graph signals has not yet been studied well. To the best of our knowledge, only two methods have been proposed to address this issue: GraphJADE [14, 15] and Graph Decorrelation (GraDe) [16, 17]. These methods use the structure of underlying graphs as a prior knowledge in the separation process and lead to the BSS of graph signals.

In addition to the significant advantages of GraphJADE and GraDe methods, there is a major drawback in using these methods in many graph-based data. In these methods, the graphs that make up the structure of the graph signals of the hidden components are assumed to be known. This assumption is not practical in many real-world applications [3, 4]. In this paper, our main goal is to provide a method for separating the graph signal sources that simultaneously takes the advantages of GraphJADE and GraDe, and also learns the data structure graphs with high accuracy. In our proposed method, called GraphJADE with Graph Learning (GraphJADE-GL), the eigenvectors of the adjacency matrix of graphs are shared with those of the sample covariance matrix of the latent components. So, our optimization problem reduces to learning the eigenvalues of the adjacency matrices of the graphs with the minimum L1-norm constraint. Finally, the proposed method is compared with classic/recent ICA methods denoted as non-graph ICAs (namely JADE, Picard, NeoICA and RobustICA), a GraDe-based method (GraDe-GL) and also with Modified GraphJADE (as described in Sec. IV.). Based on the final results, the efficiency of the proposed method is shown in both the BSS and GL steps.

Notation: We use boldface lowercase letters for vectors, boldface capital letters for matrices, and capital calligraphic letters for sets. The notations $\mathbb{E}\{\}$, $(.)^{\mathrm{T}}$, $pinv(.)$, $\langle .|. \rangle$ and $\|.\|_p$ stand for mathematical expectation, transpose, Moore-Penrose pseudo-inverse of a matrix, inner product of two vectors, and the p-norm of a matrix or vector, respectively.

The rest of the paper is organized as follows. In Sec. II, brief background materials in GSP and BSS are provided. In Sec III, the GraphJADE and GraDe methods are introduced briefly. Sec. IV introduces the proposed method in detail. In Sec. V, the numerical results are analyzed. Finally, Sec. VI concludes the paper. Sec. VII provides the Appendix. Also, the theoretical and experimental convergence analysis are provided in Sec. VIII (i.e., the supplementary material).

A. Einizade, S. Hajipour Sardouie and M. B. Shamsollahi are with the Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran, (email: aref.einizade@yahoo.com, hajipour@sharif.edu, mbshams@sharif.edu)

II. GSP AND BSS BACKGROUND

In this section, some brief background materials in GSP and BSS are provided as follows:

GSP Background. In an undirected and weighted graph $\mathcal{G}=\{\mathcal{V},\mathcal{E},\mathbf{W}\}$, $\mathcal{V}$ stands for the set of N vertices, $\mathcal{E}$ denotes the set of edges and $\mathbf{W}\in\mathbb{R}_{+}^{N\times N}$ is the symmetric adjacency matrix, which contains the non-negative weights of connection between vertices. The diagonal elements of $\mathbf{W}$ are zero because we have no self-loops. The set $\mathcal{W}$ is the set of valid adjacency matrices of the weighted and undirected graphs, which is defined as:

$$\mathcal{W}=\{\mathbf{W}\in\mathbb{R}^{N\times N}:\mathrm{W}_{ij}=\mathrm{W}_{ji}\geq 0,\mathrm{W}_{ii}=0,\forall i,j=1,..,N\} \tag{1}$$

A graph signal $\mathbf{x}:\mathcal{V}\rightarrow\mathbb{R}^{N\times 1}$ is a mapping function that assigns real values of the signal $\mathbf{x}\in\mathbb{R}^{N\times 1}$ to the vertices of $\mathcal{V}$.

BSS Background. Since our proposed method, like a large group of BSS algorithms, needs a pre-whitening step, without loss of generality, we assume that $\widetilde{\mathbf{X}}\in\mathbb{R}^{P\times T}$ denotes the pre-whitened version of the observed signals $\mathbf{X}\in\mathbb{R}^{P\times T}$, which is a linear mixture of the latent mutually independent components $\mathbf{Z}=(\mathbf{z}_1,\mathbf{z}_2,\dots,\mathbf{z}_P)^{\mathrm{T}}\in\mathbb{R}^{P\times T}$, and $\mathbf{U}\in\mathbb{R}^{P\times P}$ is an orthogonal matrix, where $\mathbf{Z}\approx\mathbf{U}\widetilde{\mathbf{X}}$. We also assume that components of $\mathbf{Z}$ have zero mean and unit variance. We aim to estimate $\mathbf{U}$ based on some statistical assumptions (such as non-Gaussianity of the estimated sources) and some graph-based information about the latent graph signal components. The final estimated unmixing matrix $\hat{\mathbf{\Gamma}}\in\mathbb{R}^{P\times P}$ can be inferenced by $\hat{\mathbf{\Gamma}}=\mathbf{U}\hat{\mathbf{S}}_0^{-\frac{1}{2}}$, where $\hat{\mathbf{S}}_0\in\mathbb{R}^{P\times P}$ is the sample covariance matrix of $\mathbf{X}$ [14].

III. GRAPHJADE AND GRADE

In the GraphJADE objective function, graph decorrelation information is considered along with the JADE criterion as:

$$\begin{aligned}f_{GraphJADE}(\mathbf{U})=b\sum_{d=1}^{D}\sum_{p=1}^{P}\left\|diag\left(\mathbf{U}\tilde{\mathbf{S}}(\widetilde{\mathbf{X}},\mathbf{W}_p^d)\mathbf{U}^{\mathrm{T}}\right)\right\|^2\\+(1-b)\sum_{j=1}^{P}\sum_{l=1}^{P}\left\|diag\left(\mathbf{U}\hat{\mathbf{C}}^{\mathrm{j},\ell}\mathbf{U}^{\mathrm{T}}\right)\right\|^2\end{aligned} \tag{2}$$

where $b\in[0,1]$ is a balancing parameter. The second term is the JADE objective function, in which the $\{\hat{\mathbf{C}}^{\mathrm{j},\ell}\}_{j,\ell=1}^{P,P}$ values are the fourth order cumulants calculated as described in [14]. The first term is related to the graph dependencies of the sources. In this term, D is the total number of time lags (i.e., matrix powers of $\mathbf{W}_p$). Also, the $\tilde{\mathbf{S}}(\widetilde{\mathbf{X}},\mathbf{W}_p^d)\in\mathbb{R}^{P\times P}$, which we call $\tilde{\mathbf{S}}_p^d$, is the graph autocorrelation matrix of the $\widetilde{\mathbf{X}}$ and the adjacency matrix of the pth source (i.e., $\mathbf{W}_p\in\mathbb{R}^{T\times T}$) defined as:

$$\tilde{\mathbf{S}}_p^d=\tilde{\mathbf{S}}(\widetilde{\mathbf{X}},\mathbf{W}_p^d)=P\frac{\widetilde{\mathbf{X}}\,\mathbf{W}_p^d\,\widetilde{\mathbf{X}}^{\mathrm{T}}}{\|\mathbf{W}_p^d\,\widetilde{\mathbf{X}}^{\mathrm{T}}\|} \tag{3}$$

As can be seen in (2), each source can have its own underlying graph (i.e., adjacency matrix). On the other hand, in the GraDe objective function, only the graph dependency information is used for the separation process, and all sources have *one* identical underlying graph (i.e., $\mathbf{W}\in\mathbb{R}^{T\times T}$). Precisely, the GraDe objective function can be stated as:

$$f_{GraDe}(\mathbf{U})=\sum_{d=1}^{D}\left\|diag\left(\mathbf{U}\tilde{\mathbf{S}}(\widetilde{\mathbf{X}},\mathbf{W}^d)\mathbf{U}^{\mathrm{T}}\right)\right\|^2 \tag{4}$$

where $\tilde{\mathbf{S}}(\widetilde{\mathbf{X}},\mathbf{W}^d)\in\mathbb{R}^{P\times P}$, which we call $\tilde{\mathbf{S}}^d$, is the graph autocorrelation matrix having a slightly different definition to $\tilde{\mathbf{S}}_p^d$ and can be written as:

$$\tilde{\mathbf{S}}^d=\tilde{\mathbf{S}}(\widetilde{\mathbf{X}},\mathbf{W}^d)=\frac{1}{T-d}\left(\widetilde{\mathbf{X}}\,\mathbf{W}^d\,\widetilde{\mathbf{X}}^{\mathrm{T}}\right) \tag{5}$$

The objective functions of GraphJADE (2) and GraDe (4) are maximized with orthogonality constraint on the matrix $\mathbf{U}$ as:

$$\mathbf{U}=\underset{\mathbf{U}^{\mathrm{T}}\mathbf{U}=\mathbf{I}_P}{\operatorname{argmax}}\,f_{GraphJADE/GraDe}(\mathbf{U}) \tag{6}$$

where $\mathbf{I}_P\in\mathbb{R}^{P\times P}$ is the identity matrix. Note that the matrix $\mathbf{U}$ can be obtained using Joint Diagonalization (JD) of the matrices $\{\tilde{\mathbf{S}}_p^d\}_{p=1,d=1}^{P,D}$/$\{\tilde{\mathbf{S}}^d\}_{d=1}^{D}$ and $\{\hat{\mathbf{C}}^{\mathrm{j},\ell}\}_{j,\ell=1}^{P,P}$ for the GraphJADE/GraDe methods. The method proposed in [18] is used to implement JD based on the Givens rotations.

IV. GRAPHJADE-GL

Our proposed method consists of two iterative steps, namely the BSS and GL steps. We co-optimize these two steps in an alternating optimization procedure with the details as:

BSS step: In our work, in the BSS step, we modify the objective function of the GraphJADE (2) [14, 15]. The objective function of the proposed GraphJADE-GL method, in the BSS step, is as follows:

$$\begin{aligned}f_{GraphJADE-GL}^{BSS\ Step}(\mathbf{U})=b\sum_{d=1}^{D}\sum_{p=1}^{P}\left\|diag\left(\mathbf{U}\left[\frac{1}{WinNum}\sum_{r=1}^{WinNum}\tilde{\mathbf{S}}_{p,r}^d\right]\mathbf{U}^{\mathrm{T}}\right)\right\|^2\\+(1-b)\sum_{j=1}^{P}\sum_{l=1}^{P}\left\|diag\left(\mathbf{U}\hat{\mathbf{C}}^{\mathrm{j},\ell}\mathbf{U}^{\mathrm{T}}\right)\right\|^2\end{aligned} \tag{7}$$

where the second term is similar to (2) and $\tilde{\mathbf{S}}_{p,r}^d$ is the graph autocorrelation matrix corresponding to the $\widetilde{\mathbf{X}}_r\in\mathbb{R}^{P\times N}$ (i.e., the rth window of $\widetilde{\mathbf{X}}\in\mathbb{R}^{P\times T}$, where $T=WinNum\times N$ and $WinNum$ denotes the number of windows) and the adjacency matrix of the pth source (i.e., $\mathbf{W}_p\in\mathbb{R}^{N\times N}$) calculated as:

$$\tilde{\mathbf{S}}_{p,r}^d=\tilde{\mathbf{S}}(\widetilde{\mathbf{X}}_r,\mathbf{W}_p^d)=P\frac{\widetilde{\mathbf{X}}_r\mathbf{W}_p^d\widetilde{\mathbf{X}}_r^{\mathrm{T}}}{\|\mathbf{W}_p^d\widetilde{\mathbf{X}}_r^{\mathrm{T}}\|} \tag{8}$$

If the graphs $\{\mathbf{W}_p\}_{p=1}^{P}$ are a priori known, we call (7) the Modified GraphJADE (M-GraphJADE) objective function.

As can be seen in (7), in the GraphJADE-GL, unlike the GraphJADE, the *average* graph dependencies are exploited. Also, for considering only the graph dependencies, similar to the GraDe, we set $b=1$ in (7) and call BSS step of the GraDe-GL method resulting to the following objective function:

$$f_{GraDe-GL}^{BSS\ Step}(\mathbf{U})=\sum_{d=1}^{D}\sum_{p=1}^{P}\left\|diag\left(\mathbf{U}\left[\frac{1}{WinNum}\sum_{r=1}^{WinNum}\tilde{\mathbf{S}}_{p,r}^d\right]\mathbf{U}^{\mathrm{T}}\right)\right\|^2 \tag{9}$$

where $\tilde{\mathbf{S}}_{p,r}^d$ is calculated as:

$$\tilde{\mathbf{S}}_{p,r}^d=\tilde{\mathbf{S}}(\widetilde{\mathbf{X}}_r,\mathbf{W}_p^d)=\frac{\widetilde{\mathbf{X}}_r\mathbf{W}_p^d\widetilde{\mathbf{X}}_r^{\mathrm{T}}}{N-d}, \tag{10}$$

Remark 1: In GraDe-GL, each latent component can have its own graph (i.e., $\{\mathbf{W}_p\}_{p=1}^{P}\in\mathbb{R}^{N\times N}$), but in GraDe, *one* graph (i.e., $\mathbf{W}\in\mathbb{R}^{T\times T}$) is shared among all graph signal sources.

Remark 2: We have replaced $\tilde{\mathbf{S}}_p^d$ in the GraphJADE objective (2) with the average of $\tilde{\mathbf{S}}_{p,r}^d;\ r=1,\dots,WinNum$ over $WinNum$ windows of the pth component. Therefore, unlike GraphJADE and GraDe, $\frac{1}{WinNum}\sum_{r=1}^{WinNum}\tilde{\mathbf{S}}_{p,r}^d$ captions *average* graph dependencies over windowed parts of the pth source.

In the BSS step, similar to the GraphJADE, assuming the graphs (i.e., $\{\mathbf{W}_p\}_{p=1}^{P}$ in GraphJADE-GL/GraDe-GL) are estimated in the previous GL step, estimation of $\mathbf{U}$ can be done using JD [18] of matrices $\{\sum_{r=1}^{WinNum}\tilde{\mathbf{S}}_{p,r}^d\}_{p=1,d=1}^{P,D}$ and $\{\hat{\mathbf{C}}^{\mathrm{j},\ell}\}_{j,\ell=1}^{P,P}$.

GL step: To learn the graphs describing the structures of the latent sources, we assume windowed parts of the latent sources are diffused on these graphs using independent identically distributed (i.i.d.) graph signals with certain distributions (a piece-wise stationary assumption). We consider two statistical processes for generation of the windowed parts of the sources [19]: Graph Moving Average (GMA) and Graph Auto-Regressive (GAR). We use the first order of these graph processes, that is, GMA(1): $\mathbf{z}_{r,p} = \theta\mathbf{W}_p\mathbf{y}_{r,p} + \mathbf{y}_{r,p}$ and GAR(1): $\mathbf{z}_{r,p} = \theta\mathbf{W}_p\mathbf{z}_{r,p} + \mathbf{y}_{r,p}$, where $\mathbf{y}_{r,p} \in \mathbb{R}^{N\times 1}$ is an i.i.d. graph signal having a certain distribution for generating the rth window of the pth latent source (i.e., $\mathbf{z}_p = \left(\mathbf{z}_{1,p}^{\mathrm{T}}, \dots, \mathbf{z}_{WinNum,p}^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{(WinNum\times N)\times 1}$). Also, θ and $\mathbf{W}_\mathrm{p}$ are a coefficient and the adjacency matrix of the underlying graph corresponding to the generation of $\mathbf{z}_p$ [19], respectively. These explanations are summarized as:

$$\mathrm{GMA(1)}: \mathbf{z}_{r,p} = \overbrace{\left(\mathbf{I}_P + \theta\mathbf{W}_p\right)}^{\mathbf{H}_{GMA}}\mathbf{y}_{r,p} = \mathbf{H}_{GMA}\mathbf{y}_{r,p} \tag{11}$$

$$\mathrm{GAR(1)}: \mathbf{z}_{r,p} = \overbrace{\left(\mathbf{I}_P - \theta\mathbf{W}_p\right)^{-1}}^{\mathbf{H}_{GAR}}\mathbf{y}_{r,p} = \mathbf{H}_{GAR}\mathbf{y}_{r,p} \tag{12}$$

Equations (11) and (12) are the specific forms of filtering the graph signal $\mathbf{y}_{r,p}$ using graph filters $\mathbf{H}_{GMA}$ and $\mathbf{H}_{GAR}$ [19], respectively. Using these equations and stationarity of GMA/GAR processes, it can be shown [20] that the sample covariance matrix $\mathbf{C}_p = \mathbb{E}\{\mathbf{z}_{r,p}\mathbf{z}_{r,p}^{\mathrm{T}}\} \in \mathbb{R}^{N\times N}$, the graph filter $\mathbf{H}_{GMA}$ ($\mathbf{H}_{GAR}$), and eventually the adjacency matrix $\mathbf{W}_p$ can have identical eigenvectors. Therefore, in order to recover these graphs, we need only learn their eigenvalues [19, 20]. To this end, we use a well-known cost function, introduced in [20], for $p = 1, \dots P$, separately, as:

$$\min_{\mathbf{W}_p,\boldsymbol{\lambda}_p} \left\|\mathbf{W}_p\right\|_1 \quad subject\ to: \mathbf{W}_p = \mathbf{V}_p diag\left(\boldsymbol{\lambda}_p\right)\mathbf{V}_p^{\mathrm{T}}, \quad \mathbf{W}_p \in \mathcal{W} \tag{13}$$

where $\mathbf{V}_p \in \mathbb{R}^{N\times N}$ is the matrix of the eigenvectors of $\mathbf{C}_p$, the eigenvalues of $\mathbf{W}_p$ are stored in $\boldsymbol{\lambda}_p \in \mathbb{R}^{N\times 1}$ and $diag\left(\boldsymbol{\lambda}_p\right)$ is a diagonal matrix of the eigenvalues $\boldsymbol{\lambda}_\mathrm{p}$. Note that for preventing the trivial all-zero solution, additional constraints in $\mathcal{W}$ can be exploited, such as $\sum_{j=1}^{N} \mathrm{W}_{1j} = 1$ [20], $\|\mathbf{W}\|_F = 1$, etc. However, based on our experimental results, exploiting these kinds of additional constraints lowers the convergence speed severely and, therefore, we didn't use any of them, and no trivial solutions were observed regarding our chosen free parameters. We can solve (13) more efficiently and just for the strictly vectorized upper triangular part (i.e., without considering zero diagonal elements) of the $\{\mathbf{W}_p\}_{p=1}^{P}$. We call these vectors $\{\mathbf{w}_p\}_{p=1}^{P}$, and define the set $\mathcal{W}_v$ for them as:

$$\mathcal{W}_v = \left\{\mathbf{w} \in \mathbb{R}^{\frac{N(N-1)}{2}\times 1} :\ \mathrm{w}_i \geq 0, \forall i = 1, \dots, \frac{N(N-1)}{2}\right\}. \tag{14}$$

Also, with defining the ith element of $\boldsymbol{\lambda}_p$ and the ith column of the $\mathbf{V}_p$ as λ_p^i and $\mathbf{V}_p^i$, respectively, the equality constraint in (13) can be rewritten using the vectorize operator (i.e., $vec(.)$) and the relation $\mathbf{W}_p = \sum_{i=1}^{N} \lambda_p^i \mathbf{V}_p^i \left(\mathbf{V}_p^i\right)^{\mathrm{T}}$ as:

$$vec\left(\mathbf{W}_p\right) = \sum_{i=1}^{N} \lambda_p^i vec\left(\mathbf{V}_p^i\left(\mathbf{V}_p^i\right)^{\mathrm{T}}\right) \tag{15}$$

We define $vech(.)$ and $vechn(.)$ as the half vectorize and half non-diagonal vectorize operators. Also, we define $\mathbf{M}_d$ as the duplication matrix [21], and $\mathbf{M}_n$ as a matrix such that $vech(\mathbf{W}) = \mathbf{M}_n vechn(\mathbf{W})$ for a zero diagonal matrix $\mathbf{W}$ (such as an adjacency matrix), respectively. Now using the equations $vec\left(\mathbf{W}_p\right) = \mathbf{M}_d vech(\mathbf{W}_p)$ [21], $vech\left(\mathbf{W}_p\right) = \mathbf{M}_n vechn\left(\mathbf{W}_p\right)$ and $vechn\left(\mathbf{W}_p\right) = \mathbf{w}_p$, we can rewrite (15) as:

$$vec\left(\mathbf{W}_p\right) = \mathbf{M}_d\mathbf{M}_n\mathbf{w}_p = \widetilde{\mathbf{V}}_p\boldsymbol{\lambda}_p;\ \widetilde{\mathbf{V}}_p = \left[vec\left(\mathbf{V}_p^1\left(\mathbf{V}_p^1\right)^{\mathrm{T}}\right) | \dots | vec\left(\mathbf{V}_p^N\left(\mathbf{V}_p^N\right)^{\mathrm{T}}\right)\right] \tag{16}$$

So, using $pinv$ operator, we can rewrite $\mathbf{W}_p = \mathbf{V}_p diag\left(\boldsymbol{\lambda}_p\right)\mathbf{V}_p^{\mathrm{T}}$ as:

$$\mathbf{w}_p = \mathbf{A}_p\boldsymbol{\lambda}_p\ ;\ \mathbf{A}_p = pinv(\mathbf{M}_d\mathbf{M}_n)\widetilde{\mathbf{V}}_p \tag{17}$$

Therefore, (13) can be rewritten in a more efficient manner as:

$$\min_{\mathbf{w}_p,\boldsymbol{\lambda}_p} \left\|\mathbf{w}_p\right\|_1 \quad subject\ to: \mathbf{w}_p = \mathbf{A}_p\boldsymbol{\lambda}_p\ , \quad \mathbf{w}_p \in \mathcal{W}_v \tag{18}$$

To solve (18), we use Alternating Direction Method of Multipliers (ADMM) [22]. This problem can be split as:

$$\min_{\mathbf{w}_p,\boldsymbol{\lambda}_p} \left\|\mathbf{w}_p\right\|_1 \quad subject\ to: \mathbf{w}_p = \mathbf{A}_p\boldsymbol{\lambda}_p\ , \mathbf{w}_p = \mathbf{s}_p\ , \mathbf{s}_p \in \mathcal{W}_v\ ; \tag{19}$$

where $\mathbf{s}_p$ is an auxiliary variable. Then, the Augmented Lagrangian and iterative scheme of (19) can be written as:

$$\begin{aligned}\left(\mathbf{w}_p^{k+1}, \boldsymbol{\lambda}_p^{k+1}, \mathbf{s}_p^{k+1}\right) = \underset{\mathbf{w},\boldsymbol{\lambda},\mathbf{s}}{\mathrm{argmin}} \Bigg[&\|\mathbf{w}\|_1 + \frac{\rho^k}{2}\left(\left\|\mathbf{w} - \mathbf{A}_p\boldsymbol{\lambda}\right\|_2^2 + \|\mathbf{s} - \mathbf{w}\|_2^2\right) \\ &- \left\langle\boldsymbol{\alpha}_1^k | \mathbf{w} - \mathbf{A}_p\boldsymbol{\lambda}\right\rangle - \left\langle\boldsymbol{\alpha}_2^k | \mathbf{s} - \mathbf{w}\right\rangle\Bigg]\end{aligned} \tag{20}$$

$$\boldsymbol{\alpha}_1^{k+1} = \boldsymbol{\alpha}_1^k - \rho^k\left(\mathbf{w}_p^k - \mathbf{A}_p\boldsymbol{\lambda}_p^k\right) \tag{21}$$

$$\boldsymbol{\alpha}_2^{k+1} = \boldsymbol{\alpha}_2^k - \rho^k\left(\mathbf{s}_p^k - \mathbf{w}_p^k\right) \tag{22}$$

$$\rho^{k+1} = \rho^k \times cnt \tag{23}$$

where $\boldsymbol{\alpha}_1$and $\boldsymbol{\alpha}_2$ are the Lagrange multipliers (dual variables), k is the iteration index, and cnt is the increase constant. Note that to update $\mathbf{w}_p$, we use the definition of the proximity operators [23]. The update steps of the primal variables $\mathbf{w}_p, \boldsymbol{\lambda}_p$ and $\mathbf{s}_p$ are as follows (the details can be found in Appendix):

$$\mathbf{w}_p^{k+1} = prox_{\frac{\|\mathbf{w}\|_1}{2\rho^k}}\left(\frac{\rho^k\mathbf{A}_p\boldsymbol{\lambda}_p^k + \rho^k\mathbf{s}_p^k + \boldsymbol{\alpha}_1^k - \boldsymbol{\alpha}_2^k}{2\rho^k}\right) \tag{24}$$

$$\boldsymbol{\lambda}_p^{k+1} = pinv\left(\mathbf{A}_p\right)\left[\mathbf{w}_p^{k+1} - \frac{\boldsymbol{\alpha}_1^k}{\rho^k}\right] \tag{25}$$

$$\mathbf{s}_p^{k+1} = \prod_{\mathcal{W}_v}\left(\mathbf{w}_p^{k+1} + \frac{\boldsymbol{\alpha}_2^k}{\rho^k}\right) \tag{26}$$

where $\prod_{\mathcal{M}}(\mathbf{a})$ denotes the Euclidean projection of $\mathbf{a}$ onto the set $\mathcal{M}$. Our proposed method is summarized in Algorithm 1.

V. Numerical Results and Discussion

We separately generate $P = 4$ Independent Components (ICs) (in 100 random repetitions) using GMA(1) and GAR(1) models. The adjacency matrices are obtained from Erdös-Rényi (ER) graph random model with $p_{ER} = 0.3$ [24]. In order to better compare our method with M-GraphJADE (with known graphs), the basics of the generation of the ICs are inspired from [14]. However, there are some differences, such as the usage of GAR(1), etc. In all simulations, we have $WinNum = 1000$, $cnt = 1000$, $D = 1$, and $b = 0.4$. The performance criterion in separation task is the Minimum Distance (MD) index [25], which can be stated as:

$$D\left(\hat{\mathbf{\Gamma}}\right) = \frac{1}{\sqrt{P-1}} \inf_{\mathbf{C}\in\mathcal{C}} \left\|\mathbf{C}\hat{\mathbf{\Gamma}}\mathbf{\Omega} - \mathbf{I}_P\right\|_F \tag{27}$$

where $\hat{\mathbf{\Gamma}}$ and $\mathbf{\Omega}$ are estimated unmixing and true mixing matrices, respectively. The set $\mathcal{C}$ is the set of all possible permutation matrices. The minimum value of MD is zero and is obtained in the case that the latent components can be recovered by just rescaling and permutation of the estimated ones [14]. Also, we use Area Under Curve (AUC) [26] as a criterion for graph recovery. We compare our proposed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LSP.2021.3093872, IEEE Signal Processing Letters

method in separation task with GraDe-GL, non-graph ICA methods (ngICAs), as well as with the M-GraphJADE (7), in which the graphs are assumed to be known, as gold standard. We generate four models of data according to Table I. The results of the BSS task in terms of the average D^2 are shown in Figure 1, and the results of GL in terms of AUC are presented in Table II. Based on the BSS phase results (Fig. 1), in all models, the performance of the GraphJADE-GL is fairly near the M-GraphJADE because of the excellent performance in recovery of the underlying graphs, in almost all data models (Table II). All methods have acceptable performances in the first model due to satisfying their statistical assumptions about the data. In the second model, GraDe-GL has failed to separate the graph signal sources effectively due to the near-identical graph contributions. In the third model, because of the Gaussianity of the sources, ngICAs have poor performance in the separation task. In the fourth model, ngICAs could not achieve good results due to the Gaussianity of the ICs. GraDe-GL also has poor performance because the values of θ in two components are identical. But, the proposed GraphJADE-GL method has the best performance in the BSS task, very close to that of the M-GraphJADE. This great performance is confirmed by the excellent performance of the GraphJADE-GL in the graph recovery task (Table II). Also, on average, the GraphJADE-GL performs robustly in the separation task against changing the number of nodes N. In this way, due to relying only on graph dependencies, GraDe-GL seems vulnerable against the graph size changes and has a different pattern with GraphJADE-GL. For example, in GMA/GAR model 1 or 3, due to relatively lower AUC in GL step (see Table II), GraDe-GL, unlike GraphJADE-GL that uses also the JADE term, has higher MD and also lower robustness.

Table I: The details of four models for data generation (t_q: t-student with q degree of freedom)

	Model 1	Model 2	Model 3	Model 4
Distribution of the latent sources	t_{70}, Uniform, Exponential and Gaussian	t_{70}, Uniform, Exponential and Gaussian	All have Gaussian distribution	t_{15}, Gaussian, Uniform and Gaussian
θ-vector	[0.04, 0.08, 0.12,0.16]	[0.05, 0.06, 0.07,0.08]	[0.05, 0.05, 0.05,0.05]	[0.05, 0.05, 0.1,0.1]
Different **W**s for sources?	No	No	Yes	No
Challenging for which BSS methods?	-	GraDe-GL, because θs are almost identical.	ngICAs, because of gaussianity of sources	All except GraphJADE-GL, because of both Model 2 and 3 challenges.

Table II: Mean of AUC for the recovery of the different graph sources over all repetitions of each model for the proposed GraphJADE-GL method.

	GMA, Model #				GAR, Model #			
N	1	2	3	4	1	2	3	4
15	0.97	0.99	0.94	0.97	0.97	0.99	0.95	0.98
20	0.98	0.98	0.96	0.98	0.97	0.99	0.96	0.98
25	0.97	0.98	0.96	0.98	0.97	0.98	0.96	0.98

VI. Conclusion

In this paper, we proposed a Blind Source Separation (BSS) method, called GraphJADE-GL (GraphJADE with Graph Learning), which uses both the structural graph information and statistical independence of graph signal sources. Based on the numerical results, the proposed method jointly separates the sources and learns the underlying graphs related to them accurately, and its superiority over the state-of-the-art BSS methods and also GraDe-based methods can be admitted.

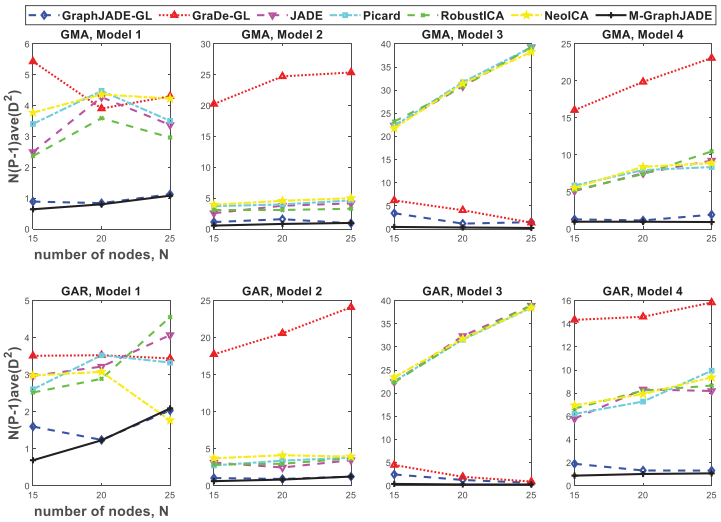

Figure 1: The average values of $N(P-1)\left(D(\mathrm{f})^2\right)$ over 100 random repetitions in GMA and GAR (top and bottom row, respectively) models 1-4.

Algorithm 1. **GraphJADE-GL** (Note that k and j are iteration index)

Input: Whitened data $\widetilde{\mathbf{X}} \in \mathbb{R}^{P\times(WinNum\times N)}$, N, $WinNum$, D, b, cnt

Output: $\mathbf{U}$, $\{\mathbf{W}_p\}_{p=1}^{P}$

Initialization: $j \leftarrow 0$ (iteration index), $\mathbf{W}_p^0 \leftarrow \mathbf{0}_{N\times N}; \forall p = 1, \dots, P$, $b \leftarrow 0$ for $j = 0$,

While convergence in $\mathbf{U}$

  maximize (7) to update $\mathbf{U}^j$, then obtain $\{\mathbf{C}_p\}_{p=1}^{P}$ from $\mathbf{Z}^j = \mathbf{U}^j\widetilde{\mathbf{X}}$

  For $p = 1{:}P$

    $k \leftarrow 0$, $\boldsymbol{\lambda}_p^0 \leftarrow$eigenvalues of $\mathbf{C}_p$, $\mathbf{w}_p^0 \leftarrow \mathbf{1}$, $\mathbf{s}_p^0 \leftarrow \mathbf{1}$, $\boldsymbol{\alpha}_1^0 \leftarrow \mathbf{w}_p^0 - \mathbf{A}_p\boldsymbol{\lambda}_p^0$, $\boldsymbol{\alpha}_2^0 \leftarrow \mathbf{s}_p^0 - \mathbf{w}_p^0$, $\rho^0 \leftarrow 1$

    While convergence

      Update $\mathbf{w}_p^{k+1}$, $\boldsymbol{\lambda}_p^{k+1}$, $\mathbf{s}_p^{k+1}$, $\boldsymbol{\alpha}_1^{k+1}$, $\boldsymbol{\alpha}_2^{k+1}$ and ρ^{k+1} using (24), (25), (26), (21), (22), and (23), respectively.

      $k \leftarrow k + 1$

    end **While**, return $\mathbf{W}_p^j$

  end **For**

  $j \leftarrow j + 1$

end **While**, return $\mathbf{U}^j$ and $\{\mathbf{W}_p^j\}_{p=1}^{P}$

VII. Appendix

The update processes of the $\mathbf{w}_p, \boldsymbol{\lambda}_p$ and $\mathbf{s}_p$ are as follows:

$$*)\ \mathbf{w}_p^{k+1} = \operatorname{argmin}_{\mathbf{w}}\left[\|\mathbf{w}\|_1 + \rho^k\|\mathbf{w}\|_2^2 - \rho^k\langle\mathbf{w}|\mathbf{A}_p\boldsymbol{\lambda}_p^k\rangle - \langle\mathbf{w}|\boldsymbol{\alpha}_1^k\rangle - \rho^k\langle\mathbf{w}|\mathbf{s}_p^k\rangle + \langle\mathbf{w}|\boldsymbol{\alpha}_2^k\rangle\right]$$

$$= \operatorname*{argmin}_{\mathbf{w}}\left[\|\mathbf{w}\|_1 + \rho^k\|\mathbf{w}\|_2^2 - \left\langle\mathbf{w}\middle|\rho^k\mathbf{A}_p\boldsymbol{\lambda}_p^k + \boldsymbol{\alpha}_1^k + \rho^k\mathbf{s}_p^k - \boldsymbol{\alpha}_2^k\right\rangle\right]$$

$$= \operatorname*{argmin}_{\mathbf{w}}\left[\|\mathbf{w}\|_1 + \rho^k\left\|\mathbf{w} - \frac{\rho^k\mathbf{A}_p\boldsymbol{\lambda}_p^k + \boldsymbol{\alpha}_1^k + \rho^k\mathbf{s}_p^k - \boldsymbol{\alpha}_2^k}{2\rho^k}\right\|_2^2\right]$$

$$\mathbf{w}_p^{k+1} = prox_{\frac{\|\mathbf{w}\|_1}{2\rho^k}}\left(\frac{\rho^k\mathbf{A}_p\boldsymbol{\lambda}_p^k + \rho^k\mathbf{s}_p^k + \boldsymbol{\alpha}_1^k - \boldsymbol{\alpha}_2^k}{2\rho^k}\right)$$

$$*)\ \boldsymbol{\lambda}_p^{k+1} = \operatorname{argmin}_{\boldsymbol{\lambda}}\left[\frac{\rho^k}{2}\|\mathbf{A}_p\boldsymbol{\lambda}\|_2^2 - \rho^k\langle\mathbf{A}_p\boldsymbol{\lambda}|\mathbf{w}_p^{k+1}\rangle + \langle\mathbf{A}_p\boldsymbol{\lambda}|\boldsymbol{\alpha}_1^k\rangle\right]$$

$$= \operatorname*{argmin}_{\boldsymbol{\lambda}}\left[\frac{\rho^k}{2}\left\|\mathbf{A}_p\boldsymbol{\lambda} - \frac{\rho^k\mathbf{w}_p^{k+1} - \boldsymbol{\alpha}_1^k}{\rho^k}\right\|_2^2\right] = pinv(\mathbf{A}_p)\left[\mathbf{w}_p^{k+1} - \frac{\boldsymbol{\alpha}_1^k}{\rho^k}\right]$$

$$*)\ \mathbf{s}_p^{k+1} = \operatorname{argmin}_{\mathbf{s}\in\mathcal{W}_v}\left[\frac{\rho^k}{2}\|\mathbf{s}\|_2^2 - \rho^k\langle\mathbf{s}|\mathbf{w}_p^{k+1}\rangle - \langle\mathbf{s}|\boldsymbol{\alpha}_2^k\rangle\right]$$

$$= \operatorname*{argmin}_{\mathbf{s}\in\mathcal{W}_v}\left[\frac{\rho^k}{2}\left\|\mathbf{s} - \frac{\rho^k\mathbf{w}_p^{k+1} + \boldsymbol{\alpha}_2^k}{\rho^k}\right\|_2^2\right] = \prod_{\mathcal{W}_v}\left(\mathbf{w}_p^{k+1} + \frac{\boldsymbol{\alpha}_2^k}{\rho^k}\right)$$

REFERENCES

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

[2] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE transactions on signal processing,* vol. 61, no. 7, pp. 1644-1656, 2013.

[3] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, "Learning graphs from data: A signal representation perspective," *IEEE Signal Processing Magazine,* vol. 36, no. 3, pp. 44-63, 2019.

[4] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning Laplacian matrix in smooth graph signal representations," *IEEE Transactions on Signal Processing,* vol. 64, no. 23, pp. 6160-6173, 2016.

[5] V. Kalofolias, "How to learn a graph from smooth signals," in *Artificial Intelligence and Statistics*, 2016, pp. 920-929.

[6] A. Hyvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural computation,* vol. 9, no. 7, pp. 1483-1492, 1997.

[7] A. Hyvarinen, "Fast and robust fixed-point algorithms for independent component analysis," *IEEE transactions on Neural Networks,* vol. 10, no. 3, pp. 626-634, 1999.

[8] J.-F. Cardoso and A. Souloumiac, "Blind beamforming for non-Gaussian signals," in *IEE proceedings F (radar and signal processing)*, 1993, vol. 140, no. 6: IET, pp. 362-370.

[9] L. Zhen, D. Peng, H. Zhang, Y. Sang, and L. Zhang, "Underdetermined mixing matrix estimation by exploiting sparsity of sources," *Measurement,* vol. 152, p. 107268, 2020.

[10] P. Ablin, J.-F. Cardoso, and A. Gramfort, "Faster independent component analysis by preconditioning with Hessian approximations," *IEEE Transactions on Signal Processing,* vol. 66, no. 15, pp. 4040-4049, 2018.

[11] P. Tillet, H. Kung, and D. Cox, "Infomax-ICA using hessian-free optimization," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017: IEEE, pp. 2537-2541.

[12] V. Zarzoso and P. Comon, "Robust independent component analysis by iterative maximization of the kurtosis contrast with algebraic optimal step size," *IEEE Transactions on neural networks,* vol. 21, no. 2, pp. 248-261, 2009.

[13] A. Belouchrani, K. Abed-Meraim, J.-F. Cardoso, and E. Moulines, "A blind source separation technique using second-order statistics," *IEEE Transactions on signal processing,* vol. 45, no. 2, pp. 434-444, 1997.

[14] J. Miettinen, S. A. Vorobyov, and E. Ollila, "Blind Source Separation of Graph Signals," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020: IEEE, pp. 5645-5649.

[15] J. Miettinen, E. Nitzan, S. A. Vorobyov, and E. Ollila, "Graph Signal Processing Meets Blind Source Separation," *IEEE Transactions on Signal Processing,* 2021.

[16] F. Blöchl, A. Kowarsch, and F. J. Theis, "Second-order source separation based on prior knowledge realized in a graph model," in *International Conference on Latent Variable Analysis and Signal Separation*, 2010: Springer, pp. 434-441.

[17] A. Kowarsch *et al.*, "Knowledge-based matrix factorization temporally resolves the cellular responses to IL-6 stimulation," *BMC bioinformatics,* vol. 11, no. 1, p. 585, 2010.

[18] D. B. Clarkson, "Remark AS R74: A least squares version of algorithm AS 211: The FG diagonalization algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics),* vol. 37, no. 2, pp. 317-321, 1988.

[19] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Stationary graph processes and spectral estimation," *IEEE Transactions on Signal Processing,* vol. 65, no. 22, pp. 5911-5926, 2017.

[20] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Transactions on Signal and Information Processing over Networks,* vol. 3, no. 3, pp. 467-483, 2017.

[21] K. M. Abadir and J. R. Magnus, *Matrix algebra*. Cambridge University Press, 2005.

[22] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[23] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-point algorithms for inverse problems in science and engineering*: Springer, 2011, pp. 185-212.

[24] M. E. Newman, "Random graphs as models of networks," *Handbook of graphs and networks,* vol. 1, pp. 35-68, 2003.

[25] P. Ilmonen, K. Nordhausen, H. Oja, and E. Ollila, "A new performance index for ICA: properties, computation and asymptotic analysis," in *International Conference on Latent Variable Analysis and Signal Separation*, 2010: Springer, pp. 229-236.

[26] J. Myerson, L. Green, and M. Warusawitharana, "Area under the curve as a measure of discounting," *Journal of the experimental analysis of behavior,* vol. 76, no. 2, pp. 235-243, 2001.

[27] D. G. Luenberger and Y. Ye, "Linear and Nonlinear Programming," ed: Springer, 2016.

[28] M. Fortin and R. Glowinski, *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*. Elsevier, 2000.

[29] R. Glowinski and P. Le Tallec, *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics*. SIAM, 1989.

[30] F. Lin, M. Fardad, and M. R. Jovanović, "Design of optimal sparse feedback gains via the alternating direction method of multipliers," *IEEE Transactions on Automatic Control,* vol. 58, no. 9, pp. 2426-2431, 2013.