Adapt-Align-Combine for Diffusion-Based Distributed Dictionary Learning

D. Ampeliotis, C. Mavrokefalidis, K. Berberidis
Department of Computer Engineering and Informatics, University of Patras
& C.T.I RU-2, 26500, Rio - Patras, Greece
E-mails: {ampeliot, maurokef, berberid}@ceid.upatras.gr

S. Theodoridis
University of Athens
Dept. of Informatics and Telecommunications
Athens 15784, Greece.
E-mail: stheodor@di.uoa.gr

***Abstract*—Diffusion-based distributed dictionary learning methods are studied in this work. We consider the classical mixed l_2-l_1 cost function, that employs an l_2 representation error term and an l_1 sparsity promoting regularizer. First, we observe that this cost function suffers from an inherent permutation ambiguity. This ambiguity may deteriorate significantly the performance of diffusion-based schemes, since the involved combination step may combine different atoms even when the same atoms exist at all dictionaries. Thus, we propose to align the dictionaries prior to the combination step. Furthermore, we define a new problem, that we call the node-specific distributed dictionary learning problem. The proposed Adapt-Align-Combine algorithm enjoys increased convergence rate as compared with a scheme that does not align the dictionaries prior to the combination. Simulation results support our findings.**

I. Introduction

Dictionary learning refers to the task of inferring an overdetermined linear model of a set of signal vectors [1]. More specifically, the dictionary is a set of representative signals, called atoms, and the signals of interest are represented as a linear combination of a small subset of those atoms. In this way, a sparse representation of the signals at hand is achieved, e.g. [2], [3]. Such models find applications as, for example, in medical imaging, audio and visual processing (e.g. denoising and data compression) and classification [4].

Distributed processing [5], on the other hand, has been in focus recently as it is able to increase the robustness and the scalability of the involved operations. For example, in a sensor network, robustness is increased by avoiding to use a central node, acting as a fusion center (FC), which is a single point of failure for the whole network. Additionally, distributed processing is more scalable as energy and communication resources are allocated only for local processing. On the contrary, in a centralized environment, the larger the network, the further the "edge" sensors are from the fusion center and, hence, this leads to increased energy and communication demands. Distributed processing has been initially developed for estimation/regression problems (e.g., in power-grids [6]) while, in recent years, this approach has been extended to other problems, including dictionary learning, e.g., [7], which is the problem that is considered in this paper, too.

The work was partially supported by the European HANDiCAMS project (Grant No. 323944) under the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission and in part by the University of Patras.

In recent literature, there have already been proposed a few notable approaches, that deal with the problem of distributed dictionary learning. In particular, in [8], the authors propose a diffusion-based adaptive dictionary learning approach employed by a network of sensors. Specifically, each sensor acquires an individual set of observations that originate from the same phenomenon (e.g., a scene recorded by a network of cameras). It is assumed that these observations can be sparsely represented using a common dictionary that sufficiently describes the phenomenon which is observed by the network. Based on the locally received observations and the dictionaries, that are exchanged among the neighbours, the sensors adapt their local copy of the dictionary using the adapt-then-combine strategy.

In [9] (and the more extended version [10]), the authors tackle the problem of distributed dictionary learning assuming that each sensor possesses only part of the whole dictionary as opposed to the method in [8]. The proposed distributed algorithm is also of the diffusion type; however, in this case, the sensors are not required to exchange with their neighbours their local copy of the dictionary. Instead, they exchange only a quantity that is related to a representation error and apply the diffusion operation on these quantities. Hence, the approach in [9], [10] minimizes the communication overhead of transmitting complete dictionaries and at the same time it takes into account privacy issues, as the sensors do not release their local copy.

The previous two approaches represent the two main directions that have appeared in the corresponding literature concerning distributed dictionary learning. Some other relevant works are the following. In [11], a consensus-based distributed algorithm, utilizing the alternate direction method of multipliers, is proposed for adapting a common dictionary. In [12] (and the more extended version [13]), the centralized K-SVD algorithm for dictionary learning is transformed into a distributed one using a consensus averaging. The authors also provide a study on the convergence behavior of their algorithm [14]. In [7], an online distributed dictionary algorithm is proposed by utilizing the recursive least squares (RLS) algorithm for learning a common dictionary. In another direction, the authors

978-0-9928-6265-7/16/$31.00 ©2016 IEEE

in [15] propose an approach for executing large-scale sparse coding and dictionary learning problems, following a parallel mode of operation, in distributed computing environments (e.g. either on multi-core or clusters of computers). This is accomplished by utilizing multi-threading and the Map-Reduce programming model. Finally, the work, in [16] (also in a different direction), proposes a dictionary learning framework in which the complete set of the data is not required to be used in each iteration. Instead, an online procedure is adopted that takes into account only single pieces or small sets of data for the desired dictionary learning. This framework is tailored also for very large data sets.

In this paper, distributed dictionary learning is studied for a network of sensors, such as cameras observing a common scene from a different point of view. Two important scenarios are considered. In the first one, the sensors aim at learning a common dictionary using a modified diffusion-based rule, with increased convergence speed. In the second scenario, we identify a setting in which a common dictionary may not be a good model for the data of all nodes. In this setting, we propose a new problem that we call the *node-specific* distributed dictionary learning problem. The proposed problem allows the dictionary of each sensor to consist of two parts, namely a common part among all sensors and a *node-specific* local part.

II. Distributed Dictionary Learning

A. Problem Formulation

Let us consider a network of N nodes, where each node $n \in \mathcal{N} = \{1, 2, \ldots, N\}$ has obtained some data that we represent by the matrix

$$\mathbf{Y}_n \in \mathbb{R}^{p \times q_n}, \quad n \in \mathcal{N} \ , \tag{1}$$

where p is the dimension of the data samples and q_n is the number of samples at node n. We assume that the nodes are interconnected as described by a graph $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{N}$ and

$$\mathcal{E} = \{(v_i, v_j) : \text{Node } i \in \mathcal{N} \text{ is connected with node } j \in \mathcal{N}\} \ .$$

We also consider the matrix

$$\mathbf{Y} = \left[\begin{array}{cccc} \mathbf{Y}_1 & \mathbf{Y}_2 & \cdots & \mathbf{Y}_N \end{array} \right] \in \mathbb{R}^{p \times q} \ , \tag{2}$$

where $q = q_1 + q_2 + \cdots + q_N$ is the total number of data samples in the network. We are interested in solving an optimization problem of the form

$$\{\mathbf{D}, \mathbf{A}\} = \arg \min_{\{\mathbf{D}, \mathbf{A}\}} \left(\frac{1}{2} \left\| \mathbf{Y} - \mathbf{D}\mathbf{A} \right\|_{\mathrm{F}}^2 + \lambda \left\| \mathbf{A} \right\|_1 \right) \ , \tag{3}$$

where $\mathbf{D} \in \mathbb{R}^{p \times K}$ is a redundant dictionary, $\mathbf{A} \in \mathbb{R}^{K \times q}$ is a matrix whose columns represent the sparse representations of the respective data vectors and K is a properly selected integer with $K >> p$. The parameter λ sets the relative weight between representation accuracy and sparsity of the matrix $\mathbf{A}$, i.e. between the Frobenious norm $\| \cdot \|_{\mathrm{F}}$ and the l_1 norm $\| \cdot \|_1$ terms, respectively. The focus here is on distributed algorithms for solving the optimization problem in (3). It is important to note that the cost function in the minimization problem (3), can be written in the form of a *sum of costs*,

$$\frac{\left\| \mathbf{Y} - \mathbf{D}\mathbf{A} \right\|_{\mathrm{F}}^2}{2} + \lambda \left\| \mathbf{A} \right\|_1 = \sum_{n=1}^{N} \left(\frac{1}{2} \left\| \mathbf{Y}_n - \mathbf{D}\mathbf{A}_n \right\|_{\mathrm{F}}^2 + \lambda \left\| \mathbf{A}_n \right\|_1 \right) , \tag{4}$$

where the matrices $\mathbf{A}_n$ denote the sparse representations of the respective data in $\mathbf{Y}_n$. As discussed in the previous paragraph, an approach for solving this optimization problem in a distributed fashion is to use a diffusion strategy [17]; that is, to iterate the following two steps:

1) Each node n uses local data $\mathbf{Y}_n$ and an updating rule $\mathcal{F}(\cdot)$ to update its local copy of the dictionary at time t - denoted as $\mathbf{D}_n^{(t)}$ - and the sparse representations matrix $\mathbf{A}_n^{(t)}$, as represented by the following equation

$$\left\{ \mathbf{D}_n^{(t+1/2)}, \mathbf{A}_n^{(t+1)} \right\} = \mathcal{F} \left(\mathbf{D}_n^{(t)}, \mathbf{A}_n^{(t)}, \mathbf{Y}_n \right) \ . \tag{5}$$

The result of this update is a new matrix of sparse representations $\mathbf{A}_n^{(t+1)}$ and an intermediate estimate of the dictionary that we have denoted as $\mathbf{D}_n^{(t+1/2)}$.

2) Neighbouring nodes exchange their local intermediate estimates of the dictionary, and each node computes a convex linear combination of the available dictionaries (i.e., those of its neighbours and its local intermediate estimate). Thus, the final estimate of the dictionary at node n and time $t+1$ is given by the expression

$$\mathbf{D}_n^{(t+1)} = \sum_{l \in \mathcal{N}_n} a_{l,n} \mathbf{D}_l^{(t+1/2)} \ , \tag{6}$$

where $\mathcal{N}_n$ denotes the set of neighbours of node n, including node n itself, and $a_{l,n}$ denote properly selected combination weights at node n, that should obey

$$\sum_{l \in \mathcal{N}_n} a_{l,n} = 1, \qquad \forall \quad n \in \mathcal{N} \ . \tag{7}$$

It is worth mentioning that in the above, the first step comprises the update of the local estimates of the dictionary and the respective sparse approximations and this step is trying to fit the local data of each node. On the other hand, the second step is trying to make all nodes agree on a common dictionary i.e., consent on a dictionary valid for all the data. As we will see in the following, these may be conflicting goals.

B. An Ambiguity of the Cost Function

We now observe that the cost function in (4) has the property that it is invariant with respect to a permutation of the atoms (columns) of matrix $\mathbf{D}$, accompanied by the inverse permutation of the rows of matrix $\mathbf{A}$. In particular, for any permutation matrix $\mathbf{P}$ with size $K \times K$, it is easy to see that

$$\frac{1}{2} \left\| \mathbf{Y} - \mathbf{D}\mathbf{A} \right\|_{\mathrm{F}}^2 + \lambda \left\| \mathbf{A} \right\|_1 = \frac{1}{2} \left\| \mathbf{Y} - \mathbf{D}\mathbf{P}^{\mathrm{T}}\mathbf{P}\mathbf{A} \right\|_{\mathrm{F}}^2 + \lambda \left\| \mathbf{P}\mathbf{A} \right\|_1 \ , \tag{8}$$

since $\mathbf{P}^T\mathbf{P} = \mathbf{I}$ and the permutation of the rows of matrix $\mathbf{A}$ does not change the sum of the absolute values of its elements. With this ambiguity in mind, we can see that the dictionary update steps in (5), performed at different neighbouring nodes, may result into similar dictionaries, but

with different orderings of their respective atoms. In particular, if the update rule $\mathcal{F}(\cdot)$ is able to compute a good fit of the local data of each node, and furthermore each node starts its update procedure from a different dictionary estimate, then the ordering of the atoms may be different. To overcome this, we propose a procedure to align the dictionaries of neighbouring nodes prior to the combination step of equation (6).

C. The Proposed Algorithm

Let us consider that node n receives the intermediate dictionary estimate $\mathbf{D}_l^{(t+1/2)}$ from a neighbouring node $l \in \mathcal{N}_n \setminus \{n\}$. Note that we use $t+1/4, t+1/2$ and $t+3/4$ to describe time instants that correspond to the computations of intermediate estimates between t and $t+1$. To cope with the ambiguity discussed in subsection II-B, prior to the combination of equation (6), node n must compute a permutation matrix $\mathbf{P}_{n,l}^{(t)}$ by solving the optimization problem

$$\mathbf{P}_{n,l}^{(t)} = \arg\min_{\mathbf{P}} \left\| \mathbf{D}_n^{(t+2/4)} - \mathbf{D}_l^{(t+2/4)} \mathbf{P} \right\|_{\mathrm{F}}^2 . \qquad (9)$$

Clearly, there are $K!$ permutation matrices with dimensions $K \times K$ and the above problem seems of combinatorial complexity. However, we show here that it can be solved as an instance of the so-called *linear assignment problem* [18]. In its classical setting, the linear assignment problem involves a number of agents and an equal number of tasks. Any agent can be assigned to perform any task, and a cost is involved for any agent-task pair. Thus, the costs can be given in a square matrix where rows stand for agents and columns for tasks. The scope is to assign each agent with exactly one task, so that the sum of costs is minimized. In our setting, we can represent agents as the atoms of one of the two dictionaries (say of node n) and tasks as the atoms of the other dictionary. Thus, we can define a matrix of costs, whose element on the i-th row and j-th column is given by the l_2-norm term

$$\left[\mathbf{C}_{n,l}^{(t)}\right]_{i,j} = \left\| \boldsymbol{d}_{n,i}^{(t+2/4)} - \boldsymbol{d}_{l,j}^{(t+2/4)} \right\|_2^2 , \qquad (10)$$

where $\boldsymbol{d}_{n,i}^{(t+2/4)}$ is the i-th atom (column) of the dictionary estimate at node n and $\boldsymbol{d}_{l,j}^{(t+2/4)}$ the j-th atom of the dictionary estimate at node l. Thus, the required permutation matrices in (9) can be computed using any algorithm $\mathcal{A}(\cdot)$ that solves the equivalent linear assignment problem, as

$$\mathbf{P}_{n,l}^{(t)} = \mathcal{A}\left(\mathbf{C}_{n,l}^{(t)}\right) . \qquad (11)$$

Several well-established algorithms can be found in literature for solving the linear assignment problem, or extensions of it, with a complexity of $\mathcal{O}(K^3)$ [18]. In our simulation results that appear in the following, we use the so-called Hungarian algorithm [19].

As we briefly also mentioned in the previous, the more the nodes act isolated from the others, the more likely it is that they come up with different dictionaries. This can be true even in the case where the data of all nodes can be represented by a common dictionary, since the permutation ambiguity is inherent in the cost function. Node cooperation helps the nodes gradually consent on a common dictionary, however, it comes at a communication cost. For saving energy due to communication, we propose to exchange the dictionary elements only once in every M iterations, instead of exchanging estimates at each iteration. Since in this case nodes work on their own for all intermediate time instants, the probability of computing dictionaries with different orderings is increased and thus the aforementioned alignment procedure should be performed prior to the combination step. The complete proposed algorithm is summarized in Table I.

INPUT: $\mathbf{Y}_n \in \mathbb{R}^{p \times q_n}$, K, λ, I, c_X, c_D, M, $a_{l,n}$
OUTPUT: Dictionary matrix $\mathbf{D}_n^{(t)}$

- Initialize $\mathbf{D}_n^{(-1)}$ with random numbers in $[0\ 1]$
- Normalize atoms $\boldsymbol{d}_{n,k}^{(0)} = \frac{1}{\left\|\boldsymbol{d}_{n,k}^{(-1)}\right\|_2} \boldsymbol{d}_{n,k}^{(-1)}, \quad k = 1, \ldots, K$
- Initialize $\mathbf{A}_n^{(0)}$ with zeros $K \times q_n$

FOR $t = 0$ TO ∞
  FOR $i = 1$ TO I
    - $\mathbf{A}_n^{(t+(i-1)/I+1/2I)} = \mathbf{A}_n^{(t+(i-1)/I)} + \lambda \frac{c_X}{\sqrt{K}} \left(\mathbf{D}_n^{(t)}\right)^{\mathrm{T}} \left(\mathbf{Y}_n - \mathbf{D}_n^{(t)} \mathbf{A}_n^{(t+(i-1)/I)}\right)$
    - $\mathbf{A}_n^{(t+i/I)} = \mathrm{SoftThreshold}_{\lambda \frac{c_X}{\sqrt{K}}} \left(\mathbf{A}_n^{(t+(i-1)/I+1/2I)}\right)$
  END
  - $\mathbf{D}_n^{(t+1/4)} = \mathbf{D}_n^{(t)} + \frac{c_D}{\left\|\mathbf{A}_n^{(t+1)}\right\|_{\mathrm{F}}} \left(\mathbf{Y}_n - \mathbf{D}_n^{(t)} \mathbf{A}_n^{(t+1)}\right) \left(\mathbf{A}_n^{(t+1)}\right)^{\mathrm{T}}$
  - $\boldsymbol{d}_{n,k}^{(t+2/4)} = \frac{1}{\left\|\boldsymbol{d}_{n,k}^{(t+1/4)}\right\|_2} \boldsymbol{d}_{n,k}^{(t+1/4)}, \quad k = 1, \ldots, K$
  IF (t MOD M) $= 0$
    - Send $\mathbf{D}_n^{(t+2/4)}$, listen for $\mathbf{D}_l^{(t+2/4)}$, $l \in \mathcal{N}_n \setminus \{n\}$
    - $\mathbf{P}_{n,l}^{(t)} = \arg\min_{\mathbf{P}} \left\| \mathbf{D}_n^{(t+2/4)} - \mathbf{D}_l^{(t+2/4)} \mathbf{P} \right\|_{\mathrm{F}}^2$, $l \in \mathcal{N}_n \setminus \{n\}$
    - Combine $\mathbf{D}_n^{(t+3/4)} = \sum_{l \in \mathcal{N}_n} \alpha_{l,n} \mathbf{D}_l^{(t+2/4)} \mathbf{P}_{n,l}^{(t)}$
    - $\boldsymbol{d}_{n,k}^{(t+1)} = \frac{1}{\left\|\boldsymbol{d}_{n,k}^{(t+3/4)}\right\|_2} \boldsymbol{d}_{n,k}^{(t+3/4)}, \quad k = 1, \ldots, K$
  ELSE
    - $\mathbf{D}_n^{(t+1)} = \mathbf{D}_n^{(t+2/4)}$
  END
END

TABLE I
THE PROPOSED "ADAPT-ALIGN-COMBINE" DISTRIBUTED DICTIONARY LEARNING ALGORITHM AT NODE n

III. NODE-SPECIFIC DISTRIBUTED DICTIONARY LEARNING

A typical application of distributed dictionary learning is in data compression. In this application the measurements collected by the nodes of the network need to be transmitted to a fusion center, for further processing. The required information, that the network must transmit, is the common dictionary and the sparse representation matrices of the individual nodes. Such a scheme has the benefit that the dictionary (i.e., the model) needs to be transmitted to the fusion center only once (or periodically, if it is time-varying), since it is common to all the nodes. On the other extreme, if each individual node computes a local dictionary, then all these dictionaries

should be transmitted to the fusion center. The latter approach, however, has the benefit of smaller representation error at the cost of increased communication cost.

In order to trade-off between the above mentioned two extremes (i.e., one common dictionary versus N individual dictionaries) we claim that the dictionary of each node n, could comprise K_g global atoms - common to all nodes - and K_l local atoms, where $K = K_g + K_l$. Thus, since nodes now try to estimate dictionaries that are only partially common, we refer to this approach as *node-specific distributed dictionary learning*. In particular, we model the dictionary of node n as

$$\mathbf{D}_n = \left[\mathbf{D}^{(g)} \quad \mathbf{D}_n^{(l)} \ , \right] \tag{12}$$

where $\mathbf{D}^{(g)} \in \mathbb{R}^{p \times K_g}$ is a matrix that contains the global atoms - common to all nodes - and $\mathbf{D}_n^{(l)}$ is a matrix that contains the local atoms at node n. In such a setting, the information required at the FC is: a) the global dictionary, b) each one of the local dictionaries and, c) the sparse approximation matrices of each node.

The proposed learning algorithm, for the node-specific distributed dictionary learning case, is an extension of the one in Table I. In particular, setting K_g in place of K, this algorithm can be used for the global part, i.e., the dictionary $\mathbf{D}^{(g)}$, where the ordering ambiguity is still relevant. Furthermore, regarding the local part $\mathbf{D}_n^{(l)}$, we define the residual signal at node n and time t as

$$\mathbf{R}_n^{(t)} = \mathbf{Y}_n - \mathbf{D}_n^{(g)(t+1)} \mathbf{A}_n^{(g)(t+1)} \ , \tag{13}$$

where $\mathbf{D}_n^{(g)(t+1)}$ and $\mathbf{A}_n^{(g)(t+1)}$ denote the updated global dictionary and the respective sparse approximation matrix estimates at time t. Then, node n must run any dictionary learning algorithm on this matrix of residuals.

IV. Numerical Results

A. A Distributed Image Denoising Example

In order to test the performance of the proposed approach, some computer simulations were conducted. In particular, we consider a simple network consisting of 5 nodes, where each node measures different but overlapping portions of an image, as depicted in Fig. 1. The original image has dimensions of 256×256 pixels, while each sensor measures a portion of the image, that is, 192×192 pixels. Each portion (sub-image) is split into patches of 8×8 pixels, thus creating the data matrices $\mathbf{Y}_1$ to $\mathbf{Y}_5$. Our scope is to compute a redundant dictionary $\mathbf{D}$ suitable for the entire image, in a distributed fashion, where nodes do not exchange their measurements; instead, they exchange their local estimates $\mathbf{D}_n^{(t)}$ of the dictionaries, following a diffusion strategy [17]. The topology of the network can be seen in Fig. 1.

The intensities of the original image are first normalized to lay in the interval $[0\ 1]$, then each portion is extracted, and in the sequel a zero mean additive white Gaussian noise is added to the data of each sub-image. A different noise variance is used for each sub-image, and in particular

$$\sigma_1^2 = 0.05, \sigma_2^2 = 0.04, \sigma_3^2 = 0.04, \sigma_4^2 = 0.05, \sigma_5^2 = 0.02 \ .$$

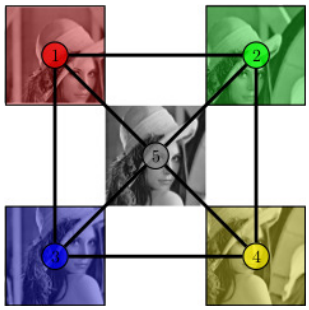

Fig. 1. Each sensor measures a different portion of the original image

The algorithm implemented is summarized in Table I. To trade-off sparsity versus representation accuracy, we select $\lambda = 0.1$ in the cost function given in (4). Also, we select the number of atoms to be $K = 128$ so that the redundant dictionary will be 64×128. For the sparse approximation step, each node performs $I = 40$ iterations, and the parameter $c_X = 0.25$. For the dictionary update step, we perform gradient descent with a suitable value for c_D so that the algorithms compared converge to the same cost. To save energy due to the exchange of the dictionaries, we choose to perform this exchange and combination once every M time iterations, and our scope in this experiment is to test the effect of this option. Also, recognizing the inherent permutation ambiguity in the cost function of equation (8), we propose to first align the dictionaries coming from neighbouring nodes and then proceed with the combination. We thus test the effect of the dictionary alignment procedure by comparing two schemes in which this alignment takes place (Adapt-Align-Combine) or not (Adapt-Combine). We solve the problem of optimally aligning two dictionaries using the so-called Hungarian algorithm [19]. Finally, the combination weights are computed by the equation

$$a_{l,n} = \frac{d_l(1/\sigma_l^2)}{\sum_{m \in \mathcal{N}_n} d_m(1/\sigma_m^2)} \ , \tag{14}$$

where d_l is the degree (number of neighbours) of node l.

In Fig. 2, we show the average (over the 5 nodes) cost as a function of the iteration index t. In particular, given the current estimate of the dictionary $\mathbf{D}_n^{(t)}$, we solve a sparse representation problem for all the data $\mathbf{Y}$, and we compute the global error. We sample the global error immediately after the combination process. In Fig. 2, we focus on four different cases, namely $M = 1$, $M = 10$, $M = 20$ and $M = 30$. For $M = 1$, i.e., when diffusion takes place 100% of the time, we note a negligible performance improvement that explains the relevant observation made in [8]. However, as M increases we notice a significant performance improvement in terms of the convergence rate.

B. Node-Specific Distributed Dictionary Learning

To demonstrate the effectiveness of the proposed node-specific distributed dictionary learning approach, we use a

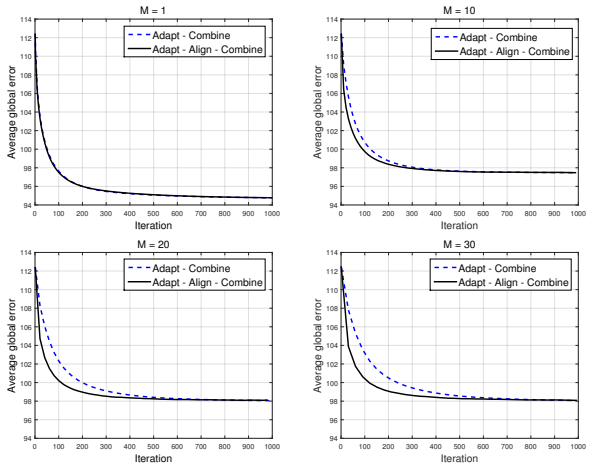

Fig. 2. Convergence comparison

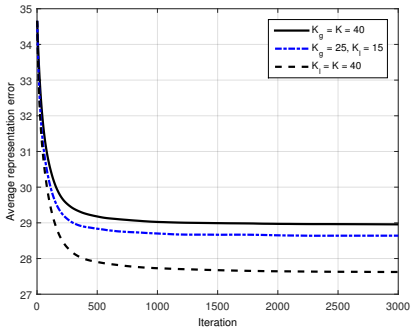

Fig. 3. Average representation error for Node-Specific DDL

synthetic dataset. In particular, we consider a two-node scenario, where the dictionary of the first node is generated from Gaussian random numbers with zero mean and unit variance. The dictionary of the second node is generated from uniform random numbers in the interval $(-0.5, 0.5)$. All atoms are normalized to unit length. Both dictionaries consist of $K = 40$ atoms of dimension $p = 20$. Using these dictionaries, we generate a random dataset for each of the two nodes, where each data vector is the sum of four randomly selected atoms.

In Fig. 3, the average representation error term (e.g. $\|\cdot\|_{\mathrm{F}}$ in (3)) versus the iteration index for three schemes is shown, namely one that considers only global atoms $(K_g = K = 40)$, one that considers only local atoms $(K_l = K = 40)$ and the node-specific case in which $K_g = 25$ and $K_l = 15$. The parameters of the algorithm were carefully chosen to achieve approximately the same number of non-zero elements in the sparse approximation matrices. From Fig. 3 it is clear that, the proposed node-specific approach does provide a trade-off between the extreme cases of a global dictionary and that of N local dictionaries.

V. Conclusions

In this work, diffusion-based distributed dictionary learning methods were considered. It was demonstrated that the permutation ambiguity of the cost function can be exploited so as to increase the convergence rate of the involved algorithm. Furthermore, a node-specific distributed dictionary learning problem was introduced and it was experimentally shown to provide some benefits in the case where the data of the nodes cannot generally be described by a common dictionary.

References

[1] I. Tosic and P. Frossard, "Dictionary learning," *Signal Processing Magazine, IEEE*, vol. 28, no. 2, pp. 27–38, March 2011.

[2] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer, 2010.

[3] S. Theodoridis, Y. Kopsinis, and K. Slavakis, "Sparsity-aware learning and compressed sensing: An overview," in *Academic Press Library in Signal Processing: Singal Processing Theory and Machine Learning*, R. Chellappa and S. Theodoridis, Eds. Academic Press, 2013, ch. 23.

[4] S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*. Academic Press, 2015.

[5] A. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, April 2014.

[6] V. Kekatos, E. Vlahos, D. Ampeliotis, G. Giannakis, and K. Berberidis, "A decentralized approach to generalized power system state estimation," in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2013 IEEE 5th International Workshop on*, Dec 2013, pp. 77–80.

[7] S. Chouvardas, Y. Kopsinis, and S. Theodoridis, "An online algorithm for distributed dictionary learning," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, April 2015, pp. 3292–3296.

[8] P. Chainais and C. Richard, "Learning a common dictionary over a sensor network," in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2013 IEEE 5th International Workshop on*, Dec 2013, pp. 133–136.

[9] J. Chen, Z. Towfic, and A. Sayed, "Online dictionary learning over distributed models," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, May 2014, pp. 3874–3878.

[10] ——, "Dictionary learning over distributed models," *Signal Processing, IEEE Transactions on*, vol. 63, no. 4, pp. 1001–1016, Feb 2015.

[11] J. Liang, M. Zhang, X. Zeng, and G. Yu, "Distributed dictionary learning for sparse representation in sensor networks," *Image Processing, IEEE Transactions on*, vol. 23, no. 6, pp. 2528–2541, June 2014.

[12] H. Raja and W. Bajwa, "Cloud k-svd: Computing data-adaptive representations in the cloud," in *Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on*, Oct 2013, pp. 1474–1481.

[13] ——, "Cloud k-svd: A collaborative dictionary learning algorithm for big, distributed data," *Signal Processing, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[14] H. Raja and W. U. Bajwa, "A convergence analysis of distributed dictionary learning based on the k-svd algorithm," in *Information Theory (ISIT), 2015 IEEE International Symposium on*, June 2015, pp. 2186–2190.

[15] V. Sindhwani and A. Ghoting, "Large-scale distributed non-negative sparse coding and sparse dictionary learning," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '12. New York, NY, USA: ACM, 2012, pp. 489–497. [Online]. Available: http://doi.acm.org/10.1145/2339530.2339610

[16] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, Mar. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756006.1756008

[17] F. Cattivelli and A. Sayed, "Diffusion lms strategies for distributed estimation," *Signal Processing, IEEE Transactions on*, vol. 58, no. 3, pp. 1035–1048, March 2010.

[18] M. Akgl, "The linear assignment problem," in *Combinatorial Optimization*, ser. NATO ASI Series, M. Akgl, H. Hamacher, and S. Tfeki, Eds. Springer Berlin Heidelberg, 1992, vol. 82, pp. 85–122.

[19] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. pp. 32–38, 1957. [Online]. Available: http://www.jstor.org/stable/2098689