FAST EDGE-CONSENSUS COMPUTING BASED ON BREGMAN MONOTONE OPERATOR SPLITTING

Kenta Niwa,[1] *Guoqiang Zhang*[2] *and W. Bastiaan Kleijn*[3]

[1]NTT Media Intelligence Laboratories, Japan
[2]University of Technology Sydney, Australia
[3]Victoria University of Wellington, New Zealand

ABSTRACT

Edge-consensus computing is a framework to optimize a global cost function when distributed nodes observe distinct data sets. The distributed primal-dual method of multipliers (PDMM) and distributed alternating direction method of multipliers (ADMM) find network-global optima for edge-consensus algorithms by exchanging variables rather than data sets among the nodes. Since the distributed PDMM follows traditional Peaceman-Rachford splitting, it has a faster convergence rate than the distributed ADMM. To further speed up the convergence rate, we propose a new edge-consensus computing algorithm based on *Bregman Peaceman-Rachford splitting*. In traditional Peaceman-Rachford splitting, the variable update is defined based on a Euclidean metric and the convergence rate and is a form of first-order gradient descent. By generalizing the metric to a Bregman divergence and designing the divergence adaptively, our fast edge-consensus computing algorithm corresponds to the Newton or an accelerated gradient descent method. The results of our experiments confirm that the proposed algorithm can significantly improve the convergence rate of edge-consensus computing over state-of-the-art algorithms.

***Index Terms*—** Distributed computing, convex optimization, monotone operator splitting (MOS), Bregman divergence

1. INTRODUCTION

Optimization algorithms are commonly used in a wide range of practical applications such as image classification, speech/audio signal processing, and natural language processing. In many cases, the optimization procedure uses data that are made available to one or more centralized (co-located) processing units. However, it is not always possible to make the data sets available to centralized processing units when the scale of the data set is very large or the processing units (network nodes) are dispersed over wide areas. In the near future, we predict that big data (e.g. speech/image/language) will be collected in spatially distributed nodes in a network. Therefore, it is natural to carry out optimization to obtain the desired results by *exchanging variables among the nodes* rather than the data sets themselves. This approach is a form of edge-consensus computing. The goal of this study was to construct a practical and fast edge-consensus computing algorithm for adapting a large amount of data accumulated one after another.

Significant advances have recently been made in running optimization algorithms over multiple processors. Example approaches include the hogwild! [1], elastic averaging stochastic gradient descent (SGD) [2], and communication-efficient coordinate ascent (COCOA) [3]. Monotone operator splitting (MOS)-based optimization methods [4, 5] are particularly attractive for distributed processing systems. MOS naturally facilitates parallel computation. Many parallel algorithms based on MOS are variants of the alternating direction method of multiplier (ADMM) [6, 7]. Since most distributed ADMM studies require a central node [8, 9], the paradigm must be modified to be suitable for edge-consensus computing [10, 11]. Although distributed ADMM studies [10, 11] are effective, their convergence rate is often relatively slow because it is equivalent to applying Douglas-Rachford splitting [12] to the constrained minimization problem [7, 13]. The distributed primal-dual method of multiplier (PDMM) [14, 15] results in faster convergence because it is based on Peaceman-Rachford splitting [16].

The contribution of this paper is a new variant of an edge-consensus computing algorithm that is faster than state-of-the-art distributed PDMM. In studies of Bregman ADMM [17] and, more recently, Bregman monotone operator splitting (B-MOS) [18], it was shown that a Euclidean metric is used in traditional MOS algorithms. This implies that the distributed PDMM follows a first-order gradient descent (GD) method [19]. We were able to significantly improve the convergence rate by following paradigms that resemble the Newton method or an accelerated gradient descent (AGD) method, e.g., [20]. That is, by generalizing the Euclidean metric to the Bregman divergence [21] and varying this divergence adaptively, we obtain a fast edge-consensus algorithm that is based on B-MOS.

This paper is organized as follows. We explain conventional algorithms in Sec. 2. We describe the proposed fast edge-consensus algorithm based on Bregman Peaceman-Rachford splitting in Sec. 3. After a discussion of the numerical experiments in Sec. 4, we conclude the paper in Sec. 5.

2. CONVENTIONAL ALGORITHMS

We define the problem to be solved in Sec. 2.1. We briefly discuss conventional algorithms to the problem introduced in Sec. 2.2.

2.1. Cost Formulation

Let us consider a computation network in which different data sets are collected in each set of V (≥ 2) distributed nodes (e.g. computation servers), as shown in Fig. 1. The edge structure is described with a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of V nodes and $\mathcal{E}$ denotes the set of undirected edges. Any convex closed proper (CCP) cost form, e.g., [22], such as the cross-entropy for multi-class classification problems or mean squared error (MSE) for regression problems, may be used. Even if the cost forms are common among V nodes, they would differ from each other because the data sets available for each node are different. The resulting CCP form at the i-th node is denoted as $F_i : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ and the variable realization at node i is $\mathbf{w}_i \in \mathbb{R}^m$. The edge-consensus computing problem seeks the $\mathbf{w}_i$ that minimizes the sum of local-node costs while constraining variable $\mathbf{w}_i$ to be identical among nodes.

The optimization problem is of the general form and given by

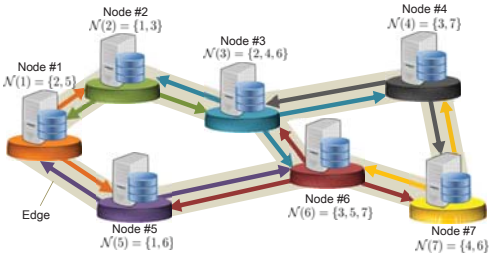

Fig. 1. Example of distributed computing system. V nodes are allowed to be connected according to arbitrary graph structure $\mathcal{G}(\mathcal{V},\mathcal{E})$, where there is no central server in them.

$$\inf_{\mathbf{w}_i}\sum_{i\in\mathcal{V}}F_i(\mathbf{w}_i)\quad \text{s.t.}\,\mathbf{A}_{i|j}\mathbf{w}_i+\mathbf{A}_{j|i}\mathbf{w}_j=\mathbf{0}\ \ (\forall i\in\mathcal{V},\forall j\in\mathcal{N}(i)),\quad(1)$$

where $\mathbf{A}_{i|j}\in\mathbb{R}^{m\times m}$ are parameters describing the constraints between the variables and $\mathcal{N}(i)=\{j\in\mathcal{V}|(i,j)\in\mathcal{E}\}$ is the set of neighbors connected with the i-th node. As we optimize variables to reduce the sum of local-node cost while striving to make the variables identical among the nodes, $\mathbf{A}_{i|j}$ and $\mathbf{A}_{j|i}$ are identity matrices with opposite signs:

$$\mathbf{A}_{i|j}=\begin{cases}\boldsymbol{I} & (i>j,\ j\in\mathcal{N}(i))\\ -\boldsymbol{I} & (j>i,\ j\in\mathcal{N}(i))\end{cases}.\quad(2)$$

Since it is usual to solve the dual problem for the constrained minimization as in (1), the associated Lagrangian is introduced:

$$\mathcal{L}=\sum_{i\in\mathcal{V}}\left(F_i(\mathbf{w}_i)-\sum_{j\in\mathcal{N}(i)}\left\langle\boldsymbol{\nu}_{i,j},\mathbf{A}_{i|j}\mathbf{w}_i\right\rangle\right),\quad(3)$$

where $\boldsymbol{\nu}_{i,j}\in\mathbb{R}^m$ denotes the dual variables w.r.t. a constraint along undirected edge (i,j). Thus, the dual problem form is given by

$$\sup_{\boldsymbol{\nu}_{i,j}}\inf_{\mathbf{w}_i}\mathcal{L}=-\inf_{\boldsymbol{\nu}_{i,j}}\sum_{i\in\mathcal{V}}F_i^\star\left(\sum_{j\in\mathcal{N}(i)}\mathbf{A}_{i|j}^{\mathrm{T}}\boldsymbol{\nu}_{i,j}\right),\quad(4)$$

where $F_i^\star$ is the convex conjugate of F_i [24]:

$$F_i^\star\left(\sum_{j\in\mathcal{N}(i)}\mathbf{A}_{i|j}^{\mathrm{T}}\boldsymbol{\nu}_{i,j}\right)=\sup_{\mathbf{w}_i}\left(\sum_{j\in\mathcal{N}(i)}\left\langle\boldsymbol{\nu}_{i,j},\mathbf{A}_{i|j}\mathbf{w}_i\right\rangle-F_i(\mathbf{w}_i)\right),\quad(5)$$

and $^{\mathrm{T}}$ denotes transposition.

To facilitate distributed computation in which each node variable w_i is updated autonomously/asynchronously, a *lifting* formalism is used e.g., [14, 15]. Thus, we split $\boldsymbol{\nu}_{i,j}$ into bidirectional dual variables $\{\boldsymbol{\lambda}_{i|j},\boldsymbol{\lambda}_{j|i}\}\in\mathbb{R}^m$. This enables variable $\boldsymbol{\lambda}_{i|j}$ to be transmitted from i to j autonomously/asynchronously for each node. However, since each pair of bidirectionalized $\{\boldsymbol{\lambda}_{i|j},\boldsymbol{\lambda}_{j|i}\}$ originates from $\boldsymbol{\nu}_{i,j}$, it must be constrained to be identical. As a result, the problem (4) is now reformulated by

$$\inf_{\boldsymbol{\lambda}_{i|j}}\sum_{i\in\mathcal{V}}F_i^\star\left(\sum_{j\in\mathcal{N}(i)}\mathbf{A}_{i|j}^{\mathrm{T}}\boldsymbol{\lambda}_{i|j}\right)\ \text{s.t.}\,\boldsymbol{\lambda}_{i|j}=\boldsymbol{\lambda}_{j|i},\ \forall i\in\mathcal{V},\forall j\in\mathcal{N}(i).\quad(6)$$

For further simplified notation, variables for each node/edge and related parameters are stacked in a vector/matrix form as $\mathbf{w}=\left[\mathbf{w}_1^{\mathrm{T}},\ldots,\mathbf{w}_V^{\mathrm{T}}\right]^{\mathrm{T}}$, $\boldsymbol{\lambda}=\left[\boldsymbol{\lambda}_{1|2}^{\mathrm{T}},\ldots,\boldsymbol{\lambda}_{1|V}^{\mathrm{T}},\ldots,\boldsymbol{\lambda}_{V|1}^{\mathrm{T}},\ldots,\boldsymbol{\lambda}_{V|V-1}^{\mathrm{T}}\right]^{\mathrm{T}}$, and $\mathbf{A}\in\mathbb{R}^{mV(V-1)\times mV}$. Therefore, we can now rewrite (6) as

$$\inf_{\boldsymbol{\lambda}}F^\star(\mathbf{A}^{\mathrm{T}}\boldsymbol{\lambda})\quad\text{s.t.}\,\boldsymbol{\lambda}_{i|j}=\boldsymbol{\lambda}_{j|i},\ \forall i\in\mathcal{V},\forall j\in\mathcal{N}(i),\quad(7)$$

where $F^\star$ is the convex conjugate of the sum over local-node cost functions $F(\mathbf{w})=\sum_{i\in\mathcal{V}}F_i(\mathbf{w}_i)$,

$$F^\star(\mathbf{A}^{\mathrm{T}}\boldsymbol{\lambda})=\sup_{\mathbf{w}}\left(\langle\boldsymbol{\lambda},\mathbf{A}\mathbf{w}\rangle-F(\mathbf{w})\right).\quad(8)$$

Problem (7) can now be rewritten as

$$\inf_{\boldsymbol{\lambda}}F^\star(\mathbf{A}^{\mathrm{T}}\boldsymbol{\lambda})+\delta_{(\boldsymbol{I}-\mathbf{P})}(\boldsymbol{\lambda}),\quad(9)$$

where the indicator function $\delta_{(\boldsymbol{I}-\mathbf{P})}$ is used to make $\boldsymbol{\lambda}_{i|j}$ and $\boldsymbol{\lambda}_{j|i}$ identical:

$$\delta_{(\boldsymbol{I}-\mathbf{P})}(\boldsymbol{\lambda})=\begin{cases}0 & (\boldsymbol{I}-\mathbf{P})\boldsymbol{\lambda}=\mathbf{0}\\ +\infty & (\text{otherwise})\end{cases},\quad(10)$$

where $\mathbf{P}$ is the permutation matrix that exchanges dual variables between connected nodes as $\boldsymbol{\lambda}_{j|i}\leftrightarrow\boldsymbol{\lambda}_{i|j}$.

It was shown in this subsection that the edge-consensus computing problem (1) is reformulated as a sum of two CCP functions (9). In the next subsection, we describe conventional algorithms for (9).

2.2. Conventional Algorithms Based on MOS

We now briefly explain several conventional algorithms of (9). The fixed point of (9) is found when its subgradient includes zero:

$$\mathbf{0}\in T_1(\boldsymbol{\lambda})+T_2(\boldsymbol{\lambda}),\quad(11)$$

where $T_1(\boldsymbol{\lambda})=\mathbf{A}\partial F^\star(\mathbf{A}^{\mathrm{T}}\boldsymbol{\lambda})$ and $T_2(\boldsymbol{\lambda})=\partial\delta_{(\boldsymbol{I}-\mathbf{P})}(\boldsymbol{\lambda})$ denote the subdifferential of each CCP function in (9), ∂ denotes the subdifferential operator, and $\in$ reflects that its output can be multi-valued when $F^\star$ includes discontinuous points, e.g., [4]. For the problem form in (11), MOS is effective in finding its fixed point.

In the distributed PDMM [14], [15], Peaceman-Rachford splitting [16], which is a MOS, is applied to (11). Although the detailed algorithmic form is not described here, it can reach the fixed point by iteratively updating variables $\{\mathbf{w}_i,\boldsymbol{\lambda}_{i|j}\}$ in the i-th node and exchanging a dual variable between connected nodes. Since each procedure is separate for each node, the variables can be updated autonomously/asynchronously.

Meanwhile, distributed ADMM based algorithms (e.g., [8, 9]) follow Douglas-Rachford splitting [12], which is another MOS. Although it has been assumed that a central server exists, our aim was to find optima by using a distributed computing system. For such a situation, it has been shown [15] that the cost form is modified, eventually resulting in (9), then Douglas-Rachford splitting is applied to (11). Since Douglas-Rachford splitting includes an averaged operator, e.g., [4], the convergence rate with the distributed ADMM (Douglas-Rachford splitting) is generally slower than the distributed PDMM (Peaceman-Rachford splitting).

In this subsection, we discussed the use of two MOS-based algorithms, distributed PDMM and ADMM. In constructing a solver, the convergence rate is an important factor. Although it is expected that the distributed PDMM would be faster than the distributed ADMM, we found possibilities to speed up the convergence rate of distributed PDMM. As shown in a study of Bregman ADMM [17] and our recent work on B-MOS [18], conventional B-MOS algorithms follow a first-order GD. By appropriately selecting the Bregman divergence metric, we will achieve a new MOS formulation that facilitates methods that resemble the Newton or an AGD method. The resulting B-MOS is discussed in Sec. 3 regarding the construction of our proposed edge-consensus computing algorithm.

3. PROPOSED ALGORITHM

After providing a brief summary of B-MOS in Sec. 3.1, we introduce our proposed algorithm that applies B-MOS to the problem formulated from (11).

3.1. Overview of B-MOS

For fast edge-consensus computing, we use B-MOS [18], which is a generalization of traditional Peaceman-Rachford and Douglas-Rachford splitting [16]. In traditional MOS algorithms, such as Peaceman-Rachford splitting and Douglas-Rachford splitting, a Euclidean metric is implicitly used in the update procedure. B-MOS

generalizes this Euclidean metric to Bregman divergence [21], which includes additional degrees of freedom, providing tuning variable space for fast convergence. An appropriate design will lead to a significantly faster convergence than conventional MOS algorithms.

We first define the Bregman divergence of $\boldsymbol{\lambda}$ and another point $\mathbf{z}$ as

$$B_D(\boldsymbol{\lambda} \,\|\, \mathbf{z}) = D(\boldsymbol{\lambda}) - D(\mathbf{z}) - \langle \nabla D(\mathbf{z}), \boldsymbol{\lambda} - \mathbf{z} \rangle , \tag{12}$$

where D is restricted to be a differentiable strictly convex function while its gradient satisfies $\nabla D(\mathbf{0}) = \mathbf{0}$. This is because applying the inverse operator of ∇D, which is denoted as $(\nabla D)^{-1}$, to both sides of (11) will not affect the fixed point condition:

$$\mathbf{0} \in (\nabla D)^{-1} \circ T_1(\boldsymbol{\lambda}) + (\nabla D)^{-1} \circ T_2(\boldsymbol{\lambda}), \quad (\text{if } \nabla D(\mathbf{0}) = \mathbf{0}), \tag{13}$$

where $\circ$ synthesizes two different operators. Since D modifies the variable space metric, its appropriate design is important for fast convergence rate. When we select $D(\boldsymbol{\lambda}) = \frac{1}{2\rho}\|\boldsymbol{\lambda}\|_2^2$ ($\rho > 0$), the Bregman divergence reduces to the Euclidean distance.

As shown in [18], reformulation of (11) introduces generalized Bregman Peaceman-Rachford (B-P-R) and Bregman Douglas-Rachford (B-D-R) splitting. An auxiliary dual variable $\mathbf{z}$ is assumed to be associated with $\boldsymbol{\lambda}$ by using the D-resolvent operator $R_i = (I + (\nabla D)^{-1} \circ T_i)^{-1}$ [25] as $\boldsymbol{\lambda} \in R_1(\mathbf{z})$. By reformulating (11), recursive variable update rules are obtained as

$$\mathbf{z} \in C_2 \circ C_1(\mathbf{z}) \qquad \text{(B-P-R splitting)}, \tag{14}$$

$$\mathbf{z} \in \alpha C_2 \circ C_1(\mathbf{z}) + (1-\alpha)\mathbf{z} \qquad \text{(B-D-R splitting)}, \tag{15}$$

where $C_i = (I + (\nabla D)^{-1} \circ T_i)^{-1} \circ (I - (\nabla D)^{-1} \circ T_i) = 2R_i - I$ is the D-Cayley operator [18] and $\alpha \in (0, 1)$ is an averaging coefficient. The variable update rules (14) and (15) are decomposed into a simpler procedures as

$$\mathbf{z}^{t+1/2} = C_1(\mathbf{z}^t), \tag{16}$$

$$\mathbf{z}^{t+1} = \begin{cases} C_2(\mathbf{z}^{t+1/2}) & \text{(B-P-R splitting)} \\ \alpha C_2(\mathbf{z}^{t+1/2}) + (1-\alpha)\mathbf{z}^t & \text{(B-D-R splitting)} \end{cases}. \tag{17}$$

Note that (14) and (15) coincide with the traditional (Euclidean) Peaceman-Rachford and Douglas-Rachford splitting, respectively when we select the Euclidean metric $D(\boldsymbol{\lambda}) = \frac{1}{2\rho}\|\boldsymbol{\lambda}\|_2^2$.

We now discuss the relationship between the convergence ratio through the D-Cayley operator and Bregman divergence design (D design). The property of T_i is assumed to be given by

$$\gamma_{\text{LB},i}\|\acute{\mathbf{z}} - \mathbf{z}\|_2 \le \|T_i(\acute{\mathbf{z}}) - T_i(\mathbf{z})\|_2 \le \gamma_{\text{UB},i}\|\acute{\mathbf{z}} - \mathbf{z}\|_2, \tag{18}$$

for any two different points $\{\mathbf{z}, \acute{\mathbf{z}}\}$, and where $0 \le \gamma_{\text{LB},i} \le \gamma_{\text{UB},i} < \infty$. Applying $(\nabla D)^{-1}$ to T_i modifies the property of T_i as

$$\sigma_{\text{LB},i}\|\acute{\mathbf{z}} - \mathbf{z}\|_2 \le \|(\nabla D)^{-1} \circ T_i(\acute{\mathbf{z}}) - (\nabla D)^{-1} \circ T_i(\mathbf{z})\|_2 \le \sigma_{\text{UB},i}\|\acute{\mathbf{z}} - \mathbf{z}\|_2, \tag{19}$$

where $0 \le \sigma_{\text{LB},i} \le \sigma_{\text{UB},i} < \infty$. As the theorem proof is noted in [18], the convergence ratio bound through the D-Cayley operator is deterministically given by

$$\|C_i(\mathbf{z}^t) - C_i(\mathbf{z}^{t-1})\|_2 \le \sqrt{1 - \frac{4\sigma_{\text{LB},i}}{(1+\sigma_{\text{UB},i})^2}}\, \|\mathbf{z}^t - \mathbf{z}^{t-1}\|_2 . \tag{20}$$

The minimum contraction by optimizing $\sigma_{\text{LB},i}$ given $\sigma_{\text{UB},i}$ is $\sigma_{\text{LB},i} = \min\left(\sigma_{\text{UB},i}, \frac{1}{4}(1+\sigma_{\text{UB},i})^2\right)$, and the contraction ratio is 0 only if

$$\sigma_{\text{LB},i} = 1, \quad \sigma_{\text{LB},i} = 1. \tag{21}$$

Thus, D should be designed so that $\{\sigma_{\text{LB},i}, \sigma_{\text{UB},i}\}$ approach 1. For $\sigma_{\text{LB},i} \neq 1$, the best choice is $\sigma_{\text{LB},i} = \sigma_{\text{UB},i}$, then the contraction ratio, denoted as η_i, satisfies $0 < \sqrt{1 - \frac{4\sigma_{\text{LB},i}}{(1+\sigma_{\text{UB},i})^2}} \le \eta_i \le 1$. This fact shows that the C_i is a non-expansive operator. If we can select D such that it approaches the condition (21) rather than the Euclidean metric, fast convergence rate can be obtained.

Algorithm 1 Proposed algorithm

Initialization of $\mathbf{w}_i^0$, $\tilde{\mathbf{z}}_{i|j}^0$
for $t = 0, \ldots, T-1$ **do**
  ▷ Primal-dual update in local node
  for all $i \in \mathcal{V}, j \in \mathcal{N}(i)$ **do**
  $\mathbf{w}_i^{t+1} = \arg\min_{\mathbf{w}_i}\Big(F_i(\mathbf{w}_i) + \sum_{j\in\mathcal{N}(i)} B_{D_{i|j}^\dagger}\left(\mathbf{A}_{i|j}\mathbf{w}_i \| \tilde{\mathbf{z}}_{i|j}^t\right)\Big)$
  $\tilde{\mathbf{z}}_{i|j}^{t+1/2} = \tilde{\mathbf{z}}_{i|j}^t - 2\mathbf{A}_{i|j}\mathbf{w}_i^{t+1}$
  ▷ Transmit/receive variable
  for all $i \in \mathcal{V}, j \in \mathcal{N}(i)$ **do**
  Transmit$_{j\to i}\left(\tilde{\mathbf{z}}_{j|i}^{t+1/2}\right)$
  $\tilde{\mathbf{z}}_{i|j}^{t+1} = \begin{cases} \tilde{\mathbf{z}}_{j|i}^{t+1/2} & \text{(B-P-R splitting)} \\ \alpha\tilde{\mathbf{z}}_{j|i}^{t+1/2} + (1-\alpha)\tilde{\mathbf{z}}_{i|j}^t & \text{(B-D-R splitting)} \end{cases}$
end for

3.2. Edge-consensus Computing Algorithm Based on B-MOS

First, the update rule for our edge-consensus computing algorithm is provided. For the subdifferential of convex conjugate function T_1, it is common that the primal $\mathbf{w}$ and dual variables $\mathbf{z}$ are alternately updated, e.g. [4]. For a nonlinearly transformed auxiliary variable $\tilde{\mathbf{z}} = \nabla D(\mathbf{z})$, the primal-dual update rule for (16) is given by

$$\mathbf{w}^{t+1} = \arg\min_{\mathbf{w}}\Big(F(\mathbf{w}) + B_{D^\dagger}\left(\mathbf{A}\mathbf{w} \| \tilde{\mathbf{z}}^t\right)\Big), \tag{22}$$

$$\tilde{\mathbf{z}}^{t+1/2} = \tilde{\mathbf{z}}^t - 2\mathbf{A}\mathbf{w}^{t+1}, \tag{23}$$

where the Bregman divergence penalty term in (22) is used to generalize a variable space metric and it includes a strictly convex function $D^\dagger$ that satisfies $\nabla D^\dagger = (\nabla D)^{-1}$. For the subdifferential of the indicator function T_2, the related D-Cayley operator becomes $C_2 = (\nabla D)^{-1}\mathbf{P}(\nabla D)$ (Sherson et al. described its basic derivation [15]), but the variable space is generalized by Bregman divergence instead of the Euclidean metric. The variable update rule for (17), where its input is $\tilde{\mathbf{z}}$, is given by using the permutation matrix in (10):

$$\tilde{\mathbf{z}}^{t+1} = \begin{cases} \mathbf{P}\tilde{\mathbf{z}}^{t+1/2} & \text{(B-P-R splitting)} \\ \alpha\mathbf{P}\tilde{\mathbf{z}}^{t+1/2} + (1-\alpha)\tilde{\mathbf{z}}^t & \text{(B-D-R splitting)} \end{cases}. \tag{24}$$

This indicates that the dual auxiliary variables $\tilde{\mathbf{z}}_{i|j}$ are exchanged between connected nodes. The resulting algorithm in a distributed node calculation manner is summarized in **Algorithm 1**.

Next, we explain a D and $D^\dagger$ design method for fast convergence. As a method of approximately satisfying (21), we use quadratic forms that satisfies $\nabla D^\dagger = (\nabla D)^{-1}$ and $\nabla D(\mathbf{0}) = \mathbf{0}$ as

$$D(\boldsymbol{\lambda}) = \frac{1}{2}\left\langle \mathbf{M}_{F^\star}(\mathbf{z})\boldsymbol{\lambda}, \boldsymbol{\lambda} \right\rangle , \tag{25}$$

$$D^\dagger(\boldsymbol{\lambda}) = \frac{1}{2}\left\langle \mathbf{M}_{F^\star}^{-1}(\mathbf{z})\boldsymbol{\lambda}, \boldsymbol{\lambda} \right\rangle . \tag{26}$$

For a Hessian design, Newton, AGD and GD methods can be utilized:

$$\mathbf{M}_{F^\star}^{-1}(\mathbf{z}) = \begin{cases} \mathbf{H}_F(\mathbf{z}) & \text{(Newton)} \\ \mathbf{L}_F(\mathbf{z}) & \text{(AGD)} \\ \frac{1}{\rho}\mathbf{I} & \text{(GD)} \end{cases}, \tag{27}$$

where $\mathbf{H}_F(\mathbf{z})$ uses a second-order gradient of the cost F, and $\mathbf{L}_F(\mathbf{z})$ is a diagonal matrix whose elements are scaled identical with $\mathbf{H}_F(\mathbf{z})$. Compared with GD, the variable space metric will be modified by using the Newton or AGD method such that approaches (21).

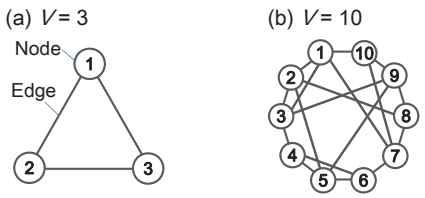

Fig. 2. Graph models $\mathcal{G}(\mathcal{V}, \mathcal{E})$

Table 1. Parameters

Number of graph structures	2 $(V = 3, 10)$
Number of classes to be discriminated, K	10
Dimension of input data, m	7,850 $=(28\times28+1)K$
Averaging coefficient, α	0.5
Step size, ρ	$1e^{-4}$
Regularization coefficient, μ	0.01

For fast convergence in the proposed algorithm, it may be better to choose (i) B-P-R splitting rather than B-D-R splitting because the averaged operator (e.g. [4]) will slow the convergence rate and (ii) design D such that follows Newton or AGD rather than GD (Euclidean metric) because it will modify the metric to approach condition (21). When we select the Euclidean metric in D, **Algorithm 1** is reduced to a conventional distributed PDMM and ADMM.

4. EXPERIMENTS

We evaluated the effectiveness of the proposed edge-consensus computing algorithm through numerical experiments.

4.1. Experimental Conditions

To show the efficacy of the proposed algorithm, we solved distributed multi-class classification problems that were based on the MNIST data set [27]. The data set is composed of images of $K = 10$ class handwritten digits ($U_{\rm tr} = 60,000$ training and $U_{\rm ev} = 10,000$ evaluation data). We prepared graph structures with $V = 3$ and $V = 10$. The graphs are shown in **Fig. 2**. The graph with $V = 10$ was designed such that both long and short paths were present. The amount of data placed on the i-th node is denoted as $U(i)$ with $\sum_{i=1}^{V} U(i) = U_{\rm tr}$. The training data set at the i-th node $\{\boldsymbol{\phi}_{i,1}, \ldots, \boldsymbol{\phi}_{i,U(i)}\}$ was randomly selected. A one-hot-function $s_{i,k,u} \in \{0, 1\}$ indicates whether an image belongs to class k. The experimental parameters are summarized in **Table 1**.

For this problem, the cross-entropy with a L_1 regularization was used as a cost: $F(\mathbf{w}) = \sum_{i=1}^{V} F_i(\mathbf{w}_i)$, where

$$F_i(\mathbf{w}_i) = \sum_{u=1}^{U(i)} \sum_{k=1}^{K} -\frac{s_{i,k,u}}{U(i)} \log d_{i,k,u}(\mathbf{w}_{i,k}) + \mu \|\mathbf{w}_{i,k}\|_1, \qquad (28)$$

where $\mathbf{w}_i = [\mathbf{w}_{i,1}^{\rm T}, \ldots, \mathbf{w}_{i,K}^{\rm T}]^{\rm T}$, and $d_{i,k,u}(\mathbf{w}_{i,k})$ is the soft-max function for $a_{i,k,u}(\mathbf{w}_{i,k}) = \langle \mathbf{w}_{i,k}, \boldsymbol{\phi}_{i,u} \rangle$ as:

$$d_{i,k,u}(\mathbf{w}_{i,k}) = \frac{\exp(a_{i,k,u}(\mathbf{w}_{i,k}))}{\sum_{j=1}^{K} \exp(a_{i,j,u}(\mathbf{w}_{i,j}))}. \qquad (29)$$

The $\mathbf{w}$-update procedure in **Algorithm 1** was implemented as an iterative fashion because it was difficult to represent as a closed variable update form.

Six algorithms were compared: conventional (i) distributed ADMM (D-ADMM), (ii) distributed PDMM (D-PDMM), the proposed (iii) B-P-R with AGD method, (iv) B-P-R with Newton method (v) B-D-R with AGD method, (vi) B-D-R with Newton method, as summarized in **Algorithm 1**. The variables, e.g., $\mathbf{w}_i$, were randomly initialized using a normal distribution $\mathrm{Norm}(0, 0.1)$. To investigate the robustness against the asynchronous variable exchange among nodes, the exchange frequency rate was controlled to be once per three iterations. Although this exchange frequency rate slows the convergence rate compared to the synchronous variable exchange case, the differences in the convergence curves did not change significantly. Thus, we present the experimental results for only the asynchronous case.

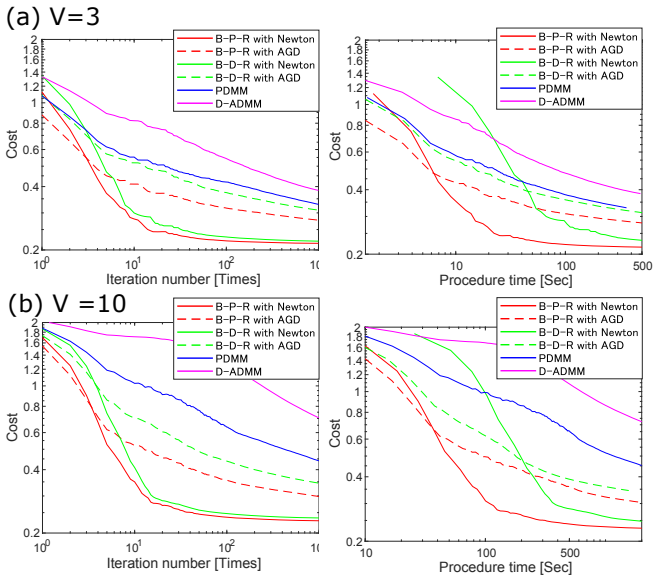

Fig. 3. Convergence curves when variables were exchanged randomly among V nodes at rate of once per three iterations

4.2. Experimental Results

We evaluated the six algorithms with the cost convergence curves. To do this, T=1, 000 iterations were computed on a GPU (NVIDIA GeForce GTX 1080 Ti) and both the cost value $F(\mathbf{w}^t) = \sum_{i\in\mathcal{V}} F_i(\mathbf{w}_i^t)$ and procedure time were recorded. Note that our goal was to study the convergence rate. The ultimate accuracy provided by the algorithms used for our experiment is limited by the simple formulation (29). Thus, the final digit-recognition accuracy is low compared to the state-of-the-art, but this was irrelevant to our study.

Figure 3 shows the relationships between the six algorithms and the cost value $F(\mathbf{w}^t)$. The results are shown separately for iteration number and procedure time for each graph. From the convergence curves, B-P-R with Newton had the fastest convergence rate and B-D-R with Newton was the second best. Although the computational cost of B-P-R/B-D-R with Newton was the heaviest per iteration, they were significantly faster than the conventional algorithms. Although only dual variables were exchanged between connected nodes, the differences in variable $\mathbf{w}_i$ between nodes were very small.

5. CONCLUSION

We proposed a fast edge-consensus computing algorithm based on B-MOS. We replaced the Euclidean variable space metric used in conventional algorithms with the Bregman divergence and it that was appropriately designed to follow the Newton or AGD method. The proposed algorithm finds the global fixed point by exchanging the dual variables between nodes in an arbitrary edge structure. Our experimental results show that the convergence rate is improved significantly under various conditions.

6. REFERENCES

[1] B. Recht, C. Re, S. Wright and F. Niu, Hogwild: a lock-free approach to parallelizing stochastic gradient descent, *Advances in Neural Information Processing Systems*, 693–701, 2011.

[2] S. Zhang, A. E. Choromanska and Y. LeCun, Deep learning with elastic averaging SGD, *Advances in Neural Information Processing Systems*, 685–693, 2015.

[3] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann and M. I. Jordan, Communication-efficient distributed dual coordinate ascent, *Advances in Neural Information Processing Systems*, 3068–3076, 2014.

[4] E. K. Ryu and S. Boyd, Primer on monotone operator methods, *Applied and Computational Mathematics*, 15(1), 3–43, 2016.

[5] H. H. Bauschke and P. L. Combettes, *Convex analysis and monotone operator theory in Hilbert spaces*, Springer, 2017.

[6] D. Gabay and B. Mercier, A dual algorithm for the solution of nonlinear variational problems via finite element approximation, *Computers & Mathematics with Applications*, 2(1), 17–40, 1976.

[7] S. Boyd and N. Parikh and E. Chu and B. Peleato and J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, *Advances in Neural Information Processing Systems*, 3, 1–122, 2011.

[8] E. Wei and A. Ozdaglar, Distributed alternating direction method of multipliers, *Institute of Electrical and Electronics Engineers (IEEE)*, 2012.

[9] R. Zhang and J. Kwok, Asynchronous distributed ADMM for consensus optimization, *Proc. of 31st international conference on machine learning (ICML'14)*, 1701–1709, 2014.

[10] Q. Ling and A. Ribeiro, Decentralized linearized alternating direction method of multiplier, *Proc. of IEEE international conference on acoustics, speech and signal processing (ICASSP'14)*, 5447–5451, 2014.

[11] A. Mokhtari and W. Shi and Q. Ling and A. Ribeiro, Decentralized quadratically approximated alternating direction method of multipliers, *Proc. of IEEE global conference on signal and information processing (GlobalSIP'15)*, 795–799, 2015.

[12] J. Douglas and H. H. Rachford, On the numerical solution of heat conduction problems in two and three space variables, *Transactions of the American Mathematical Society*, 82(2), 421–439, 1956.

[13] J. Eckstein and D. P. Bertsekas, On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators, *Mathematical Programming, Springer*, 55, 293–318, 1992.

[14] G. Zhang and R. Heusdens, Distributed optimization using the primal-dual method of multipliers, *IEEE Transactions on Signal and Information Processing over Networks*, 4(1), 173–187, 2017.

[15] T. Sherson, R. Heusdens and W. B. Kleijn, Derivation and analysis of the primal-dual method of multipliers based on monotone operator theory, *arXiv preprint arXiv:1706.02654*, 2017.

[16] D. W. Peaceman and H. H. Rachford, The numerical solution of parabolic and elliptic differential equations, *Journal of the SIAM*, 3(1), 28–41, 1955.

[17] H. Wang and A. Banerjee, Bregman alternating direction method of multipliers, *Advances in Neural Information Processing Systems*, 2816–2824, 2014.

[18] K. Niwa and W. B. Kleijn, Bregman monotone operator splitting, *arXiv preprint arXiv:1807.04871*, 2018.

[19] A. Cauchy, Méthode générale pour la résolution des systemes d'équations simultanées, *Comptes Rendus de l'Academie des Sciences*, 25, 536–538, 1847.

[20] R. Johnson and T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, *Advances in Neural Information Processing Systems*, 315–323, 2013.

[21] L. M. Bregman, The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming, *USSR Computational Mathematics and Mathematical Physics*, 7(3), 200–217, 1967.

[22] R. T. Rockafellar, *Convex analysis*. Princeton university press, 1970.

[23] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[24] W. Fenchel, On conjugate convex functions, *Canad. J. Math*, 1, 73–77, 1949.

[25] H. H. Bauschke, M. Jonathan and P. L. Combettes, Bregman monotone optimization algorithms, *SIAM Journal on control and optimization*, 42(2), 596–636, 2003.

[26] K. Lange, D. R. Hunter and I. Yang, Optimization transfer using surrogate objective functions. *Journal of computational and graphical statistics, Taylor & Francis*, 9(1), 1–20, 2000.

[27] Y. LeCun, C. Cortes and C. J. Burges, MNIST handwritten digit database, *AT&T Labs. Available: http://yann.lecun.com/exdb/mnist*, 2010.