Graph-aware Weighted Hybrid ADMM for Fast Decentralized Optimization

Meng Ma Georgios B. Giannakis
Department of ECE and Digital Technology Center, University of Minnesota
Emails: {maxxx971, georgios}@umn.edu

***Abstract*—Distributed optimization has gained popularity in many areas because it can to cope with the increasing volume of data, and the corresponding demand for computational resources. A popular distributed solver is the alternating direction method of multipliers (ADMM). Fully decentralized (D) ADMM arises when node-to-node communications have to abide by the underlying network connectivity. DADMM's convergence however, slows down as the network diameter grows large. To deal with this challenge, the recently proposed hybrid (H) ADMM provides considerable performance boost over DADMM by exploiting local topology information. But HADMM only applies to unweighted graphs. The present contribution develops a weighted hybrid (WH) consensus-based ADMM approach that can deal with weighted graphs. The resultant scheme further improves the performance of HADMM through graph-aware weight tuning. Theoretical analysis offers convergence guarantees and establishes linear convergence rate, while numerical tests on various graphs demonstrate the WHADMM merits.**

***Index Terms*—decentralized optimization, ADMM, weighted ADMM, hybrid ADMM**

I. Introduction

Unprecedented amounts of data and the demand for computational resources involved in large-scale problems motivate well the growing interest for distributed processing. Playing a key role in distributed systems, decentralized optimization has attracted increasing attention in many areas, including machine learning [1], Internet-of-Things (IoT) [2], as well as distributed detection and estimation [3], [4], to name a few. In the context of distributed optimization, a group of networked computing nodes, each holding its own privately available data, work together to minimize a certain objective function. Every node takes decision by minimizing its local cost function, as well as by collaborating with others.

Among various solvers of decentralized optimization problem, the alternating method of multipliers (ADMM) stands out because of its simplicity, fast convergence, and decomposability [5], [6]. Depending on whether or not a central coordinator is involved, the distributed optimization task can be solved in a centralized [6] or decentralized manner [3], [7], which we abbreviate as CADMM and DAMM. The present work focuses on the latter. In a nutshell, each iteration of DADMM comprises: (i) an update step, where each node decreases its own cost based on information of its neighbors; and (ii) a communication step, where nodes exchange information with their single-hop neighbors. DADMM treats each node equally regardless of how many neighbors it is connected to. This extra information could be used to characterize the importance of a node and possibly leads to smart ways of solving the same problem. This paper introduces a weighted hybrid ADMM (WHADMM) solver that relies on the same two-step procedure, but is capable of effectively leveraging local topology information.

This work is supported in part by NSF grants 10121, 12321, 12345.

Related works. There are many approaches to solving distributed optimization problems, including those based on (sub)gradients [8], Nesterov accelerated (sub)gradients [9], dual averaging [10], gossip [11], ADMM [3], [7], and accelerated ADMM [12]. Among those, ADMM features the ability to decompose complex cost functions to simple sub-problems that can be easily handled, and even enjoys fast convergence to moderately accurate solutions [6]. When cost functions are strongly convex and Lipschitz continuous ADMM converges linearly to optimal solutions [7], [13], [14]. However, most algorithms consider only unweighted graphs, and treat all edges equally. Weighted decentralized (WD) ADMM accounts for edge weights, and is known to outperform its unweighted counterparts [15].

Existing ADMM-based algorithms achieve consensus by exchanging information with single-hop neighbors while being unaware of local topology. Our hybrid (H) ADMM carefully considers local neighborhood structure to enable faster information flow so as to achieve better convergence [16]. However, HADMM works only for undirected graphs.

Contributions. The present paper introduces WHADMM to cope with weighted graphs, and also exploits topology related information. Moreover, for certain types of networks, WHADMM can distinguish important from unimportant ones, and emphasize more the role of critical edges by assigning larger weights. Unlike WDADMM, which works best for graphs with clusters [15], WHADMM shows improvement over DADMM on almost any kind of graph.

II. Background and problem statement

Distributed optimization typically deals with minimizing an aggregate cost function $f(\mathbf{x}) = \sum_i f_i(\mathbf{x})$, by finding a common optimal decision vector $\mathbf{x} \in \mathbb{R}^p$, where each local cost f_i is privately available to node i only, while it may also contain node-private data [3]. The decision variable $\mathbf{x}$ is shared by all nodes, which couples the cost summands and precludes parallel processing. A common decoupling strategy is to create local copies of the decision vector at each node.

The optimal solution can be obtained by enforcing consensus across all network nodes to ensure all local copies agree with the global decision variable. Such a consensus-based technique starts with the following concrete problem formulation

$$\begin{aligned} \min_{\mathbf{x}} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ \text{s. to} \quad & \mathbf{x}_1 = \mathbf{x}_2 = \ldots = \mathbf{x}_N \end{aligned} \tag{1}$$

where N is the number of nodes and $\mathbf{x}_i$ is the local copy of global decision variable at node i. To reach consensus, each node needs to communicate with others. Such communication may be constrained by many factors, such as location, physical connectivity, or privacy concerns.

The communication constraints of a network can be conveniently specified by a graph, denoted by a tuple $\mathscr{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ is the vertex set corresponding to N networked nodes, and $\mathcal{E} := \{(i,j) | i, j \in \mathcal{V}\}$ is the edge set indicating the connectivity between pairs of nodes. Nodes i and j can communicate if $(i,j) \in \mathcal{E}$. We consider possible weighted but undirected graphs with no self-loops.

The communication constraints of HADMM are the same as those of DADMM, but the actual exchange of information is differently represented by a hypergraph, in order to take advantage of the underlying topology information. A hypergraph is also a tuple $\mathscr{H} = (\mathcal{E}, \mathcal{V})$, where $\mathcal{V} := \{1, 2, \ldots, N\}$ is the vertex set and $\mathcal{E} := \{\mathcal{E}_i | \mathcal{E}_i \subseteq V\}$ is the edge set, each edge $\mathcal{E}_i$ now being a subset of $\mathcal{V}$. A hyperedge may involve multiple nodes. Nodes in the same hyperedge are neighbors, and can thus exchange information. The hypergraph boils down to a simple graph when every hyperedge comprises exactly two nodes, that is, $|\mathcal{E}_i| = 2$. Therefore, hypergraphs generalize the notion of simple graphs, and provide a unifying means of describing communication constraints.

Let $\mathscr{H}$ be a hypergraph with N nodes and M hyperedges. The incidence matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$ of $\mathscr{H}$ is defined with $c_{ij} = 1$ if node i and edge j are connected, and $c_{ij} = 0$ otherwise. The degree of node i is the total number of incident edges, namely $d_i = \sum_j C_{ij}$; the degree of edge j is the total number of incident nodes, meaning $e_j = \sum_i c_{ij}$. Let $\mathbf{D} := \mathrm{diag}(\mathbf{d})$ and $\mathbf{E} := \mathrm{diag}(\mathbf{e})$ be diagonal matrices, where $\mathbf{d}$ and $\mathbf{e}$ are vectors collecting $\{d_i\}$ and $\{e_j\}$.

Problem statement. Given N networked nodes, whose connectivity is described by a hypergraph $\mathscr{H}$, the decentralized optimization problem can be reformulated as

$$\begin{aligned} \underset{\{\mathbf{x}_i\}, \{\mathbf{z}_j\}}{\text{minimize}} \quad & \sum_i f_i(\mathbf{x}_i) \\ \text{subject to} \quad & \mathbf{x}_i = \mathbf{z}_j \quad i \in \mathcal{E}_j \end{aligned} \tag{2}$$

where $\mathbf{z}_j \in \mathbb{R}^p$ is an auxiliary variable associated with hyperedge j, playing a role similar to a *local fusion center* [16]. As long as the hypergraph is connected, problems (2) and (1) are equivalent, and thus admit the same solution(s).

III. Weighted hybrid ADMM

The decentralized optimization problem (2) can be solved efficiently using HADMM. By creating hypergraphs properly, HADMM can take advantage of local connectivity, achieve faster convergence, and save communication cost [16].

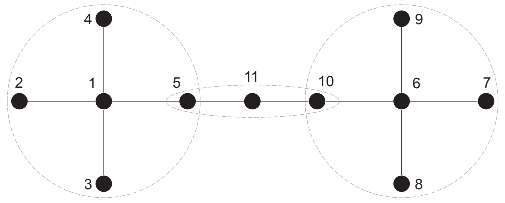


Fig. 1. A graph consisting of two clusters. Solid black circles represent nodes, solid lines indicate connectivity between nodes and dashed circles identify hyperedges.

Example. Consider the graph in Fig. 1. There are three hyperedges, namely $\mathcal{E}_1 := \{1 \ldots 5\}$, $\mathcal{E}_2 := \{6 \ldots 10\}$, and $\mathcal{E}_3 := \{5, 10, 11\}$, shown as gray dashed circles. Iterative steps of ADMM solving (2) involve information exchanges between $\mathbf{x}_2$ and $\mathbf{z}_1$, which can be effected via communication of node 2 with node 1, upon realizing that $\mathbf{z}_1$ can be viewed as a *virtual fusion center* [16] residing in node 1. Similar tricks can be devised for $\mathcal{E}_2$ and $\mathcal{E}_3$. By creating *virtual fusion centers*, HADMM can employ topology information using the same communication network as DADMM.

HADMM works only for unweighted graphs, treating every edge equally. WHADMM, on the other hand, takes the importance of an edge into account as expressed by its edge weight. Even when weights are not readily available, WHADMM can identify important edges and assign to them larger weights.

Let $\mathcal{D} \in \mathbb{R}^{N \times N}_{++}$ denote the set of diagonal positive definite matrices, and $\mathcal{C} \in \mathbb{R}^{N \times N}_{+}$ with nonzero (i,j)th entry if and only if $c_{ij} = 1$. Let also $\mathbf{X} := [\mathbf{x}_1^\top; \ldots; \mathbf{x}_N^\top] \in \mathbb{R}^{N \times p}$, and $\mathbf{Y} := [\mathbf{y}_1^\top; \ldots; \mathbf{y}_N^\top] \in \mathbb{R}^{N \times p}$, where $\mathbf{y}_i \in \mathbb{R}^p$ is the concatenation of dual variables $\mathbf{y}_i \in \mathbb{R}^p$. The WHADMM amounts to running the following iterations (indexed by k)

$$\begin{aligned} \mathbf{X}^{k+1} &= \underset{\mathbf{X}}{\operatorname{argmin}}\, f(\mathbf{X}) + \frac{\rho}{2}\langle \mathbf{X}, \mathbf{D}_w \mathbf{X} \rangle + \langle \mathbf{X}, \mathbf{Y}^k - \rho \mathbf{H}\mathbf{H}^\top \mathbf{X}^k \rangle \\ \mathbf{Y}^{k+1} &= \mathbf{Y}^k + \rho(\mathbf{D}_w - \mathbf{H}\mathbf{H}^\top)\mathbf{X}^{k+1} \end{aligned} \tag{3}$$

where $\mathbf{D}_w \in \mathcal{D}$, $\mathbf{H} \in \mathcal{C}$, and ρ is some tunable parameter.

Remark. *Although ρ can be absorbed by $\mathbf{D}$ and $\mathbf{H}$, we keep it here for consistency with HADMM and DADMM.*

If at least one optimal solutions of problem (2) exists, sequences generated by WHADMM converge to one optimal solution, provided the following conditions are satisfied.

$$\mathbf{D}_w - \mathbf{H}\mathbf{H}^\top \succeq \mathbf{0} \tag{4}$$

$$\mathbf{H}\mathbf{H}^\top \succeq \mathbf{0} \tag{5}$$

$$\mathrm{Null}(\mathbf{D}_w - \mathbf{H}\mathbf{H}^\top) = \mathrm{span}\{\mathbf{1}\} \tag{6}$$

where $\mathrm{Null}(\mathbf{A})$ denotes the null space of $\mathbf{A}$ that is given by $\{\mathbf{x} | \mathbf{A}\mathbf{x} = \mathbf{0}\}$. These conditions ensure $\mathbf{D}_w - \mathbf{H}\mathbf{H}^\top$ is a valid Laplacian matrix of the hypergraph [17].

The WHADMM algorithm (3) is amenable to decentralized implementation. To recognize this, observe both $\mathbf{D}$ and $\mathbf{H}$

have special structure that renders (3) separable across nodes. If $\mathbf{H} = \mathbf{C}$, the jth row of $\mathbf{Z} = \mathbf{H}^\top \mathbf{X}$ can be expressed as

$$\mathbf{z}_j = \sum_{i=1}^{N} c_{ij}\mathbf{x}_i = \sum_{i=1}^{N} \mathbb{1}_{\{i\in\mathcal{E}_j\}}\mathbf{x}_i = \sum_{i\in\mathcal{E}_j} \mathbf{x}_i. \tag{7}$$

Consequently, if $\mathbf{V} = \mathbf{H}\mathbf{Z} = \mathbf{H}\mathbf{H}^\top\mathbf{X}$, the ith row of $\mathbf{V}$ is given by

$$\mathbf{v}_i = \sum_{j=1}^{M} c_{ij}\mathbf{z}_j = \sum_{j=1}^{M} \mathbb{1}_{i\in\mathcal{E}_j} \sum_{k=1}^{N} \mathbb{1}_{k\in\mathcal{E}_j}\mathbf{x}_k = \sum_{j:i\in\mathcal{E}_j}\sum_{k\in\mathcal{E}_j} \mathbf{x}_k. \tag{8}$$

It is clear that $\mathbf{v}_i$ is available to node i through communication with neighbors. At node i, consider now the explicit WHADMM iterations

$$\begin{aligned} \mathbf{x}_i^{k+1} &= \operatorname*{argmin}_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \frac{\rho}{2} d_i^w \|\mathbf{x}_i\|^2 + \langle \mathbf{x}_i, \mathbf{y}_i^k - \rho\mathbf{v}_i^k\rangle \\ \mathbf{y}_i^{k+1} &= \mathbf{y}_i^k + \rho(d_i^w\mathbf{x}_i^{k+1} - \mathbf{v}_i^{k+1}) \end{aligned} \tag{9}$$

where d_i^w is the ith diagonal entry of $\mathbf{D}_w$, and $\mathbf{v}_i$ is defined as in (8). Node i only needs $\mathbf{v}_i$, which can be obtained by communicating with neighbors, to update $\mathbf{x}_i^{k+1}$ and $\mathbf{y}_i^{k+1}$.

A. Relationships with existing algorithms

The WHADMM algorithm is closely related to several ADMM-based decentralized optimization algorithms. It turns out that WHADMM reduces to HADMM when $\mathbf{H} = \mathbf{C}\mathbf{E}^{-1/2}$. To see this, notice that HADMM iterations are equivalent to (cf. [16])

$$\begin{aligned} \mathbf{X}^{k+1} &= \operatorname*{argmin}_{\mathbf{X}} f(\mathbf{X}) + \frac{\rho}{2}\mathbf{X}^\top\mathbf{D}\mathbf{X} + \\ &\qquad \langle \mathbf{X}, \mathbf{Y}^k - \rho\mathbf{C}\mathbf{E}^{-1}\mathbf{C}^\top\mathbf{X}^k\rangle \\ \mathbf{Y}^{k+1} &= \mathbf{Y}^k + \rho(\mathbf{D} - \mathbf{C}\mathbf{E}^{-1}\mathbf{C}^\top)\mathbf{X}^{k+1} \end{aligned} \tag{10}$$

where $\mathbf{Z}^k = \mathbf{E}^{-1}\mathbf{C}^\top\mathbf{X}^k$ is eliminated. Since $\mathbf{C}\mathbf{E}^{-1}\mathbf{C}^\top = \mathbf{H}\mathbf{H}^\top$, algorithm (3) is equivalent to (10). Consequently, as discussed in [16], we can recover popular distributed ADMM variants depending on the topology of the underlying $\mathscr{G}$.

- i) when $\mathscr{G}$ is a star graph, then $\mathscr{H}$ has only one hyperedge comprising all nodes; that is $\mathbf{C} = \mathbf{1}$, and we recover centralized ADMM [5], [6];
- ii) when $\mathscr{G}$ is an ordinary graph, then each hyperedge $\mathcal{E}_j$ involves exactly two nodes, and we recover fully the decentralized ADMM [3], [7]; and
- iii) when $\mathscr{H}$ is an ordinary hypergraph, which is the most general case, we recover HADMM [16].

IV. Convergence analysis

A. Convergence

ADMM has been extensively studied for decades, and its convergence is well established, see [7], [12]–[14], [18], [19]. Derived from ADMM, our WHADMM is guaranteed to converge under mild conditions, see e.g. [6, §3.2] for details.

B. Linear convergence rate

When cost functions are strongly convex and Lipschitz continuous, sequences obtained by ADMM converge linearly to some optimal solution [13], [14]. However, no explicitly rate estimate is provided in [14], and full row rank of constraint coefficient matrices is required for the approach in [13]. Linear convergence of DADMM was established in [7], where the full row rank condition is absent. Similar results can be found in [16] since DADMM is a special case of HADMM, and the analysis of HADMM carries over to DADMM.

Let us consider now the linear convergence rate of WHADMM, starting with the definition of the semi-norm induced by matrix $\mathbf{G}$ as $\|\mathbf{x}\|_{\mathbf{G}}^2 := \mathbf{x}^\top\mathbf{G}\mathbf{x}$, where

$$\mathbf{G} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}\mathbf{H}^\top \end{bmatrix}. \tag{11}$$

Let Γ denote the largest eigenvalue of $\mathbf{H}\mathbf{H}^\top$, γ the smallest nonzero eigenvalue of $\mathbf{D} - \mathbf{H}\mathbf{H}^\top$, and $\mathbf{x}^\star$ the limiting point of WHADMM sequence $\{\mathbf{x}^k\}$. Based on the analysis of HADMM [16], we arrive at the following convergence result.

Theorem 1. *If local cost functions f_i are σ-strongly convex with L-Lipschitz continuous gradients, and the solution of* (2) *exists, then for $\beta \in (0,1)$ WHADMM iteration in* (3) *converges linearly to its optimal solution*

$$\|\mathbf{x}^{k+1} - \mathbf{x}^\star\|_G^2 \le c\left(\frac{1}{1+\delta}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_G^2 \tag{12}$$

where c, δ are both positive constants, and δ satisfies

$$\delta \le \min\left\{\frac{2\beta\sigma}{\rho(\Gamma + \frac{2\Gamma}{\gamma})}, \frac{(1-\beta)\rho\gamma}{L}\right\}. \tag{13}$$

Proof. Since $\mathbf{H}\mathbf{H}^\top \succeq \mathbf{0}$ and $\mathbf{D} - \mathbf{H}\mathbf{H}^\top$ is a valid Laplacian matrix, satisfying all properties of $\mathbf{C}\mathbf{E}\mathbf{C}^\top$ and $\mathbf{D} - \mathbf{C}\mathbf{E}\mathbf{C}^\top$, by replacing $\mathbf{C}\mathbf{E}\mathbf{C}^\top$ with $\mathbf{D} - \mathbf{H}\mathbf{H}^\top$, the proof in [16] continues to hold, which concludes the proof. □

Corollary 1. *Maximizing the constant δ with respect to β and ρ, the best convergence rate bound is*

$$\delta_{opt} = \frac{1}{\sqrt{\kappa_F\kappa_G(1 + 2\kappa_G)}} \tag{14}$$

where $\kappa_F := L/\sigma$ ($\kappa_G := \Lambda/\lambda$) denotes the condition number of the cost function (hypergraph).

Remark. *It should not be surprising that δ_{opt} depends on the cost function and the graph topology. This observation suggests it is possible to improve WHADMM by optimizing the graph condition number by tuning the edge weights.*

V. Optimal weights

Corollary 1 provides an explicit formula for the best rate bound that also suggests an approach to accelerate WHADMM by optimizing the parameters κ_F and κ_G. For a given problem however, cost functions and graph topology are prescribed; hence, it is only possible to tune the edge weights.

Optimizing the convergence rate boils down to maximizing δ_{opt}, which is an upper bound of the actual convergence speed. Maximizing δ_{opt} is equivalent to solving

$$\begin{aligned} \max_{\mathbf{D}_h,\mathbf{H}} \quad & \delta_{opt} \\ \text{s.t.} \quad & \mathbf{D}_h \in \mathcal{D},\ \mathbf{H} \in \mathcal{C} \\ & \mathbf{D}_h - \mathbf{H}\mathbf{H}^\top \succeq \mathbf{0},\ \mathbf{H}\mathbf{H}^\top \succeq \mathbf{0} \\ & \text{Null}(\mathbf{D}_h - \mathbf{H}\mathbf{H}^\top) = \text{span}\{\mathbf{1}\}. \end{aligned} \tag{15}$$

Problem (15) is challenging to solve because the objective is the ratio of the largest eigenvalue Γ over the smallest nonzero eigenvalue γ. Instead, we choose to maximize the smallest eigenvalue, while keeping the largest eigenvalue bounded to avoid scale ambiguity. Alternatively, one can also minimize Γ while keeping γ bounded. We use CVX to solve this problem. However, CVX does not accept quadratic terms as objective except for scalars, so $\gamma = \lambda_2(\mathbf{D}-\mathbf{H}\mathbf{H}^\top)$ cannot be optimized directly. To circumvent this obstacle, we relax the original problem and treat $\mathbf{W} = \mathbf{H}\mathbf{H}^\top$ as a new variable. Then problem (15) translates to

$$\begin{aligned} \max_{\mathbf{D}_w,\mathbf{W}} \quad & \lambda_2(\mathbf{D}_w - \mathbf{W}) \\ \text{s.t.} \quad & \mathbf{D}_w \in \mathcal{D}, \mathbf{W} \in \mathcal{W} \\ & \mathbf{W} \succeq \mathbf{0},\ \mathbf{D}_w - \mathbf{W} \succeq \mathbf{0} \\ & \text{Null}(\mathbf{D}_w - \mathbf{W}) = \text{span}\{\mathbf{1}\} \\ & \lambda_N(\mathbf{W}) \le q \end{aligned} \tag{16}$$

where q is some constant, and $\mathcal{W} \in \mathbb{R}^{N\times N}$ denotes the set of matrices with the same sparsity pattern as $\mathbf{C}\mathbf{C}^\top$. If we choose q such that $\Gamma \le \lambda_N(\mathbf{C}\mathbf{E}\mathbf{C}^\top)$ of HADMM while maximizing γ, then WHADMM is guaranteed to have a larger rate bound, which often leads to faster convergence in practice.

To find weights of the underlying $\mathscr{G}$, we need to recover $\mathbf{H}$ from $\mathbf{W}$. Since $\mathbf{H} \in \mathcal{C}$, we find that $\text{rank}(W) = \text{rank}(H) = \min\{N, M\}$. By the way of constructing $\mathscr{H}$, we have $M \le N$; thus, $\mathbf{W} \in \mathbb{R}^{N\times N}$ has the same rank as $\mathbf{H}$. We can take advantage of this low-rank structure to recover $\mathbf{H}$. In particular, upon obtaining $\mathbf{W}$, we can recover $\mathbf{H}$ by solving a nonnegative matrix factorization (NMF) problem

$$\begin{aligned} \min_{\mathbf{H}} \quad & \|\mathbf{H}\mathbf{H}^\top - \mathbf{W}\|_F^2 \\ \text{s.t.} \quad & \mathbf{H} \in \mathcal{C} \end{aligned} \tag{17}$$

where $\mathbf{W}$ is the solution of (16). Problem (17) can be solved by many NMF algorithms, but it turns out that the simple projected gradient descent suffices.

VI. Experiments

In this section, we compare the performance of WHADMM with existing algorithms, including HADMM, WDADMM, and DADMM. The problem we are dealing with is network averaging, which can be formulated using the least-squares cost $f(\mathbf{X}) = \|\mathbf{X} - \mathbf{O}\|_F^2/2$, where $\mathbf{O} := [\mathbf{o}_i^\top; \ldots; \mathbf{o}_N^\top] \in \mathbb{R}^{N\times p}$ collects nodal observations $\{\mathbf{o}_i\}$. The cost function is separable across nodes, that is, $f(\mathbf{X}) = \sum_i f_i(\mathbf{x}_i)$, where $f_i(\mathbf{x}_i) = \|\mathbf{x}_i - \mathbf{o}_i\|_2^2$. Each nodal observation $\mathbf{o}_i$ is generated with entries drawn from the uniform distribution $U[0, N]$. We test the algorithms on two graphs, one with two clusters and a line graph. We set $N = 31$ for both graphs. Though there are adaptive ways to tune the penalty parameter ρ, our focus in this experiment is to demonstrate convergence of four decentralized optimization algorithms, so we keep ρ fixed, and tune it optimally for each algorithm. We obtain edge weights by solving (16) and (17).

TABLE I
Optimal weights obtained by solving (16) and (17).

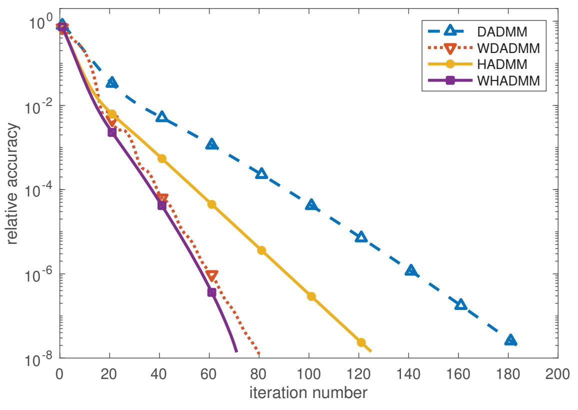
left cluster		right cluster		middle	
edge	weight	edge	weight	edge	weight
(1, 1)	0.4131	(6, 6)	0.4131	(5, 5)	0.4140
(1, 2)	0.4131	(6, 7)	0.4131	(5, 11)	0.6172
(1, 3)	0.4131	(6, 8)	0.4131	(10, 11)	0.6174
(1, 4)	0.4131	(6, 9)	0.4131		
(1, 5)	0.4969	(6, 10)	0.4968		

We measure the progress of each algorithm by the number of iterations needed to attain certain solutions with relative error $\|\mathbf{x} - \mathbf{x}^\star\|/\|\mathbf{x}^\star\|$. We plot relative accuracy against the number of iterations, as in Fig. 2. In particular, Fig. 1(a) shows that for the graph with two clusters, both WHADMM and WDADMM have similar performance; HADMM, while slower than the former two, still significantly outperforms DADMM. Fig. 12b shows that WHADMM exhibits similar convergence speed with HADMM, which implies that tuning weights does not help much. Also, in this case, WDADMM and DADMM perform almost the same, confirming the ineffectiveness of weight tuning for line graphs. These observations conform with those of [15], [16]. In both cases, we deduce that WHADMM consistently outperforms WDADMM and HADMM, although this margin may be small for line graphs.

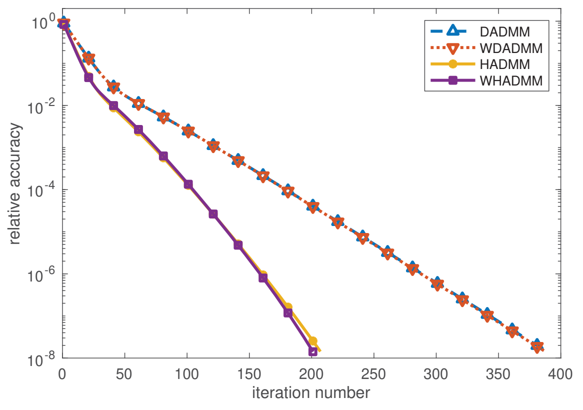
By inspecting the weights obtained by WHADMM, one can make some interesting observations. Consider the graph in Fig. 1 with two clusters connected by a bridge node. The weights obtained by solving (16) (17) are listed in Table I, which demonstrates that WHADMM is able to identify critical edges; thus, larger weights are assigned to edges (1, 5), (6, 10), (5, 11) and (10, 11). As a result, sitting at node 1, updates from node 5 are regarded as more important than those from nodes 1, 2, 3 and 4, which partially explains why WHADMM can outperform the alternatives.

VII. Conclusions and discussion

This paper introduced WHADMM that is able to deal with weighted graphs, and can leverage topology information to markedly improve performance. By maximizing the convergence rate, the optimal weights obtained admit an intuitive interpretation since critical edges are regarded as more important. Future research directions include investigation of dropping edges by assigning zero weights in order to reduce



(a) Convergence speed over a graph of two clusters.



(b) Convergence speed over a line graph.

Fig. 2. Performance comparison of four decentralized algorithms over different graphs.

communication overhead, and broadening the scope to optimization methods other than ADMM.

References

[1] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling Distributed Machine Learning with the Parameter Server." in *OSDI*, vol. 14, 2014, pp. 583–598.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[3] G. B. Giannakis, Q. Ling, G. Mateos, I. D. Schizas, and H. Zhu, "Decentralized Learning for Wireless Communications and Networking," in *Splitting Methods in Communication, Imaging, Science, and Engineering*, R. Glowinski, S. J. Osher, and W. Yin, Eds. Cham: Springer, 2016, pp. 461–497.

[4] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in Ad Hoc WSNs With Noisy Links – Part I: Distributed Estimation of Deterministic Signals," *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 350–364, Jan. 2008.

[5] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1989, vol. 23.

[6] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Mirection Method of Multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[7] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the Linear Convergence of the ADMM in Decentralized Consensus Optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, Apr. 2014.

[8] A. Nedic and A. Ozdaglar, "Distributed Subgradient Methods for Multi-Agent Optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[9] D. Jakovetić, J. Xavier, and J. M. F. Moura, "Fast Distributed Gradient Methods," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1131–1146, May 2014.

[10] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 592–606, Mar. 2012.

[11] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip Algorithms for Distributed Signal Processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, Nov. 2010.

[12] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," *SIAM Journal on Imaging Sciences*, vol. 7, no. 3, pp. 1588–1623, 2014.

[13] W. Deng and W. Yin, "On the Global and Linear Convergence of the Generalized Alternating Direction Method of Multipliers," *J Sci Comput*, vol. 66, no. 3, pp. 889–916, May 2015.

[14] M. Hong and Z.-Q. Luo, "On the linear convergence of the alternating direction method of multipliers," *Math. Program.*, vol. 162, no. 1-2, pp. 165–199, Mar. 2017.

[15] Q. Ling, Y. Liu, W. Shi, and Z. Tian, "Weighted ADMM for Fast Decentralized Network Optimization," *IEEE Transactions on Signal Processing*, vol. 64, no. 22, pp. 5930–5942, Nov. 2016.

[16] M. Ma, A. N. Nikolakopoulos, and G. B. Giannakis, "Hybrid ADMM: A Unifying and Fast Approach to Decentralized Optimization," *EURASIP Journal on Advances in Signal Processing*, December 2018.

[17] M. Bolla, "Spectra, Euclidean representations and clusterings of hypergraphs," *Discrete Mathematics*, vol. 117, no. 1-3, pp. 19–39, Jul. 1993.

[18] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Computers & Mathematics with Applications*, vol. 2, no. 1, pp. 17–40, Jan. 1976.

[19] P. Lions and B. Mercier, "Splitting Algorithms for the Sum of Two Nonlinear Operators," *SIAM J. Numer. Anal.*, vol. 16, no. 6, pp. 964–979, Dec. 1979.