A PENALTY ALTERNATING DIRECTION METHOD OF MULTIPLIERS FOR DECENTRALIZED COMPOSITE OPTIMIZATION

*Jiaojiao Zhang** *Anthony Man-Cho So** *Qing Ling*†

*Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong

†School of Data and Computer Science and Guangdong Province Key Laboratory of Computational Science, Sun Yat-Sen University

ABSTRACT

This paper proposes a penalty alternating direction method of multipliers (ADMM) to minimize the summation of convex composite functions over a decentralized network. Each agent in the network holds a private function consisting of a smooth part and a nonsmooth part, and can only exchange information with its neighbors during the optimization process. We consider a penalized approximation of the decentralized optimization problem; but unlike the existing penalty methods, here the penalty parameter can be very small such that the approximation error is negligible. On the other hand, the small penalty parameter makes the penalized objective ill-conditioned, such that the popular proximal gradient descent method has to use a small step size, and is hence slow. To address this issue, we propose to solve the penalized formulation with ADMM. We further utilize the composite structures of the private functions through linearizing the smooth parts so as to reduce computational costs, and handling the nonsmooth parts with proximal operators. The proposed penalty ADMM (abbreviated as PAD) is provably convergent when the private functions are convex, and linearly convergent when the smooth parts are further strongly convex. Numerical experiments corroborate the theoretical analyses, and demonstrate the advantages of PAD over existing state-of-the-art algorithms, such as DL-ADMM, PG-EXTRA and NIDS.

***Index Terms*—** decentralized optimization, alternating direction method of multipliers (ADMM), composite optimization

1. INTRODUCTION

This paper focuses on the following decentralized composite optimization problem defined over an undirected and connected network with n agents, in the form of

$$\min_{x\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n}\big(f_i(x)+g_i(x)\big), \tag{1}$$

where $f_i:\mathbb{R}^p\to\mathbb{R}\cup\{+\infty\}$ and $g_i:\mathbb{R}^p\to\mathbb{R}\cup\{+\infty\}$ are two convex functions privately owned by agent i. We assume that f_i is convex and smooth, while g_i is convex, nonsmooth and its proximal mapping is easy to solve. Every agent aims to obtain an optimal solution $\hat{x}^*$ of (1) via local computation and communication with its neighbors. Such decentralized composite optimization problems appear in various fields, including network optimization [1,2], optimization-based cooperative control [3], machine learning [4, 5], to name a few.

Decentralized optimization methods have been widely studied in the literature. They usually operate in either primal or dual domain. For the primal domain methods, distributed gradient descent (DGD) and subsequent extensions are studied in [6–9]. With a fixed step size, DGD converges fast but to a neighborhood of an optimal solution, and the size of the neighborhood is proportional to the step size [6, 7]. Similar local convergence result also holds for the nonconvex case [8]. With a diminishing step size, DGD is able to converge to an optimal solution, but the speed is slow [9]. The inaccuracy is due to the fact that DGD is essentially a gradient descent method to solve a penalized approximation of (1), where the step size determines the penalty parameter, and hence the approximation error [7]. Second-order methods are also applicable to solving the penalized approximation, with the same accuracy-speed tradeoff [10, 11].

The dual domain methods include decentralized alternating direction method of multipliers (ADMM) [12–15], decentralized augmented Lagrangian-based algorithms [16], dual accelerated schemes [17], and so on. All of these algorithms rewrite (1) to an equivalent form with consensus constraints, and solve it in the dual domain. The use of fixed step sizes enables them to have fast and exact convergence. When the local functions are smooth, some of these algorithms are proven to linearly converge to an exact optimal solution [13–17]. When the local functions are nonsmooth and have a composite form as in (1), proximal decentralized linearized ADMM (DL-ADMM) is proposed in [18] with convergence guarantee. Its ergodic convergence rate $O(\frac{1}{k})$, where k is the number of iterations, is established in [19].

There also exist some other decentralized methods that do not explicitly operate in the dual domain, but are still able to converge to an exact optimal solution with fixed step sizes [20–25]. To be specific, the EXTRA algorithm in [20] considers the case that the local functions only have the smooth parts. When the local functions are composite, [21] proposes PG-EXTRA and [22] proposes NIDS, which converge to an optimal solution at the rates of $O(\frac{1}{k})$ and $o(\frac{1}{k})$, respectively. A recent work [23] establishes a linear convergence rate, but assuming that the smooth parts are common across all the agents. For decentralized nonconvex composite and constrained optimization problems, [24] and [25] develop gradient-tracking methods with local convergence guarantee.

In this paper, we also consider solving a penalized approximation of (1) like the primal domain methods. However, our approach differs from them since we allow the penalty parameter to be very small such that the approximation error is negligible. Since the small penalty parameter makes the penalized objective ill-conditioned, the popular proximal gradient descent method has to use a small step size, and is hence slow. To address this issue, we propose to solve the penalized formulation with ADMM. We further utilize the composite structures of the private functions through linearizing the smooth parts so as to reduce computational costs, and handling the nonsmooth parts with proximal operators. The proposed penalty ADMM (abbreviated as PAD) is provably convergent when the private functions are convex, and linearly convergent when the smooth

parts are further strongly convex. We also demonstrate the connection between PAD and the dual domain method EXTRA. When the nonsmooth parts of the local functions are absent, EXTRA is shown to be a special case of PAD. Numerical experiments corroborate the theoretical analyses, and demonstrate the advantages of PAD over several existing state-of-the-art algorithms, such as DL-ADMM, PG-EXTRA and NIDS.

Notations. Consider a bidirectionally connected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} \triangleq \{1, \ldots, n\}$ denotes the set of agents (nodes), and $\mathcal{E}$ denotes the set of directed communication links (edges). Let $\mathcal{N}_i \triangleq \{j \in \mathcal{N} : (i,j) \in \mathcal{E} \text{ and } (j,i) \in \mathcal{E}\}$ denote the set of neighboring agents of $i \in \mathcal{N}$. $\mathbf{1}_{q\times r} \in \mathbb{R}^{q\times r}$ represents the matrix with all ones, and $I \in \mathbb{R}^{n\times n}$ is the identity matrix. $\lambda_{\min}(\cdot)$ and $\lambda_{\max}(\cdot)$ denote the minimum and maximum eigenvalues of a square matrix, respectively.

2. ALGORITHM DEVELOPMENT

In the algorithm development and convergence analysis, we make the following assumptions on the smooth and nonsmooth parts of the local functions.

Assumption 1. *Each g_i is convex and nonsmooth. The proximal mapping of g_i is given as*

$$\text{prox}_{cg_i}(x) \triangleq \text{argmin}_y\{g_i(y) + \frac{1}{2c}\|y - x\|^2\},$$

where $c > 0$ is a scalar. We assume the proximal mapping can be computed easily.

Assumption 2. *Each $f_i(x)$ is convex and differentiable with a Lipschitz continuous gradient such that*

$$\|\nabla f_i(x) - \nabla f_i(y)\| \le L_f\|x - y\|,\ \forall x, y \in \mathbb{R}^p,$$

where $L_f > 0$ is the Lipschitz constant.

The proposed algorithm is going to utilize the composite structure of the local functions. We will show its convergence under Assumptions 1 and 2. To show its linear convergence, we also need another assumption.

Assumption 3. *Each $f_i(x)$ is strongly convex such that*

$$\big\langle x - y, \nabla f_i(x) - \nabla f_i(y)\big\rangle \ge \mu_f\|x - y\|^2,\ \forall x, y \in \mathbb{R}^p,$$

where $\mu_f > 0$ is the strong convexity constant.

We begin with a consensus-constrained reformulation of (1). Let each agent i hold a local variable $x_i \in \mathbb{R}^p$. Define $\mathbf{x} \in \mathbb{R}^{n\times p}$ whose i-th row is x_i^T. We say that $\mathbf{x}$ is consensual if all the rows are identical, i.e., $x_1^T = \cdots = x_n^T$. To characterize this consensus property, we introduce weight w_{ij} between agents i and j, and collect all the weights in a mixing matrix $W = [w_{ij}] \in \mathbb{R}^{n\times n}$. We make the following assumption on W, which is standard in decentralized optimization.

Assumption 4. *(Mixing matrix): W is symmetric and doubly stochastic, i.e., $W = W^T$ and $W\mathbf{1}_{n\times 1} = \mathbf{1}_{n\times 1}$. The null space of $I - W$ is* span $(\mathbf{1}_{n\times 1})$. *Further, $i \ne j$ and $(i,j) \notin \mathcal{E}$, then $w_{ij} = 0$; otherwise, $w_{ij} > 0$.*

According to the Perron-Frobenius theorem [26], Assumption 4 implies that the eigenvalues of W lie in $(-1, 1]$ and the multiplicity of eigenvalue 1 is one. Because the null space of $I - W$ is span $(\mathbf{1}_{n\times 1})$, so is its square root $(I - W)^{\frac{1}{2}}$. Therefore, $(I - W)^{\frac{1}{2}}\mathbf{x} = 0$ if and only if $x_1^T = \cdots = x_n^T$. The mixing matrix W satisfying Assumption 4 can be generated by the ways introduced in [27], when the underlying network is connected. Also define functions $f(\mathbf{x}) \triangleq \sum_{i=1}^n f_i(x_i)$ and $g(\mathbf{x}) \triangleq \sum_{i=1}^n g_i(x_i)$. Then, (1) is equivalent to the following constrained problem

$$\hat{\mathbf{x}}^* = \arg\min_{\mathbf{x}}\ f(\mathbf{x}) + g(\mathbf{x}), \quad \text{s.t. } (I - W)^{\frac{1}{2}}\mathbf{x} = 0. \tag{2}$$

Penalized Approximation Formulation. Instead of solving the constrained problem (2), we consider its penalized approximation in the form of

$$\mathbf{x}^* = \arg\min_{\mathbf{x}}\ f(\mathbf{x}) + g(\mathbf{x}) + \frac{1}{2\epsilon}\|(I - W)^{\frac{1}{2}}\mathbf{x}\|_{\mathcal{F}}^2, \tag{3}$$

where $\|\cdot\|_{\mathcal{F}}$ denotes the Frobenius norm and $\epsilon > 0$ is the penalty parameter. The approximation error, i.e., the gap between $\hat{\mathbf{x}}^*$ and $\mathbf{x}^*$, is controlled by ϵ. When ϵ is sufficiently small, the approximation error is negligible.

When $g(\mathbf{x}) = 0$, (3) can be solved with gradient descent

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \gamma\big(\nabla f(\mathbf{x}^k) + \frac{1}{\epsilon}(I - W)\mathbf{x}^k\big), \tag{4}$$

where $\gamma > 0$ is the step size. However, when ϵ is very small, the Lipschitz constant of the gradient $\nabla f(\mathbf{x}) + \frac{1}{\epsilon}(I - W)\mathbf{x}$ is in the order of $O(\frac{1}{\epsilon})$, which suggests that γ must be in the order of $O(\epsilon)$ to guarantee convergence [28]. In fact, setting $\gamma = \epsilon$ recovers the DGD update $\mathbf{x}^{k+1} = W\mathbf{x}^k - \epsilon\nabla f(\mathbf{x}^k)$ [6, 7]. However, such a small step size makes DGD converges slowly, although to a very small neighborhood of $\hat{\mathbf{x}}^*$. This dilemma demonstrates the unfavorable accuracy-speed tradeoff of DGD. The same statement holds true when we apply the proximal gradient method to handle the case of $g(\mathbf{x}) \ne 0$ [22].

Algorithm Development. To address this issue, we propose an ADMM-based algorithm to solve (3) – note that ADMM is a popular dual domain method for decentralized optimization, but here we apply it to the penalized approximation formulation in the primal domain. By introducing an auxiliary variable $\mathbf{z} \in \mathbb{R}^{n\times p}$, (3) is equivalent to

$$\begin{aligned}(\mathbf{x}^*, \mathbf{z}^*) = \arg\min_{\mathbf{x},\mathbf{z}}\ & f(\mathbf{x}) + g(\mathbf{x}) + \frac{1}{2\epsilon}\|\mathbf{z}\|_{\mathcal{F}}^2,\\ & \text{s.t. } (I - W)^{\frac{1}{2}}\mathbf{x} = \mathbf{z}.\end{aligned} \tag{5}$$

The augmented Lagrangian function of (5) is $L_\alpha(\mathbf{x}, \mathbf{z}, \Pi) \triangleq f(\mathbf{x}) + g(\mathbf{x}) + \frac{1}{2\epsilon}\|\mathbf{z}\|_{\mathcal{F}}^2 + \big\langle\Pi, (I - W)^{\frac{1}{2}}\mathbf{x} - \mathbf{z}\big\rangle + \frac{\alpha}{2}\|(I - W)^{\frac{1}{2}}\mathbf{x} - \mathbf{z}\|_{\mathcal{F}}^2$, where $\Pi \in \mathbb{R}^{n\times p}$ is the Lagrange multiplier and $\alpha > 0$ is the penalty parameter. ADMM minimizes the augmented Lagrangian function over the primal variables $\mathbf{x}$ and $\mathbf{z}$ in an alternating direction manner, followed by updates the dual variable Π through dual gradient ascent. However, on account of the nonsmooth term $\mathbf{g}(\mathbf{x})$ and the coefficient $(I - W)^{\frac{1}{2}}$ in the quadratic term in $L_\alpha(\mathbf{x}, \mathbf{z}, \Pi)$, the subproblem w.r.t. $\mathbf{x}$ does not have a closed-form solution. Thus, we utilize the composite structure to inexactly update $\mathbf{x}$ through linearizing the smooth part and calculating the proximal mapping of the nonsmooth part, as follows.

Update of $\mathbf{x}$. We separate $L_\alpha(\mathbf{x}, \mathbf{z}^k, \Pi^k)$ as a smooth part $\tilde{L}_\alpha(\mathbf{x}, \mathbf{z}^k, \Pi^k) \triangleq f(\mathbf{x}) + \frac{1}{2\epsilon}\|\mathbf{z}^k\|_{\mathcal{F}}^2 + \big\langle\Pi^k, (I - W)^{\frac{1}{2}}\mathbf{x}\big\rangle + \frac{\alpha}{2}\|(I - W)^{\frac{1}{2}}\mathbf{x} - \mathbf{z}^k\|_{\mathcal{F}}^2$ plus a nonsmooth part $g(\mathbf{x})$. Replace the smooth part $\tilde{L}_\alpha(\mathbf{x}, \mathbf{z}^k, \Pi^k)$ by a quadratic approximation $\tilde{L}_\alpha(\mathbf{x}, \mathbf{z}^k, \Pi^k) \approx \tilde{L}_\alpha(\mathbf{x}^k, \mathbf{z}^k, \Pi^k) + \big\langle\nabla_{\mathbf{x}}\tilde{L}_\alpha(\mathbf{x}^k, \mathbf{z}^k, \Pi^k), \mathbf{x} - \mathbf{x}^k\big\rangle + \frac{1}{2c}\|\mathbf{x} - \mathbf{x}^k\|_{\mathcal{F}}^2$

Algorithm 1 PAD run by agent i

Require: Choose the parameters ϵ, α and c. Initialize the local variables to x_i^0, ${\bar{z}_i}^0$ and $\bar{\pi}_i^0$.

1: **for** $k = 1, 2, \cdots$ **do**

2: Update local variable x_i^{k+1} by

$$x_i^{k+1} = \text{prox}_{cg_i}\Big(x_i^k - c\big[\nabla f_i(x_i^k) + \alpha(x_i^k - \sum_{j\in\mathcal{N}_i} w_{ij}x_j^k - \bar{z}_i^k) + \bar{\pi}_i^k\big]\Big).$$

3: Transmit x_i^{k+1}/ receive x_j^{k+1} from neighbors $j \in \mathcal{N}_i$.

4: Update local auxiliary variable $\bar{z}_i^{k+1}$ by

$$\bar{z}_i^{k+1} = \frac{1}{\alpha + \frac{1}{\epsilon}}\big[\bar{\pi}_i^k + \alpha(x_i^{k+1} - \sum_{j\in\mathcal{N}_i} w_{ij}x_j^{k+1})\big].$$

5: Update local dual variable $\bar{\pi}_i^{k+1}$ by

$$\bar{\pi}_i^{k+1} = \bar{\pi}_i^k + \alpha\big[(x_i^{k+1} - \sum_{j\in\mathcal{N}_i} w_{ij}x_j^{k+1}) - \bar{z}_i^{k+1}\big].$$

6: **end for**

centered at $\mathbf{x}^k$, where $c > 0$ is a parameter and $\nabla_{\mathbf{x}}\tilde{L}_\alpha(\mathbf{x}, \mathbf{z}^k, \Pi^k)$ is the gradient of $\tilde{L}_\alpha(\mathbf{x}, \mathbf{z}^k, \Pi^k)$ w.r.t. $\mathbf{x}$. Using this approximation in $\min_{\mathbf{x}} L_\alpha(\mathbf{x}, \mathbf{z}^k, \Pi^k)$ leads to the primal update $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} g(\mathbf{x}) + \tilde{L}_\alpha(\mathbf{x}^k, \mathbf{z}^k, \Pi^k) + \langle \nabla_{\mathbf{x}}\tilde{L}_\alpha(\mathbf{x}^k, \mathbf{z}^k, \Pi^k), \mathbf{x} - \mathbf{x}^k\rangle + \frac{1}{2c}\|\mathbf{x} - \mathbf{x}^k\|_{\mathcal{F}}^2$, which is the proximal mapping of $g(\mathbf{x})$ and has a closed-form solution

$$\mathbf{x}^{k+1} = \text{prox}_{cg}\Big(\mathbf{x}^k - c\big[\nabla f(\mathbf{x}^k) + \alpha(I-W)^{\frac{1}{2}}((I-W)^{\frac{1}{2}}\mathbf{x}^k - \mathbf{z}^k + \frac{\Pi^k}{\alpha})\big]\Big). \tag{6}$$

Update of $\mathbf{z}$. Update $\mathbf{z}^{k+1}$ from $\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \frac{1}{2\epsilon}\|\mathbf{z}\|_{\mathcal{F}}^2 + \big\langle \Pi^k, (I-W)^{\frac{1}{2}}\mathbf{x}^{k+1} - \mathbf{z}\big\rangle + \frac{\alpha}{2}\|(I-W)^{\frac{1}{2}}\mathbf{x}^{k+1} - \mathbf{z}\|_{\mathcal{F}}^2$. It is equivalent to

$$\mathbf{z}^{k+1} = \frac{1}{\alpha + \frac{1}{\epsilon}}\big[\Pi^k + \alpha(I-W)^{\frac{1}{2}}\mathbf{x}^{k+1}\big]. \tag{7}$$

Update of Π. Finally, the update of Π^{k+1} is given by

$$\Pi^{k+1} = \Pi^k + \alpha\big[(I-W)^{\frac{1}{2}}\mathbf{x}^{k+1} - \mathbf{z}^{k+1}\big]. \tag{8}$$

The updates in (6), (7) and (8) can be simplified by introducing $\bar{\Pi}^k \triangleq (I-W)^{\frac{1}{2}}\Pi^k$ and $\bar{\mathbf{z}}^k \triangleq (I-W)^{\frac{1}{2}}\mathbf{z}^k$. With these notations and splitting the updates to the agents, we outline the proposed penalty ADMM (PAD) in Algorithm 1.

Connection between PAD and EXTRA. Although PAD solves the penalized approximation formulation (3) (and (5) equivalently) that is often considered in designing the primal domain methods, below we show that the dual domain method EXTRA [20] is a special case of PAD when $g(\mathbf{x}) = 0$. It will be verified that PAD recovers EXTRA after eliminating $\bar{\mathbf{z}}$ and $\bar{\Pi}$ and choosing suitable parameters.

Combining (8) and (7) to eliminate $(I-W)^{\frac{1}{2}}\mathbf{x}^{k+1}$, we get $\frac{1}{\epsilon}\mathbf{z}^{k+1} = \Pi^{k+1}$. When the variables are also initialized as $\frac{1}{\epsilon}\mathbf{z}^0 = \Pi^0$, we have $\frac{1}{\epsilon}\mathbf{z}^k = \Pi^k$ and consequently, $\frac{1}{\epsilon}\bar{\mathbf{z}}^k = \bar{\Pi}^k$. When $g(\mathbf{x}) = 0$, substituting $\frac{1}{\epsilon}\bar{\mathbf{z}}^k = \bar{\Pi}^k$ into (6) yields

$$\mathbf{x}^{k+1} = \mathbf{x}^k - c\big[\nabla f(\mathbf{x}^k) + \alpha(I-W)\mathbf{x}^k + (1-\alpha\epsilon)\bar{\Pi}^k\big], \tag{9}$$

and

$$\mathbf{x}^{k+2} = \mathbf{x}^{k+1} - c\big[\nabla f(\mathbf{x}^{k+1}) + \alpha(I-W)\mathbf{x}^{k+1} + (1-\alpha\epsilon)\bar{\Pi}^{k+1}\big]. \tag{10}$$

Subtracting (10) by (9) and eliminating $\bar{\Pi}^{k+1} - \bar{\Pi}^k$ by (8), we obtain

$$\mathbf{x}^{k+2} = \frac{1}{1+\alpha\epsilon}\Big\{\big((2+\alpha\epsilon - 2c\alpha)I + 2c\alpha W\big)\mathbf{x}^{k+1} - \big[I - c\alpha(I-W)\big]\mathbf{x}^k - c\big[(I+\alpha\epsilon)\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k)\big]\Big\}. \tag{11}$$

Note that the update of EXTRA is

$$\mathbf{x}^{k+2} = (I+W)\mathbf{x}^{k+1} - \tilde{W}\mathbf{x}^k - c\big[\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k)\big],$$

where $\tilde{W} = \frac{I+W}{2}$ and c is the step size. Comparing it with (11), we observe that PAD recovers EXTRA if $c\alpha = \frac{1}{2}$ and $\epsilon = 0$. Actually, ϵ can never be zero; it tunes the accuracy-speed tradeoff. In addition, $c\alpha$ can be set to other values than $\frac{1}{2}$. These flexibilities enable PAD to converge faster than EXTRA, as we will illustrate in the numerical experiments.

3. CONVERGENCE AND RATE OF CONVERGENCE

This section analyzes convergence and rate of convergence for PAD. Instead of considering the simplified sequence $\{\bar{\mathbf{z}}^k\}$ and $\{\bar{\Pi}^k\}$, we analyze the sequences $\{\mathbf{z}^k\}$ and $\{\Pi^k\}$ generated by (6), (7) and (8), because they are equivalent and the sequence $\{\mathbf{x}^k\}$ is identical. The proofs are left in a longer version of this paper.

Convergence Analysis under General Convexity. Define a triple $\mathbf{u}^k \triangleq (\mathbf{x}^k, \mathbf{z}^k, \Pi^k)$ and $\mathbf{u}^* \triangleq (\mathbf{x}^*, \mathbf{z}^*, \Pi^*)$ where $\mathbf{x}^*$, $\mathbf{z}^*$ and Π^* are the optimal primal and dual solutions of (5). To characterize the convergence, define a matrix $Q \triangleq I - \alpha c(I-W)$ and a triple $P \triangleq (\frac{1}{2c}Q, \frac{\alpha}{2}I, \frac{1}{2\alpha}I)$ of matrices. Define $\|\mathbf{u}^k - \mathbf{u}^*\|_P^2 = \|\mathbf{x}^k - \mathbf{x}^*\|_{\frac{1}{2c}Q}^2 + \|\mathbf{z}^k - \mathbf{z}^*\|_{\frac{\alpha}{2}I}^2 + \|\Pi^k - \Pi^*\|_{\frac{1}{2\alpha}I}^2$ as the squared distance from $\mathbf{u}^k$ to $\mathbf{u}^*$. In the following lemma, we show that $\|\mathbf{u}^k - \mathbf{u}^*\|_P^2$ decreases sufficiently fast when the parameters are properly chosen.

Lemma 1. *Under Assumptions 1, 2 and 4, if the parameters α and c are chosen such that $\frac{1}{2c}\big[I - \alpha c(I-W)\big] - \frac{L_f}{2}I \succ 0$, then it holds for all $k \geq 1$ that $\|\mathbf{u}^k - \mathbf{u}^*\|_P^2 - \|\mathbf{u}^{k+1} - \mathbf{u}^*\|_P^2 \geq \beta\|\mathbf{u}^k - \mathbf{u}^{k+1}\|_P^2$, where $\beta > 0$ is some positive constant.*

With Lemma 1, $\{\|\mathbf{u}^k - \mathbf{u}^*\|_P^2\}$ converges to 0. Because $Q \succ 0$ given that $\frac{1}{2c}\big[I - \alpha c(I-W)\big] - \frac{L_f}{2}I \succ 0$, we conclude that $\{\mathbf{u}^k\}$ converges to $\mathbf{u}^*$, stated as follows.

Theorem 1. *Under Assumptions 1, 2 and 4, if the parameters are chosen as in Lemma 1, then the sequence $\{\mathbf{u}^k\}$ generated by* (6)*,* (7) *and* (8) *from any initial point $\mathbf{u}^0$ converges to the optimal solution $\mathbf{u}^*$ of the penalized problem* (5)*.*

Theorem 1 shows the convergence of PAD to the optimal solution of the penalized problem (5) (and (3) equivalently). Note that it holds true for any ϵ. Therefore, we can choose a sufficiently small ϵ so that the approximation error between (3) and (2) is negligible. Theorem 1 also provides guidelines for setting the parameters c and α. The condition $\frac{1}{2c}\big[I - \alpha c(I-W)\big] - \frac{L_f}{2}I \succ 0$ implies that $\frac{1}{c} > \alpha\lambda_{\max}(I-W) + L_f$, where $\lambda_{\max}(I-W)$ is the largest eigenvalue of $I - W$.

Linear Convergence Rate under Strongly Convexity. The following theorem shows that PAD is linearly convergent under the assumption of strong convexity.

Theorem 2. *Under Assumption 1–4, for any constant $\delta \triangleq 1 - \frac{L_f^2 c}{\lambda_{\min}(Q)\mu_f} > 0$, where $\lambda_{\min}(Q)$ is the smallest eigenvalue of Q, the sequence $\{\mathbf{x}^k, \mathbf{z}^k\}$ generated by* (6)*,* (7) *and* (8) *converges to $(\mathbf{x}^*, \mathbf{z}^*)$, as $\|\mathbf{x}^* - \mathbf{x}^{k+1}\|_Q^2 + \|\mathbf{z}^* - \mathbf{z}^{k+1}\|_{\mathcal{F}}^2 \leq \eta(\|\mathbf{x}^* - \mathbf{x}^k\|_Q^2 + \|\mathbf{z}^* - \mathbf{z}^k\|_{\mathcal{F}}^2)$, in which $\eta \triangleq \max\{\frac{1}{c_2 c + 1}, \frac{\alpha^2\epsilon^2 + 1}{\alpha\epsilon + \alpha^2\epsilon^2 + 1}\} < 1$ and $c_2 \triangleq \frac{\mu_f\delta}{(1+\delta)\lambda_{\max}(Q)}$ are two constants.*

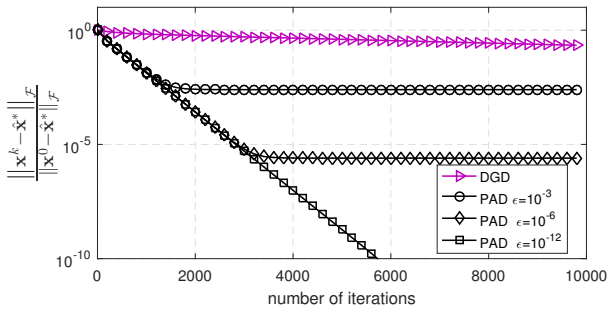

Fig. 1: Relative error of DGD (with $\epsilon = 10^{-3}$) and PAD (with $\epsilon = 10^{-3}, \epsilon = 10^{-6}$ and $\epsilon = 10^{-12}$, respectively) in the decentralized logistic regression problem. For DGD, its step size is $\gamma = \epsilon$. For PAD, $\alpha = 0.6$ and $c = 0.032$ according to the condition in Theorem 1.

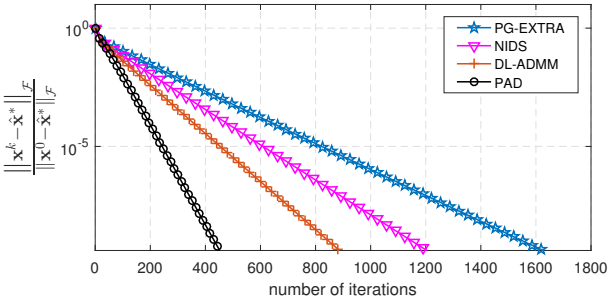

Fig. 2: Relative error of PG-EXTRA, NIDS, DL-ADMM and PAD in the decentralized quadratic programming problem. For PAD, $\epsilon = 10^{-12}$, $\alpha = 1.2$ and $c = 0.2$ as suggested in Theorem 2. For PG-EXTRA, the step size $\alpha = \frac{2\lambda_{\min}(\tilde{W})}{L_f}$ and $\tilde{W} = \frac{I+W}{2}$ as suggested in [21]. For NIDS, the step size is $\alpha = \frac{1.9}{L_f}$ as suggested in [23]. The parameters of DL-ADMM are hand-optimized.

4. NUMERICAL EXPERIMENTS

We conduct numerical experiments over a randomly generated connected network with n agents and $\frac{\tau n(n-1)}{2}$ undirected edges, where $\tau \in (0, 1]$ is the connectivity ratio. We compare PAD with several state-of-the-art algorithms, DGD [6], DL-ADMM [18], PG-EXTRA [21] and NIDS [23]. The mixing matrix W is generated with the Metropolis-Hastings rule. Performance is evaluated by the relative error $\|\mathbf{x}^k - \hat{\mathbf{x}}^*\|_{\mathcal{F}} / \|\mathbf{x}^0 - \hat{\mathbf{x}}^*\|_{\mathcal{F}}$, where $\hat{\mathbf{x}}^*$ is the optimal solution of (2) calculated by CVX.

Decentralized Logistic Regression. Each agent i holds local training data $\left(M_{(i)j}, y_{(i)j}\right) \in \mathbb{R}^p \times \{-1, +1\}$, $j = 1, \cdots, m_i$ where $M_{(i)j}$ is j-th feature data of agent i and $y_{(i)j}$ is the corresponding binary label. Entries of each $M_{(i)j}$ (except for the last one that is set as 1) are generated according to the standard normal distribution, followed by changing its magnitude such that $\lambda_{\max}(M_{(i)}^T M_{(i)}) = 30$. Each label $y_{(i)j}$ is generated according to the uniform distribution. The problem is $\min_x \frac{1}{n}\sum_{i=1}^n \left\{\sum_{j=1}^{m_i} \ln\left(1 + \exp(-(M_{(i)j}x)y_{(i)j})\right)\right\}$. We set $n = 30$, $\tau = 0.5$, $p = 10$ and $m_i = 5, \forall i$. As shown in Fig. 1, PAD and DGD with a fixed step size all converge to a neighborhood of $\hat{\mathbf{x}}^*$, whose size is proportional to ϵ. However, since ϵ is small, the step size of DGD must be in the same order and the convergence speed is very slow. For PAD, it converges with fast speed under different ϵ. The smaller ϵ brings higher accuracy.

Decentralized Quadratic Programming. Each agent i has a local quadratic function $f_i(x) = \frac{1}{2}x^T Q_i x + h_i^T x$ and a local linear constraint $a_i^T x \leq b_i$, where $Q_i \in \mathbb{R}^{p\times p} \succ 0$, $h_i \in \mathbb{R}^p, a_i \in \mathbb{R}^p$, and $b_i \in \mathbb{R}$. The nonsmooth function is defined as $g_i(x) = 0$ if $a_i^T x \leq b_i$, and $g_i(x) = +\infty$ otherwise. We set $n = 10$, $\tau = 0.4$ and $p = 50$. The matrices $\{Q_i\}$ are generated such that the Lipschitz gradient and the strongly convex constants of each f_i are $L_f = 1$ and $\mu_f = 0.5$, respectively.

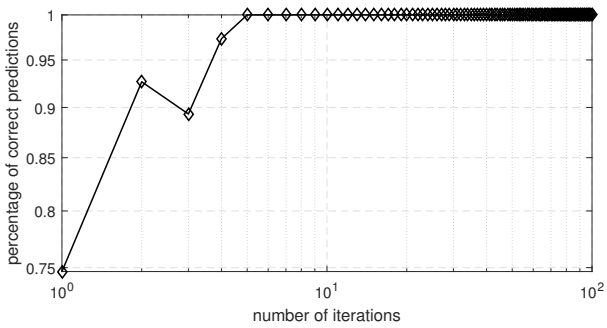

Fig. 3: Percentage of correct predictions of PAD in the classification for breast cancer data. We set $\epsilon = 10^{-12}$, $\alpha = 0.2$ and $c = 0.9$ by hand-optimization.

Fig. 2 shows that PAD needs less than 450 iterations to attain 10^{-9} relative error, while the others need at least two times of iterations to attain the same accuracy. PAD converges linearly, which is consistent with Theorem 2.

Application in Classification for Breast Cancer Data. There are 683 samples among which 458 are benign and 341 are malignant [29]. Each sample features 9 pieces of cell information and a binary outcome. The outcomes are transfered to be -1 (benign) or $+1$ (malignant) and all the features are min-max normalized. We randomly choose 650 samples, among which 500 are training data and 150 are testing data. Set $n = 50$ and $\tau = 0.5$. Each node i evenly holds $m_i = 10$ training samples $(M_{(i)j}, y_{(i)j}) \in \mathbb{R}^p \times \{-1, +1\}, j = 1, \cdots, m_i$. The last entry of $M_{(i)j}$ is set as 1 so that $p = 10$. The decentralized logistic regression with ℓ_1 regularization is $\min_x \frac{1}{n}\sum_{i=1}^n \sum_{j=1}^{m_i} \ln\left(1 + \exp\left(-\left(M_{(i)j}x\right) y_{(i)j}\right)\right) + \frac{1}{n}\sum_{i=1}^n \lambda_i \|x\|_1$, where $\lambda_i = \frac{0.1}{n}$ is the regularization parameter. For the testing, each agent i randomly holds 3 testing samples and uses its own x_i^k to predict. At each iteration, the total number of correct predictions from all 50 nodes are divided by 150 to compute the percentage of correct predictions. Fig. 3 shows that PAD achieves 100% correct predictions within 10 iterations.

5. CONCLUSION

In this paper, we proposed a consensus-based decentralized algorithm called PAD to solve the penalized approximation of the decentralized composite problem. Unlike the existing penalized methods, the proposed PAD is fast even with a very small penalty coefficient so that the approximation error caused by penalization is negligible. The proposed PAD is provably convergent when the private functions are convex, and linearly convergent when the smooth parts are further strongly convex. Numerical experiments demonstrate the advantages of PAD over existing state-of-the-art algorithms, such as PG-EXTRA and NIDS.

Acknowledgement. A. M.-C. So is supported in part by the CUHK Research Sustainability of Major RGC Funding Schemes Project 3133236. Qing Ling is supported in part by NSF China Grants 61573331 and 61973324, and Fundamental Research Funds for the Central Universities.

6. REFERENCES

[1] S. Pu, W. Shi, J. Xu, and A. Nedić, "A push-pull gradient method for distributed optimization in networks," in *IEEE Conference on Decision and Control*, 2018, pp. 3385–3390.

[2] K. Scaman, F. Bach, S. Bubeck, L. Massoulié, and Y. T. Lee, "Optimal algorithms for non-smooth distributed optimization in networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 2740–2749.

[3] R. Dutta, L. Sun, and D. Pack, "A decentralized formation and network connectivity tracking controller for multiple unmanned systems," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 6, pp. 2206–2213, 2017.

[4] C. Zhang, P. Zhao, S. Hao, Y. C. Soh, B. S. Lee, C. Miao, and S. C. Hoi, "Distributed multi-task classification: a decentralized online learning approach," *Machine Learning*, vol. 107, no. 4, pp. 727–747, 2018.

[5] A. Koppel, S. Paternain, C. Richard, and A. Ribeiro, "Decentralized online learning with kernels," *IEEE Transactions on Signal Processing*, vol. 66, no. 12, pp. 3240–3255, 2018.

[6] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[7] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

[8] J. Zeng and W. Yin, "On nonconvex decentralized gradient descent," *IEEE Transactions on Signal Processing*, vol. 66, no. 11, pp. 2834–2848, 2018.

[9] D. Jakovetić, J. Xavier, and J. M. Moura, "Fast distributed gradient methods," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1131–1146, 2014.

[10] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network Newton distributed optimization methods," *IEEE Transactions on Signal Processing*, vol. 65, no. 1, pp. 146–161, 2016.

[11] D. Bajovic, D. Jakovetic, N. Krejic, and N. K. Jerinkic, "Newton-like method with diagonal correction for distributed optimization," *SIAM Journal on Optimization*, vol. 27, no. 2, pp. 1171–1203, 2017.

[12] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Püschel, "D-ADMM: A communication-efficient distributed algorithm for separable optimization," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2718–2723, 2013.

[13] 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, 2014.

[14] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "DLM: Decentralized linearized alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 63, no. 15, pp. 4051–4064, 2015.

[15] M. Maros and J. Jaldén, "On the Q-linear convergence of distributed generalized ADMM under non-strongly convex function components," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 3, pp. 442 – 453, 2018.

[16] D. Jakovetić, J. M. Moura, and J. Xavier, "Linear convergence rate of a class of distributed augmented lagrangian algorithms," *IEEE Transactions on Automatic Control*, vol. 60, no. 4, pp. 922–936, 2014.

[17] K. Scaman, F. Bach, S. Bubeck, Y. T. Lee, and L. Massoulié, "Optimal algorithms for smooth and strongly convex distributed optimization in networks," in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 3027–3036.

[18] T. H. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 482–497, 2014.

[19] N. S. Aybat, Z. Wang, T. Lin, and S. Ma, "Distributed linearized alternating direction method of multipliers for composite convex consensus optimization," *IEEE Transactions on Automatic Control*, vol. 63, no. 1, pp. 5–20, 2017.

[20] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[21] ——, "A proximal gradient algorithm for decentralized composite optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 22, pp. 6013–6023, 2015.

[22] Z. Li, W. Shi, and M. Yan, "A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates," *IEEE Transactions on Signal Processing*, vol. 67, no. 17, pp. 4494–4506, 2019.

[23] S. A. Alghunaim, K. Yuan, and A. H. Sayed, "A linearly convergent proximal gradient algorithm for decentralized optimization," *arXiv preprint arXiv:1905.07996*, 2019.

[24] P. Di Lorenzo and G. Scutari, "NEXT: In-network nonconvex optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.

[25] G. Scutari and Y. Sun, "Distributed nonconvex constrained optimization over time-varying digraphs," *Mathematical Programming*, vol. 176, no. 1-2, pp. 497–544, 2019.

[26] S. U. Pillai, T. Suel, and S. Cha, "The Perron-Frobenius theorem: some of its applications," *IEEE Signal Processing Magazine*, vol. 22, no. 2, pp. 62–75, 2005.

[27] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing Markov chain on a graph," *SIAM Review*, vol. 46, no. 4, pp. 667–689, 2004.

[28] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[29] O. L. Mangasarian and W. H. Wolberg, "Cancer diagnosis via linear programming," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 1990.