Distributed and Parallel ADMM for Structured Nonconvex Optimization Problem

Xiangfeng Wang, Junchi Yan, *Member, IEEE*, Bo Jin, and Wenhao Li

***Abstract*—The nonconvex optimization problems have recently attracted significant attention. However, both efficient algorithm and solid theory are still very limited. The difficulty is even pronounced for structured large-scale problems in many real-world applications. This article proposes an application-driven algorithmic framework for structured nonconvex optimization problems with distributed and parallel techniques, which jointly handles the high dimensionality of model parameters and distributed training data. The theoretical convergence of our algorithm is established under moderate assumptions. We apply the proposed method to popular multitask applications, including a multitask reinforcement learning problem. The promising performance demonstrates our framework is effective and efficient.**

***Index Terms*—Distributed, large-scale optimization, multitask reinforcement learning, nonconvex optimization, parallel.**

I. Introduction and Related Work

In this article, we consider the following nonconvex optimization problem, that is:

$$\min_{\{\mathbf{x}_i\in\mathbb{R}^{n_i}\}} \underbrace{\sum_{i=1}^{m} f_i(\mathbf{x}_i) + h(\mathbf{x}_1,\ldots,\mathbf{x}_m)}_{g(\mathbf{x}_1,\ldots,\mathbf{x}_m)} \tag{1}$$

where $f_i:\mathbb{R}^{n_i}\to\mathbb{R}$ is assumed smooth, but can be nonconvex and Lipschitz differentiable with gradient Lipschitz constant L_i; and h is a Lipschitz continuous function (possibly nonconvex and nonsmooth) with respect to all variables $\{\mathbf{x}_1,\ldots,\mathbf{x}_m\}$. To motivate our work, we first present some discussion on machine-learning applications based on (1).

Manuscript received March 27, 2019; revised August 20, 2019 and October 15, 2019; accepted October 18, 2019. Date of publication December 30, 2019; date of current version September 8, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61702188 and Grant 11871279, in part by the General Program of Shanghai Science and Technology Commission under Grant 19ZR1414200, and in part by the NSFC-Zhejiang Joint Fund for the Integration of Industrialization and Information under Grant U1609220. This article was recommended by Associate Editor D. Zhao. *(Corresponding author: Bo Jin.)*

X. Wang, B. Jin, and W. Li are with the School of Computer Science and Technology, East China Normal University, Shanghai 200062, China, also with the MOE Key Laboratory for Advanced Theory and Application in Statistics and Data Science, East China Normal University, Shanghai 200062, China, and also with the Shanghai Institute of Intelligent Science and Technology, Tongji University, Shanghai 200092, China (e-mail: xfwang@cs.ecnu.edu.cn; bjin@cs.ecnu.edu.cn).

J. Yan is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China, and also with the MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: yanjunchi@sjtu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2019.2950337.

Digital Object Identifier 10.1109/TCYB.2019.2950337

A. Motivating Application: Multitask Learning

Many machine-learning applications can be incorporated into this general formulation (1), e.g., the multitask learning (MTL) problem. MTL aims to jointly learn a batch of tasks, which allows information to be transferred throughout related tasks, in contrast to learning each task separately. MTL has been widely applied in a variety of applications, such as survival analysis [1], subspace segmentation [2], fine-grained visual categorization [3], and fMRI analysis [4].

Ideally, when all tasks in question share common structures, direct joint MTL can be conducted in an outlier task agnostic way. There exist effective algorithms by introducing penalty terms, e.g., group lasso or nuclear norm [5], [6]. However, in real-world problems, the hidden relationship among tasks can be very complicated. Usually, a majority of tasks can be clustered into natural groups with shared commonality, while there might be still a few tasks hardly relevant to the major groups and treated as an outlier. Accordingly, two concepts, that is, task grouping and task outlier, outlier task detection is introduced to address the challenge, and are often jointly performed with task grouping.

Assuming that we are given m distributed tasks with the training data $\{\mathbf{A}_i,\mathbf{y}_i\}_{i=1}^m$, $\mathbf{A}_i\in\mathbb{R}^{m_i\times n}$ and $\mathbf{y}_i\in\mathbb{R}^{m_i}$ denote the data matrix and the supervision (label) of the ith task. Each row of $\mathbf{A}_i$ represents a sample with dimension n, and m_i denotes the number of samples of the ith task. For the machine-learning problems, we usually consider learning the mapping g_i for each task: $g_i(\mathbf{A}_i,\mathbf{x}_i)\to\mathbf{y}_i$, where $\mathbf{x}_i\in\mathbb{R}^n$ denotes the model parameter vector for task i. Define target weight matrix $\mathbf{X}$ as a stack of all parameter vectors of the m tasks: $\mathbf{X}=[\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_m]^T\in\mathbb{R}^{m\times n}$. In general, there are two types of MTL loss functions.

In the Absence of Outlier Task: Assuming that all tasks have common relationship, we consider the following model:

$$\min_{\{\mathbf{x}_i\}} \underbrace{\sum_{i=1}^{m} f_i(\mathbf{x}_i)}_{\text{Empirical loss}} + \underbrace{\gamma h(\mathbf{X})}_{\text{Penalty}} \tag{2}$$

where $f_i(x_i)\triangleq\ell(g_i(\mathbf{A}_i,\mathbf{x}_i))$ is the empirical loss function with respect to the ith task, and $\ell(\cdot)$ is a basic loss function. For the reason that the mapping g_i could both be linear and nonlinear, the empirical loss function of each task f_i can be linear regression, logistic regression, and even the loss function of deep learning. Thus, f_i could be convex or nonconvex. h is considered as the penalty function, which, in fact, encodes the relationship among tasks. For some popular convex functions,

such as the group-lasso penalty [5], [7] and the nuclear-norm penalty [6], [8], h can be nonconvex, while it is often carefully designed with some special properties, e.g., smoothly clipped absolute deviation (SCAD) or minimax concave plus (MCP) penalty [9]–[11].

In the Presence of Outlier Task: Assume that some of the tasks are outliers, that is, they do not belong to any task group. To achieve related tasks grouping and outlier task detection, one generalized loss is given by [12]

$$\min_{\{(\mathbf{s}_i,\boldsymbol{\ell}_i)\}} \underbrace{\sum_{i=1}^{m} f_i(\mathbf{s}_i,\boldsymbol{\ell}_i)}_{\text{Empirical loss}} + \underbrace{\gamma_s h_1(\mathbf{S}) + \gamma_\ell h_2(\mathbf{L})}_{\text{Penalty}}$$
$$\text{s.t. } \mathbf{S} = [\mathbf{s}_1,\ldots,\mathbf{s}_m]^T,\ \mathbf{L} = [\boldsymbol{\ell}_1,\ldots,\boldsymbol{\ell}_m]^T \tag{3}$$

where the model parameter $\mathbf{x}_i$ is separated into two terms $\mathbf{s}_i$ and $\boldsymbol{\ell}_i$. The empirical loss of each task $f_i(\mathbf{s}_i,\boldsymbol{\ell}_i)$ has the same assumptions as the case above. There are two separate penalty terms h_1 and h_2, which are often used to describe the cross-task structure for task grouping and outlier task detection, respectively. For instance, in [13], $h_1(\mathbf{S})$ is set to be $\|\mathbf{S}\|_\star$ for coupling the related tasks, while $h_2(\mathbf{L})$ is designed to be $\|\mathbf{L}\|_{1,2}$ for identifying the outlier tasks. In [14], a robust MTL formulation is presented with $h_1(\mathbf{S}) = \|\mathbf{S}\|_{1,2}$ and $h_2(\mathbf{L}) = \|\mathbf{L}^T\|_{1,2}$, where the first group-lasso penalty encourages all related tasks to share a common set of features, and the second group-lasso penalty introduces all zero or nonzero columns of $\mathbf{L}$ with the nonzero columns corresponding to outlier tasks. Meanwhile, the nonconvex SCAD-type penalty and MCP-type penalty also can be introduced to describe the specific necessary problem structure [9].

Both problems (2) and (3) can be incorporated in the general problem (1). In the following text, we focus on this general optimization problem (1) and check the efficiency of this article from both an algorithmic and theoretical perspective through some MTL application problems.

B. Related Work

Problem (1) can be considered as a special case of a more general optimization problem, that is

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) + h(\mathbf{x}) \tag{4}$$

where f is assumed to be Lipschitz differentiable (possibly nonconvex) and h is assumed to be a Lipschitz continuous (possibly nonconvex and nonsmooth). It is obvious that $f(\mathbf{x})$ has a separable structure for the specific problem (1). Various data fitting problems in machine learning, signal processing, and statistics can be formulated as problem (4), where f is a loss function measuring the deviation of a solution from the observations and h is a regularizer intended to induce a certain structure in the solution. With the advent of big data era, the problem instances are typically large scale, and the first-order methods are popularly used to solve problem (4) in recent years, such as the proximal gradient (PG) method, block coordinate descent-type (BCD-type) methods, and extended accelerated variants.

In general, when f and h are both convex functions, the $\mathcal{O}(1/k)$ convergence of most first-order methods can be obtained, with respect to the objective function value sequence converge to the optimal value [15]. Recently, the fast iterative shrinkage-thresholding algorithm (FISTA) was proposed in [16] based on Nesterov's extrapolation techniques [17] and accelerates the PG method to the $\mathcal{O}(1/k^2)$ convergence rate. Moreover, the objective functions often have an explicit form or specific structure in plenty of applications. We can observe that the numerical performance of many first-order methods reflects significantly better than that suggested by theory. The structure of the problem can benefit the algorithm to obtain a sharper convergence curve. In other words, more assumptions of problem structures are needed, and should be well employed to guarantee a better convergence rate of the PG method. Schmidt *et al.* [18] proved a global linear convergence rate with respect to the objective function value sequence with f strongly convex and h convex assumptions. More recently, Tao *et al.* [19] established the local linear convergence of the PG algorithm, and FISTA applied to the specific LASSO problem, by using f as is a least squares loss function and h as $\tau\|\cdot\|_1$, where $\tau > 0$.

Under the nonconvex setting, recently, the convergence rate analysis of many first-order methods also gains popularity. Attouch *et al.* [20] claimed that under the assumption that the objective function f admits the Kurdyka–Lojasiewicz (KL) property, a bounded sequence can be guaranteed iteratively, as long as one can ensure the *sufficient decrease* property through the algorithm (this property is obviously valid for convex case, however, for nonconvex, extra assumptions must be added). Shortly thereafter, Bolte *et al.* [21] introduced a proximal alternating linearized minimization (PALM) algorithm. In particular, they derived a self-contained convergence analysis framework building on the powerful KL property. On the other hand, the so-called iPiano algorithm was presented in [22]. Specifically, they focused on the case where f is smooth (possibly nonconvex) and h is convex (possibly nonsmooth). Based on the KL property, Ochs *et al.* [22] revealed that any accumulation point of the sequence generated by iPiano is a critical point. Moreover, the objective function value converges, and the iPiano converges in a sublinear rate. Xu and Yin [23] proposed a BCD-type algorithm for nonconvex optimization and established its global convergence (of the entire sequence) to a critical point based on the KL property. In [9] and [24], a general iterative shrinkage and thresholding algorithm was proposed for the nonconvex regularized optimization problems in the form of (4). Another iteratively reweighted nuclear-norm algorithm was given in [10] to solve a nonconvex nonsmooth low-rank minimization problem, which is also in the form of (4).

C. Main Contributions

However, all of these algorithms are designed for the general problem (4), and have not yet taken advantage of the special structure of (1). Moreover, the separable structure of function f can benefit more for distributed and parallel computing, especially for large-scale problems. Each "task" belongs to each

agent and may be distributed deployed which significantly drive the distributed computing. Both the extremely growing problem dimension scale and data volume push us to introduce distributed and parallel computing. Further, formulation (1) has separable structure with different smooth task-driven functions for difference agents, however, the regularization term is combined together. All of these reasons motivate us to design some novel efficient algorithms for problem (1) based on the distributed and parallel computing techniques. Therefore, we will propose a new distributed and parallel algorithm framework to handle the nonconvexity and specific structure of (1). The main contributions of this article are as follows.

1) We devise a new algorithm framework for the general nonconvex optimization problem (1), which takes full advantage of the separable structure of (1).
2) We establish the theoretical analysis for the convergence of our algorithm framework under rather moderate assumptions. This is novel in the literature, because the ADMM-type methods on the nonconvex optimization problems are still limited. Our new convergence results can reinforce the understanding of ADMM for the nonconvex problems. To the best of our knowledge, this is the first convergence analysis result of the ADMM-type method when both f_i and h are nonconvex.
3) As shown in the experiment, our framework can incorporate existing MTL formulations either in the presence of outlier tasks or not, and can routinely adapt them to the parallel and distributed computing setting.

The remainder of this article is organized as follows. Section II introduces the distributed and parallel algorithm framework, and more further discussions followed by a theoretical analysis in Section III. The experimental results on MTL applications are reported in Section IV, and Section V concludes this article.

Notations: Before we go into the details, some notations are defined first. Scalars, vectors, matrices, and sets are denoted by lower case letters, boldface lower case letters, boldface capital letters, and calligraphic capital letters, respectively. x_i and x^j denote the ith entry and the jth block of a vector $\mathbf{x}$, while x_{ij} denotes the (i,j)th entry of a matrix $\mathbf{X}$. $\|\cdot\|_1$ and $\|\cdot\|_2$ denote the ℓ_1-norm and ℓ_2-norm of a vector. $\|\cdot\|_\star$ and $\|\cdot\|_F$ denote the nuclear norm and Frobenius norm of a matrix, further $\ell_{p,q}$-norm of a matrix X is defined as $\|\mathbf{X}\|_{p,q} = \{\sum_i [(\sum_j x_{ij}^q)^{1/q}]^p\}^{1/p}$.

II. Proposed Algorithm Framework

First, we reformulate (1) to the following *linear constrained separable minimization problem*:

$$\begin{aligned} \min_{\{\mathbf{x}_i\in\mathbb{R}^{n_i},\mathbf{z}_i\in\mathbb{R}^{n_i}\}} \quad & \sum_{i=1}^{m} f_i(\mathbf{x}_i) + h(\mathbf{z}_1,\ldots,\mathbf{z}_m) \\ \text{s.t.} \quad & \mathbf{x}_i = \mathbf{z}_i,\ i = 1,\ldots,m \end{aligned} \tag{5}$$

where $\{\mathbf{z}_i\}_{i=1}^m$ are introduced as the auxiliary variables, and help decouple $f_i(\cdot)$ and $h(\cdot)$ to allow f_i to be computed in a distributed way. This is often a preferred or must have characteristic in many real-world scenarios, e.g., for privacy preservation.

We define the augmented Lagrangian function as

$$\begin{aligned} \mathcal{L}(\mathbf{x}_i,\mathbf{z}_i,\boldsymbol{\lambda}_i) = & \sum_{i=1}^{m} f_i(\mathbf{x}_i) + h(\mathbf{z}_1,\ldots,\mathbf{z}_m) \\ & - \sum_{i=1}^{m}\langle\boldsymbol{\lambda}_i,\mathbf{x}_i-\mathbf{z}_i\rangle + \sum_{i=1}^{m}\frac{\beta_i}{2}\|\mathbf{x}_i-\mathbf{z}_i\|^2 \end{aligned} \tag{6}$$

where $\{\boldsymbol{\lambda}_i\}_{i=1}^m$ are the Lagrangian multipliers, and $\{\beta_i\}_{i=1}^m$ are the weights for the linear constraint violation penalties. A popular algorithm, so called the alternating direction method of multipliers (ADMM) [25], is usually considered to solve (5), whose steps in iteration k+1 are given as follows:

$$\begin{cases} \left(\mathbf{z}_1^{k+1},\ldots,\mathbf{z}_m^{k+1}\right) = \arg\min\ h(\mathbf{z}_1,\ldots,\mathbf{z}_m) \\ \qquad\qquad + \sum_{i=1}^{m}\frac{\beta_i}{2}\left\|\mathbf{z}_i - \left(\mathbf{x}_i^k - \frac{\boldsymbol{\lambda}_i^k}{\beta_i}\right)\right\|^2 & \text{(7a)} \\ \mathbf{x}_i^{k+1} = \arg\min f_i(\mathbf{x}_i) + \frac{\beta_i}{2}\left\|\mathbf{x}_i - \mathbf{z}_i^{k+1} - \frac{\boldsymbol{\lambda}_i^k}{\beta_i}\right\|^2 & \text{(7b)} \\ \boldsymbol{\lambda}_i^{k+1} = \boldsymbol{\lambda}_i^k - \beta_i\left(\mathbf{x}_i^{k+1} - \mathbf{z}_i^{k+1}\right). & \text{(7c)} \end{cases}$$

Note that the above algorithm has the exact form as the classical ADMM, where the variables $\{\mathbf{z}_i\}_{i=1}^m$ are taken as the first block of primal variable, and the collection $\{\mathbf{x}_i\}_{i=1}^m$ as the second block. The two primal blocks are updated in a sequential manner, followed by an inexact dual ascent step ($\{\boldsymbol{\lambda}_i\}$). The ADMM was originally proposed in [26] for nonlinear elliptic equations, and it recently has found a wide range of applications in various areas, such as image processing, statistical learning, computer vision, wireless communication networks, and so on. It becomes a benchmark first-order solver for the convex minimization models with separable objective functions, and is being extensively explored in other various contexts, such as the nonconvex or multiblock contexts. We refer to [25] and [27] for a more comprehensive treatment of ADMM. Unlike the convex case, for which the behavior of ADMM has been investigated quite extensively, when the objective becomes nonconvex, the convergence issue of ADMM remains largely open. Interestingly, it has been observed by many researchers that the ADMM works empirically extremely well for various applications involving nonconvex objectives [28]–[30]. However, to the best of our knowledge, existing convergence analysis of ADMM for the nonconvex problems is very limited, and all known global convergence analysis needs to impose uncheckable conditions on the sequence generated by the algorithm. Recently, Hong *et al.* [31] established the convergence of the ADMM for certain types of nonconvex problems, including the consensus and sharing problems without any assumption on the iterations. Problem (5) has a similar structure with the reformulation of the consensus problem. However, the consensus problem only contains a function h about a consensus variable $\mathbf{x}$ with all $\mathbf{x}_i = \mathbf{x}$ constraint, which can be considered as a special case of (5).

Before proposing our algorithm framework, we give some assumptions and computing techniques, in order to consider a

Algorithm 1 Distributed and Parallel ADMM

Initialization 1. Set parameters $\{\beta_i > 0\}$, zero $\{\mathbf{x}_i^0\}$, $\{\boldsymbol{\lambda}_i^0\}$.
Procedure 2. Calculate step (a)-(c) until converge.
(a) At iteration k+1, pick an index set $\mathcal{C}^{k+1}$;
(b) If $0 \in \mathcal{C}^{k+1}$, compute:

$$\left(\mathbf{z}_1^{k+1}, \cdots, \mathbf{z}_m^{k+1}\right) = \arg\min \ h(\mathbf{z}_1, \cdots, \mathbf{z}_m) + \sum_{i=1}^{m} \frac{\beta_i}{2} \left\| \mathbf{z}_i - \left(\mathbf{x}_i^k - \frac{\boldsymbol{\lambda}_i^k}{\beta_i}\right) \right\|^2,$$

Else $\mathbf{z}_i^{k+1} = \mathbf{z}_i^k$;
(c) If $i \in \mathcal{C}^{k+1}$ and $i \neq 0$, compute

$$\mathbf{x}_i^{k+1} = \arg\min f_i(\mathbf{x}_i) + \frac{\beta_i}{2} \left\| \mathbf{x}_i - \mathbf{z}_i^{k+1} - \frac{\boldsymbol{\lambda}_i^k}{\beta_i} \right\|^2, \quad (9)$$

Update the related dual variable

$$\boldsymbol{\lambda}_i^{k+1} = \boldsymbol{\lambda}_i^k - \beta_i\left(\mathbf{x}_i^{k+1} - \mathbf{z}_i^{k+1}\right), \quad (10)$$

Else $\mathbf{x}_i^{k+1} = \mathbf{x}_i^k$ and $\boldsymbol{\lambda}_i^{k+1} = \boldsymbol{\lambda}_i^k$. $k = k + 1$;

more flexible algorithm in choosing the order of the update of both the primal and the dual variables. Let $t = 0$ be the index for the primal variable $(\mathbf{z}_1, \ldots, \mathbf{z}_m)$, let $t = 1, \ldots, m$ be the indices for the primal variable blocks $\mathbf{x}_1, \ldots, \mathbf{x}_m$, respectively, and let $\mathcal{C}^k \subseteq \{0, 1, \ldots, m\}$ denote the set of variables updated in iteration k. We define a period-T essentially cyclic update rule (*essentially cyclic update rule*) [32]: there exists a given period $T \geq 1$, during which each index is updated at least once. More specifically, at iteration k, update all the variables in an index set $\mathcal{C}^k$ whereby

$$\bigcup_{i=1}^{T} \mathcal{C}^{k+i} = \{0, 1, \ldots, m\}, \quad \forall t. \quad (8)$$

This rule is commonly used and has a very wide range of accommodation. The scheme to determine the index set $\mathcal{C}^k$ in each iteration can be either deterministic (determine the sequence $\{\mathcal{C}^k\}$ beforehand) or randomized (each variable is chosen randomly with probability in $\mathcal{C}^k$).

We propose our algorithm framework for (5) in the spirit of ADMM, and the main algorithmic pipeline is depicted in the following algorithm. This algorithm framework is a standard ADMM-type algorithm, and is similar to the one proposed in [31], except that their algorithm is focused on the consensus problem which can be considered as a special case of (5). Moreover, they only consider the case that h is assumed to be a convex function. In contrast, our algorithm can deal with both nonconvex functions f and h.

In the following text, we give some concrete discussions on the algorithm framework below, especially for efficient implementation.

1) *Distributed (Privacy Preserving) Computing:* Our algorithm framework is built on a network of star topology, where there is a master node to handle the computation involving the function h. All other distributed task nodes handle the subproblems involving their own private data, while updates of multipliers are well arranged to task nodes. From the point of view of data security, privacy preserving can be guaranteed for the reason that we need no data transfer between all the nodes.
2) *Asynchronous Parallel Computing:* In step 2.c, only the selected variables are updated in parallel and distributed, together with the relevant Lagrangian multipliers. This parallel scheme is highly asynchronous, especially for the randomized selection technique. Moreover, the essentially cyclic update rule can be considered as a controller to dominate the asynchronous degree. The period-T essentially cyclic update rule ensures that the maximum asynchronous bound is less than T, which is generally a necessary condition to guarantee algorithm convergence.
3) *Efficient Parallel Solution of (9):* The most important issue is the huge dimension of parameters $\{\mathbf{x}_i\}$. One of the key calculation techniques is to separate problem (9) into low-dimensional subproblems, and further to unify them with specific schemes. For instance, we can substitute (9) with the following computing scheme, that is, for $j = 1, \ldots, r_i$:

$$\mathbf{x}_i^j = \arg\min \ f_i\left(\left(\mathbf{x}_i^1\right)^k, \ldots, \mathbf{x}_i^j, \ldots, \left(\mathbf{x}_i^{m_i}\right)^k\right) + \frac{\beta_i}{2} \left\| \mathbf{x}_i^j - \left(\mathbf{z}_i^{k+1} + \frac{\boldsymbol{\lambda}_i^k}{\beta_i}\right)^j \right\|^2 + \frac{m_i - 1}{2} \left\| \mathbf{x}_i^j - \left(\mathbf{x}_i^j\right)^k \right\|^2 \quad (11)$$

where variable $\mathbf{x}_i$ is separated into r_i blocks, which are denoted as

$$\left((\mathbf{x}_i)^1, \ldots, (\mathbf{x}_i)^j, \ldots, (\mathbf{x}_i)^{r_i}\right).$$

Equation (11) means that a Jacobi-type pattern is taken to inexactly compute $\mathbf{x}_i^{k+1}$ in parallel, in other words, (9) is replaced by m_i subproblems (11). We not only separate their computation to distributed task nodes but also use the parallel computing technique to block wisely calculate sub-blocks $\{(\mathbf{x}_i)^j\}$, respectively.
4) *Efficient Inexact Implementation of (9)*: In fact, the optimal solution of problem (9) is hard to guarantee even when the closed-form solution can be expressed, e.g., $f_i(\mathbf{x}_i)$ is a quadric function and in the form of $\frac{1}{2}\mathbf{x}_i^T\mathbf{A}_i\mathbf{x}_i$. With properly chosen β_i, problem (9) becomes a convex problem; as a result, the optimal solution of (9) can be written as $(\beta_i\mathbf{I} + \mathbf{A}_i)^{-1}(\beta_i\mathbf{z}_i^{k+1} + \boldsymbol{\lambda}_i^k)$ in closed form. However, the inverse of $(\beta_i\mathbf{I} + \mathbf{A}_i)$ is still difficult to calculate, and we need to estimate the inverse or introduce an iterative algorithm to inexactly solve (9). The above computing scheme (11) can also be considered as an inexact version as we guarantee an inexact estimation of the optimal solution of (9) through a series of parallel subproblems. The key issue of implementing an iterative algorithm for (9) is how to choose a proper inner iteration stopping criterion. Recently, Yue *et al.* [33] introduced a new efficient implementation scheme of ADMM for

a specific linear constrained convex separable problem which includes (5) if f_i is assumed to be quadratic.

A. Discussions on the Joint Function h

Here, we focus on the joint function $h(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ of problem (1), which is greatly related to the problem structure. For instance, in MTL applications (2), h can be chosen as the $\ell_{1,2}$-norm, which indicates the fully similar parameter activation structure for all tasks; or matrix nuclear norm (trace norm) which indicates that all task parameters have a linear relationship. Recently, the nonconvex penalty functions start to draw attention to the machine learning community. Although they have shown potential in obtaining better accuracy and generalization, principled algorithms are not naturally raised in contrast to on-the-shelf convex optimization solvers. In [10], [24], [34], and [35], some nonconvex regularizers are introduced to machine-learning applications, such as SCAD [36] and MCP [37]. But they have statistical oracle properties, that is, unbiasedness, sparsity, and continuity. Further, some matrix form SCAD-type or MCP-type penalties are established in [10] and [34].

In this article, the proximal operator with respect to function $h(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ is defined as follows:

$$\text{prox}_h^{\gamma}(\{\mathbf{a}_i\}) := \arg\min \; h(\mathbf{x}_1, \ldots, \mathbf{x}_m) + \frac{1}{2\gamma}\sum_{i=1}^{m}\|\mathbf{x}_i - \mathbf{a}_i\|_2^2. \tag{12}$$

More specifically, if function $h(\cdot)$ is defined on the matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m]$ like $h(\mathbf{X})$ in MTL applications, (12) can be equivalently written as

$$\text{prox}_h^{\gamma}(\mathbf{A}) := \arg\min \; h(\mathbf{X}) + \frac{1}{2\gamma}\|\mathbf{X} - \mathbf{A}\|_F^2$$

where $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m]$. In order to have a well-defined subproblem, usually h is required to be prox-bounded. That is, there exists a $\gamma_h > 0$ such that when γ is selected as $0 < \gamma < \gamma_h$, prox_h^{γ} is well defined. In addition, we assume that the global optimum of optimization problem (12) is easy to calculate for any $\{\mathbf{a}_i\}$ along with an appropriate $\gamma > 0$. If h is convex, the prox-bounded assumption can be directly satisfied because (12) will be strongly convex. Specifically, we can guarantee the closed-form global minimization of (12) if h is chosen to be $\ell_{1,2}$-norm or nuclear norm. As for the nonconvex case, for example, the SCAD-type or MCP-type function, we can also obtain analytical solutions, respectively, for $\gamma < a - 1$ and $\gamma < a$ accordingly, where the definition of a can be found in the definition of SCAD and MCP in the Appendix. Due to space limitations, we leave the analytical solutions in the Appendix.

III. Theoretical Convergence Results

In this section, we briefly discuss the theoretical results of the above proposed algorithm framework. Although our method can be considered as a modified extension of classical ADMM, the nonconvexity brings many new challenges. The following theoretical achievement is novel, especially as the first convergence result of ADMM for problems with nonconvex joint function h. To this end, we make a few assumptions as follows.

Assumptions:

1) There exists a positive constant $L_i > 0$ such that $\forall i$

$$\|\nabla f_i(\mathbf{x}_i) - \nabla f_i(\mathbf{y}_i)\| \le L_i\|\mathbf{x}_i - \mathbf{y}_i\|, \;\; \forall \mathbf{x}_i, \mathbf{y}_i. \tag{13}$$

2) The parameter β_i is chosen large enough such that the x_i subproblem (9) is strongly convex with modulus γ_i, $\beta_i\gamma_i \ge 2L_i^2$ and $\beta_i \ge L_i$ for all i.
3) g is bounded from below, that is, $\underline{g} = \min g(\mathbf{x}_1, \ldots, \mathbf{x}_m) > -\infty$.

We make some comments based on the above assumptions. As β_i increases, subproblem (9) eventually will be strongly convex with respect to $\mathbf{x}_i$, and the corresponding strong convexity modulus γ_i is a monotonic increasing function of β_i. For the reason that f_i is nonconvex, the condition $\beta_i\gamma_i \ge 2L_i^2$ implies $\beta_i \ge L_i$ ($\beta_i > \gamma_i$) [31].

Theorem 1: Suppose the above assumptions hold and let $\{\{\mathbf{x}_i^k\}, \{\mathbf{z}_i^k\}, \{\boldsymbol{\lambda}_i^k\}\}$ be the sequence generated by Algorithm 1, then we have the following:

1) $\lim_{k\to\infty} \|\mathbf{x}_i^k - \mathbf{z}_i^k\| = 0, \;\; \forall i$;
2) let $(\{\mathbf{x}_i^\star\}, \{\mathbf{z}_i^\star\}, \{\boldsymbol{\lambda}_i^\star\})$ denote any limit point of the sequence $\{\{\mathbf{x}_i^k\}, \{\mathbf{z}_i^k\}, \{\lambda_i^k\}\}$. Then, the following statement is true:

$$0 = \nabla f_i\big(\mathbf{x}_i^\star\big) - \boldsymbol{\lambda}_i^\star, \quad \forall i \tag{14a}$$

$$\big(\{\mathbf{z}_i^\star\}\big) \in \arg\min \; h(\{\mathbf{z}_i\}) - \sum_{i=1}^{m}\big\langle \boldsymbol{\lambda}_i^\star, \mathbf{x}_i^\star - \mathbf{z}_i\big\rangle \tag{14b}$$

$$\mathbf{x}_i^\star = \mathbf{z}_i^\star, \quad \forall i \tag{14c}$$

which means any limit point of Algorithm 1 is a stationary solution of problem (5).

Proof: Unlike the convex case, it becomes significantly difficult to guarantee the decreasing of the distance of $\{\{\mathbf{x}_i^k\}, \{\mathbf{z}_i^k\}, \{\boldsymbol{\lambda}_i^k\}\}$ to a stationary point. Instead, at first, we will prove the sufficient decrease of the augmented Lagrangian function value sequence; based on this, we will prove the limitation of the augmented Lagrangian function value sequence exists and is lower bounded; then we will reach the above convergence results a) and b).

In the following text, we briefly present these two major steps.

1) The sufficient decrease with respect to the augmented Lagrangian function value

$$\mathcal{L}\Big(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^{k+1}\Big) - \mathcal{L}\Big(\mathbf{x}_i^k, \mathbf{z}_i^k, \boldsymbol{\lambda}_i^k\Big) \le \sum_{i\ne 0, i\in\mathcal{C}^{k+1}}\left(\frac{L_i^2}{\rho_i} - \frac{\gamma_i}{2}\right)\Big\|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\Big\|^2. \tag{15}$$

It is obvious that (18) implies that if the assumption $\beta_i\gamma_i \ge 2L_i^2$ holds, then the augmented Lagrangian function value will always decrease. This is the key property in order to guarantee convergence results.
2) The limit of the augmented Lagrangian function value exists and is lower bounded by $\underline{g}$ defined in the third assumption, that is, $\lim_{k\to\infty} \mathcal{L}(\mathbf{x}_i^k, \mathbf{z}_i^k, \boldsymbol{\lambda}_i^k) \ge \underline{g}$, which ensures the limit can be achieved.

Due to space limitations, we leave the proof details to the supplementary material. ■

A. Discussions and Computational Complexity

This is the first theoretical result to analyze the convergence of ADMM for solving a certain nonconvex problem (5), where both $f_i(\mathbf{x}_i)$ and $h(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ are nonconvex. We need to control that $\{\beta_i\}$ are large enough to guarantee convergence, while in fact, the large enough $\{\beta_i\}$ ensures that the $(\mathbf{z}_1, \ldots, \mathbf{z}_m)$ subproblems (9) is well defined and the $\{\mathbf{x}_i\}$ subproblems (9) are all strongly convex for all i. For efficient parallel and inexact implementation cases of (9), that is, bullets 3 and 4 in the discussion part of our framework, we can also obtain the convergence by following the same analytic procedure. We leave the details and extensions in future work.

In the following text, we give a brief discussion about the computational complexity of the proposed algorithm framework. In each iteration, the subproblem of $(\mathbf{z}_1, \ldots, \mathbf{z}_m)$ (9) can usually be computed in an efficient elementwise fashion, with a complexity of $\mathcal{O}(\sum_{i=1}^m n_i)$. In addition, some extra calculations are needed, like SVD, and the worst-case complexity should be $\mathcal{O}(\max(m, \max(n_i))^3)$. The subproblem for $\mathbf{x}_i$ (9) has analytical solutions for quadratic f_i, while the main issue is to compute the inverse of a series of $n_i \times n_i$ matrices, but it need to be computed only once and this can be done in initialization. The complexity for $\mathbf{x}_i$ subproblem should be $\mathcal{O}(n_i^2)$ for matrix-vector multiplication. The complexity for updating $\boldsymbol{\lambda}_i$ and $\boldsymbol{\mu}_i$ is $\mathcal{O}(n_i)$ and can be ignored. The overall computational complexity of our algorithm framework in each iteration is $\mathcal{O}(\max(m, \max(n_i))^3 + \sum_{i=1}^m n_i^2 + 2\sum_{i=1}^m n_i)$.

IV. Experiments and Discussion

A. Traditional Multitask Learning

First, we consider traditional MTL applications, and mainly focus on problem (2) in the absence of outlier task. We use the logistic regression for the empirical loss function f_i. h can be chosen as the group-SCAD, the group-MCP, the matrix-SCAD, and the matrix-MCP, respectively, which are all nonconvex functions. The concrete formulas are moved to the supplementary material. We compare with the popular FISTA method [14], [16], which has been widely used to solve MTL applications. In order to fairly and effectively compare with FISTA, we also distributed implement FISTA on our computing cluster. The master node handles the iterative update while the worker nodes calculate the gradient with respect to their own task and dataset. The iterates and gradients are transferred through the network.

We use three real-world standard benchmark datasets for performance evaluation and comparison, which have been widely used in literature and are described as follows.

1) *SARCOS [38]:* The dataset relates to an inverse dynamics problem for a seven degrees-of-freedom SARCOS anthropomorphic robot arm. The task is to map from a 21-D input space (seven joint positions, seven joint velocities, and seven joint accelerations) to the corresponding seven joint torques. There are 44 484 training examples. As a result there are seven tasks to learning simultaneously, that is, m =7, while the feature dimension n_i equals to 21.
2) *SCHOOL [39]:* The dataset is from the inner London education authority. It consists of the examination scores of 15 362 students from 139 secondary schools in 1985, 1986, and 1987. There are 139 tasks in total, corresponding to examination scores prediction in each school. The input features include the year of the examination, four school-dependent features, and three student-dependent features. We follow the same setup as previous MTL works and obtain a 27-D binary variable for each example.
3) *E2006-tfidf [40]:* The dataset has 16 087 training samples and 3308 testing samples with $n = 150\,360$ features per sample, which involves a feature representation of a series of "Form 10-K" company annual reports. Variables include history and organization of the company, equity and subsidiaries, financial information, etc. We randomly separate the entire training dataset into ten parts. Each part represents one task for MTL with 1600 samples, while the last task contains 1687 samples.

Experiments are carried on a cluster with ten task nodes and one master node, where each machine is equipped with 2 Intel Xeon E5450@3.00 GHz CPUs, 16-GB memory, and 10-Gbs Ethernet. Codes are written in MATLAB 8.6 (R2015b) using MPI for communication.

The parameter γ is set to be 0.1. The stopping criterion of FISTA for problem (1) is set by

$$\frac{\left\|\left(\mathbf{x}_1^{k+1}, \ldots, \mathbf{x}_m^{k+1}\right) - \left(\mathbf{x}_1^{k}, \ldots, \mathbf{x}_m^{k}\right)\right\|^2}{\left\|\left(\mathbf{x}_1^{k}, \ldots, \mathbf{x}_m^{k}\right)\right\|^2 + 1} \leq \epsilon$$

where the step size for FISTA is set to be 0.1. The stop criterion of Algorithm 1 for the reformulation problem (3) is

$$\max\left\{\frac{\left\|\mathbf{x}^{k+1} - \mathbf{z}^{k+1}\right\|^2}{\max\left\{\left\|\mathbf{x}^{k+1}\right\|^2, \left\|\mathbf{z}^{k+1}\right\|^2\right\} + 1}, \frac{\left\|\mathbf{x}^{k+1} - \mathbf{x}^{k}\right\|^2}{\left\|\mathbf{x}^{k}\right\|^2 + 1}\right\} \leq \epsilon$$

where $\mathbf{x}^k = [\mathbf{x}_1^k, \ldots, \mathbf{x}_m^k]$ and $\mathbf{z}^k = [\mathbf{z}_1^k, \ldots, \mathbf{z}_m^k]$. The parameter β_i is set to be 100. It is unclear that whether this setting satisfies the second assumption, because it is difficult to estimate the Lipschitz constant L_i. But we still can guarantee the convergence of our algorithm under this setting.

For Algorithm 1, two variable selection schemes are used: 1) all the variables $\{\mathbf{x}_i\}$ are chosen to update and 2) $\lceil (m/2) \rceil$ variables are randomly chosen to update, while after T iterations, we will check the essentially cyclic update rule and force the unselected variables to update. We denote these two specific algorithms: 1) *Alg-1-full* and 2) *Alg-1-rand*. For all algorithms, we set the stopping criterion parameter $\epsilon = 10^{-4}$, $T = 2m$.

In Table I, we evaluate the efficiency of our algorithm framework (Algorithm 1), and compare it with FISTA for formulation (2) by using four nonconvex penalties (group-SCAD, group-MCP, matrix-SCAD, and matrix-MCP) on three real datasets. We compare the iteration number "Iter #," objective function value "Objective," constraint violation "Constraint," and computing time "Time," respectively.

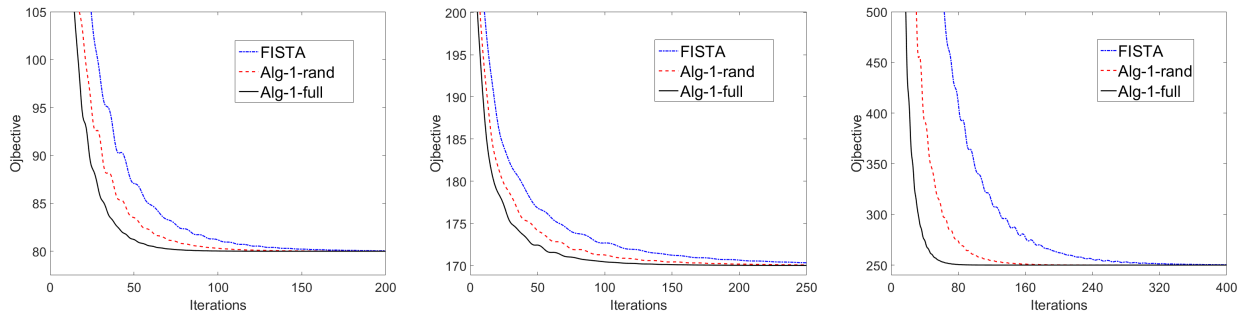


Fig. 1. Objective value with respect to the iteration number for logistic regression with group-SCAD penalty. (a) SARCOS. (b) SCHOOL. (c) E2006-tfidf.

TABLE I
COMPARISON ON LOGISTIC REGRESSION WITH FOUR NONCONVEX PENALTY CASES OF ALG-1-FULL, ALG-1-RAND, AND FISTA

Penalty	Dataset	Algorithm	Iter. #	Objective	Constraint	Time(s)
group-SCAD	SARCOS	FISTA	172	86.274	/	326.51
		Alg-1-full	69	86.198	2.79895e-05	120.74
		Alg-1-rand	96	86.204	4.17219e-05	191.25
	SCHOOL	FISTA	219	173.291	/	578.21
		Alg-1-full	104	172.973	3.26145e-05	209.35
		Alg-1-rand	127	173.216	4.27198e-05	286.14
	E2006-tfidf	FISTA	367	251.592	/	2634.19
		Alg-1-full	198	251.392	1.87159e-05	1273.43
		Alg-1-rand	227	251.501	2.77236e-05	1549.16
group-MCP	SARCOS	FISTA	189	73.192	/	376.19
		Alg-1-full	76	73.054	2.37415e-05	144.07
		Alg-1-rand	105	73.137	3.16273e-05	213.26
	SCHOOL	FISTA	234	151.293	/	709.14
		Alg-1-full	124	150.874	3.01725e-05	241.59
		Alg-1-rand	148	151.171	2.19754e-05	313.26
	E2006-tfidf	FISTA	391	227.195	/	2819.21
		Alg-1-full	226	226.771	1.37726e-05	1606.37
		Alg-1-rand	272	227.019	3.07138e-05	1927.56
matrix-SCAD	SARCOS	FISTA	156	103.291	/	431.27
		Alg-1-full	72	103.177	2.76731e-05	207.19
		Alg-1-rand	98	103.224	3.07752e-05	263.25
	SCHOOL	FISTA	179	219.376	/	723.19
		Alg-1-full	84	219.113	2.15142e-05	294.33
		Alg-1-rand	117	219.275	3.32994e-05	389.26
	E2006-tfidf	FISTA	317	433.257	/	4219.46
		Alg-1-full	182	432.774	3.18741e-05	2613.69
		Alg-1-rand	231	432.976	3.76221e-05	3117.57
matrix-MCP	SARCOS	FISTA	176	106.132	/	489.14
		Alg-1-full	80	106.029	1.97210e-05	237.26
		Alg-1-rand	106	106.122	2.18729e-05	317.04
	SCHOOL	FISTA	216	189.231	/	771.32
		Alg-1-full	142	189.017	3.18162e-05	329.51
		Alg-1-rand	161	189.069	3.29071e-05	426.44
	E2006-tfidf	FISTA	372	369.256	/	3316.57
		Alg-1-full	241	368.974	2.33561e-05	1817.32
		Alg-1-rand	293	369.175	2.86173e-05	2153.16

Table I shows that the proposed algorithm framework (Algorithm 1) has efficient performance for logistic regression with all of these four nonconvex penalty cases. Both Alg-1-full and Alg-1-rand can exceed FISTA from both the computing speed and calculation accuracy, which means less computing time and less objective function value, respectively. Indeed, the result proves the convergence of our method as well as the validity of reformulation (5). Moreover, the convergence theoretical result has been verified for the nonconvex problems. Alg-1-full performs better than Alg-1-rand because all of the variables $\{\mathbf{x}_i\}$ are updated in each iteration while only half of the variables are renewed. However, only half variables do not mean half convergence speed, and this implies that the asynchronous parallel actually accelerates the convergence of the proposed algorithm. Fig. 1 contains the objective value sequence with respect to the iteration number for logistic regression with group-SCAD penalty on three datasets. Here, we only present the case of group-SCAD penalty; as for the other three penalties, the comparison seems to be the same.

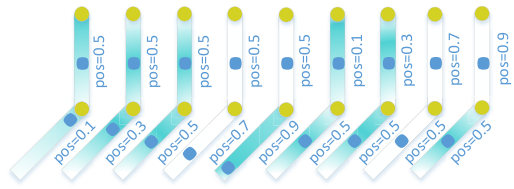


Fig. 2. Multitask environment Arcobot-LCP.

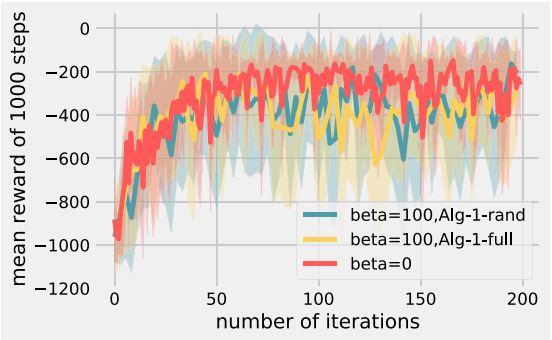


Fig. 3. Mean rewards of all three algorithms: Alg-1-rand (beta = 100, Alg-1-rand), Alg-1-full (beta = 100, Alg-1-full), and Baseline (beta = 0).

B. Multitask Reinforcement Learning

Furthermore, we consider the multitask reinforcement learning application, whose model can be included into the general formulation (1). Based on [41], we design a new multitask reinforcement learning model utilizing our proposed nonconvex optimization framework. Before presenting the concrete formulation, we describe the experiment environment as follows.

The Acrobot-v1 in the OpenAI open source Gym toolkit [42] is employed as the fundamental environment here. Acrobot is a 2-link pendulum with only the second joint actuated, while both links point downward initially. The goal is to

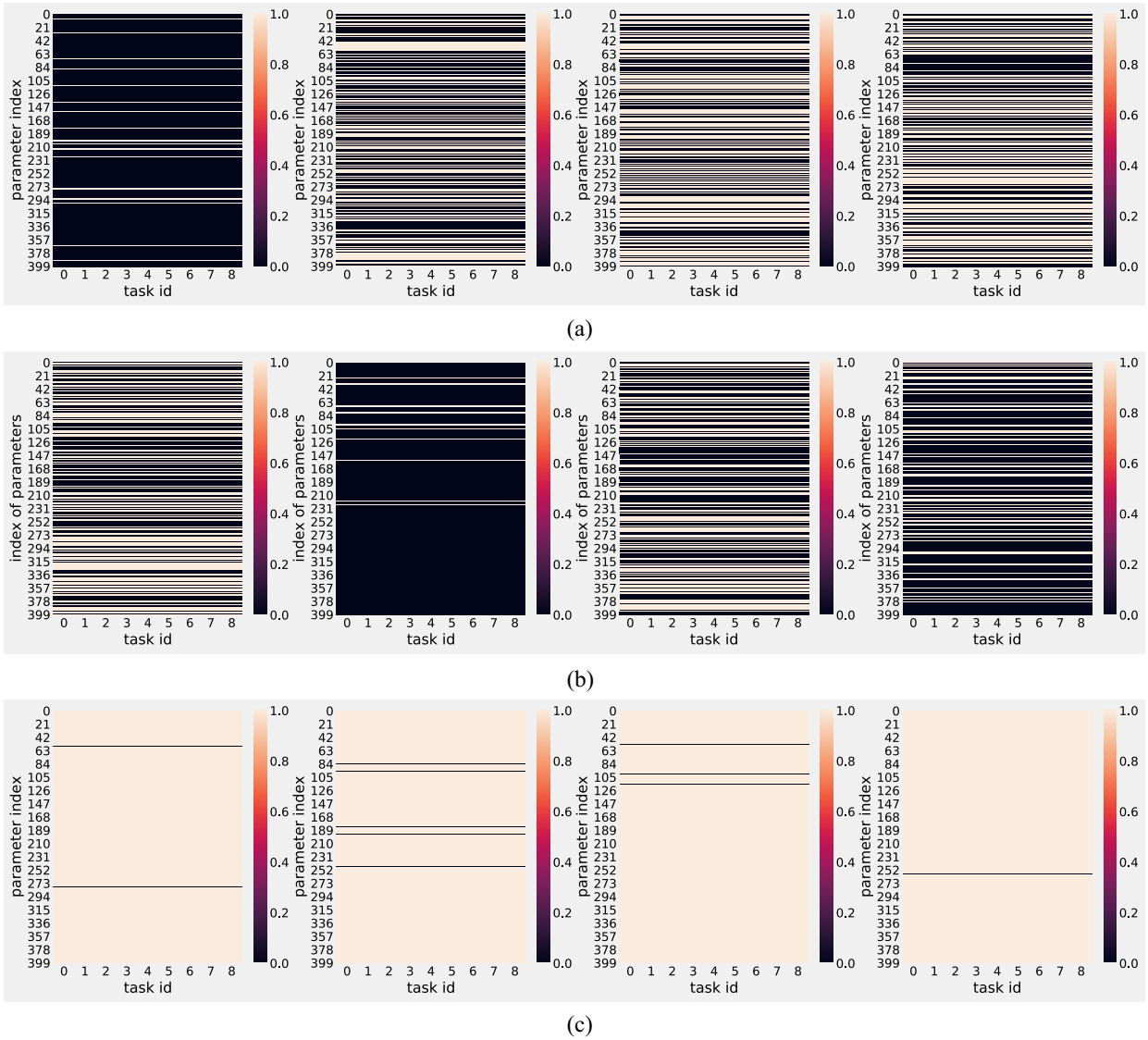


Fig. 4. Grouped lasso constraint satisfaction. (a) Alg-1-rand. (b) Alg-1-full. (c) Baseline.

swing the end effector at a height at least the length of one link above the baseline. Both links can swing freely and can pass by each other, which means they do not collide when they have the same angle.

The state space consists of the sine and cosine values of the two rotational joint angles θ_1 and θ_2, together with the joint angular velocities γ_1 and γ_2, that is, $[\cos(\theta_1), \sin(\theta_1), \cos(\theta_2), \sin(\theta_2), \gamma_1, \gamma_2]$. θ_2 denotes the angle between these two links. For instance, the zero angle of the first link corresponds to the first link pointing downward, while the zero angle of the second link corresponds to having the same angle between the two links. The action is either applying $+1$, 0 or -1 torque on the joint between the two pendulum links.

Our multitask environment is designed via modifying Acrobot-v1. Specifically, we consider different mass centers of the two links in the nine-task system instead of fixing the middle, and the new environment is called Arcobot-LCP (see Fig. 2). The detailed information of the mass centers can also be found in Fig. 2, where the blue dot in each link denotes their mass center.

The dimension of the state space of this environment is a total of 400. (See the hyperparameter setting section for more details.) In our model, the Q-network is introduced with the state vector as input, while the huge state space dimension will lead to poor training efficiency. Therefore, the sparse Q-network is adopted to adaptively select the states. We assume that Q-networks of all tasks share the same sparse structure, so that we add a group sparse penalty on all parameters.

For this specific structured multitask reinforcement learning problem, the related formulation (1) can be described as follows:

$$f_i(\mathbf{x}_i) = \frac{1}{N_i}\sum_{j=1}^{N_i} \Big\| r_i\big(s_{i,j}, a_{i,j}\big) + \gamma \max_{a'_{i,j}} Q_i\Big(s'_{i,j}, a'_{i,j}; \mathbf{x}_i\Big) - Q_i(s_{i,j}, a_{i,j}; \mathbf{x}_i)\Big\|_2^2$$

$$h(\mathbf{x}_1, \ldots, \mathbf{x}_m) = \lambda \|\mathbf{X}\|_{2,1}$$

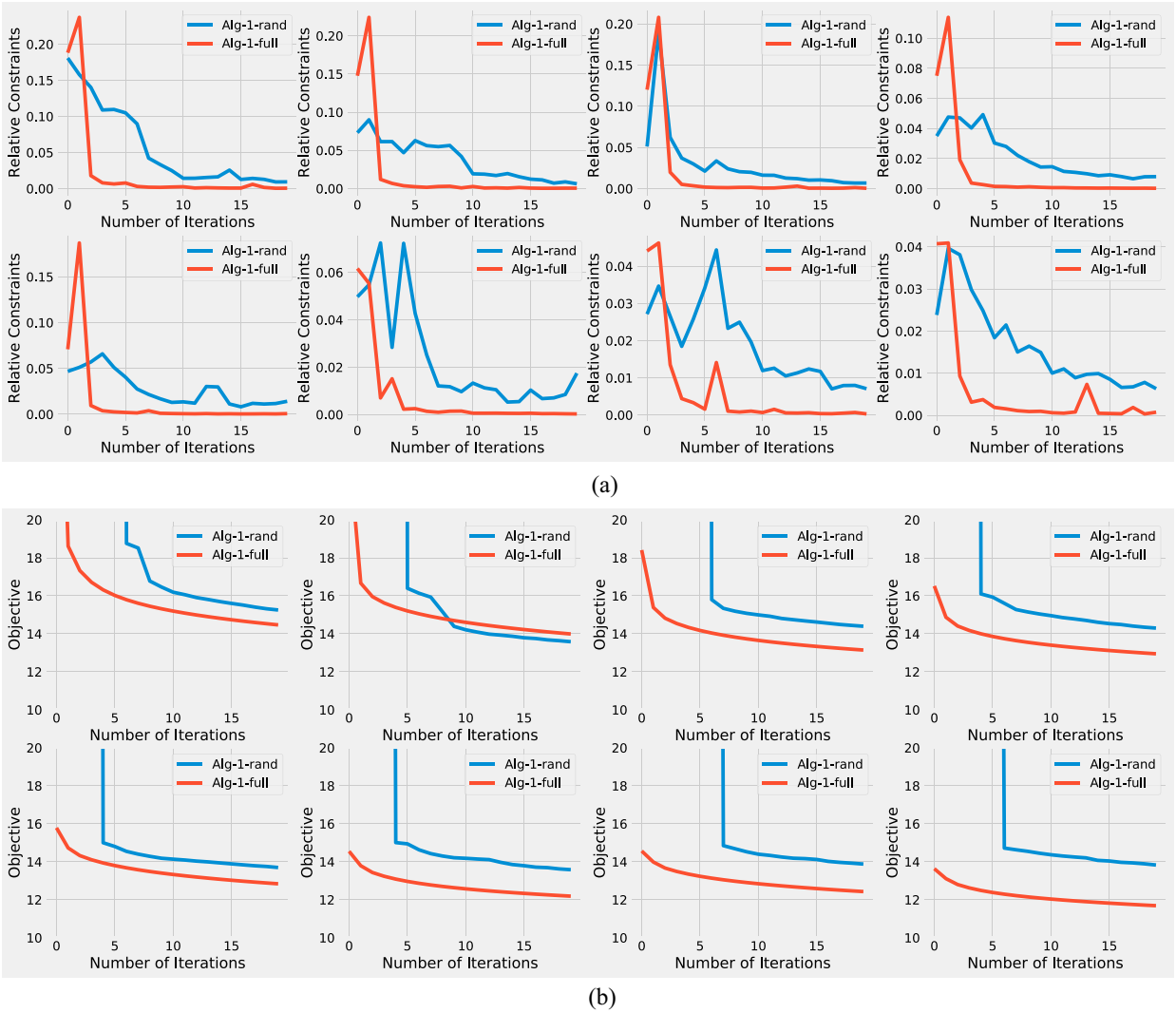


Fig. 5. (a) Consensus constraints violation and (b) objective function values.

where $\{s_{i,j}, a_{i,j}, s'_{i,j}\}_{j=1}^{N_i}$ is sampled from the experience replay buffer of task i, r_i is the reward function of the task i, and Q_i is the Q-value function of the task i. $Q_i = \sigma^{\mathrm{T}}(\phi(s_i, a_i)^{\mathrm{T}}\mathbf{x}_i^1)\mathbf{x}_i^2$ is a two-layer multiple layer perceptron, and $\mathbf{x}_i^1$ and $\mathbf{x}_i^2$ denote the first-layer parameters and the second-layer parameters separately. $\sigma(x) = \max(0, x)$ is the ReLU elementwise nonlinear function, and ϕ is a common feature extractor. $\mathbf{x}_i$ denotes the ith column of $\mathbf{X}$ and $\|\mathbf{X}\|_{2,1} = \sum_{i=1}^{m} \|\mathbf{x}_i\|_2$.

In this experiment, a two-layer multilayer perceptron with a ReLU nonlinear activation function was used as the Q-value function estimator. The number of neurons in the hidden layer is 32. The size of the experience replay buffer is set to 50 000, the number of samples per training is 1000 per task, and the data collected by each episode per task is 100 transitions. It is worth noting that we first use the RBF sampler (variant of [43]) with variances of 0.5, 1.0, 1.5, and 2.0 for the input data (6-D state vector) to be up to 400 dimensions; thus, the input dimension of the Q-value function is 400. We use the L-BFGS algorithm as the nonconvex function optimizer. The penalty coefficient of the consensus constraint is set to 100, and the penalty coefficient of the sparse constraint is set to 0.01. Due to the large number of parameters (since the action of the agents is discrete and each action corresponds to a Q-function estimator, so the total number of parameters for each task is 38 496), the number of loops of the ADMM algorithm is set to 20 in order to save training time.

We further choose the results of nine task independent training without sparse constraints as the benchmark results. Since the implementation of the related algorithm using ADMM in this article, the above benchmark algorithm can be implemented by setting the penalty coefficient of the consensus constraint of the ADMM algorithm to 0. In addition, we also compare the experimental results of Alg-1-rand and Alg-1-full. For the former, $\lceil (m/4) \rceil = 3$ variables are randomly chosen to update, while after $T = 20$ iterations, we will check the essentially cyclic update rule and force the unselected variables to update.

Fig. 3 shows a comparison of the average rewards of the three algorithms over the test set (1000 timesteps) as the number of training epochs increases. It can be seen from the figure that the performance of the three algorithms is basically the same. Fig. 4 shows the satisfaction of different algorithms for grouped lasso sparse constraints. Specifically, we select the Q-value function estimator corresponding to one action of the

agent, that is, the first-layer parameter of the two-layer perceptron, and the dimension is $\mathbb{R}^{400\times 32}$. The 32 hidden layer neurons can be regarded as 32 different implicit semantic embeddings for 400-D input data. Due to space limitations, we randomly select four embeddings for display. The dark portion of each row in the figure represents a parameter value of less than 0.001. It can be seen that the algorithm proposed in this article has the ability to achieve very sparse feature selection, and the selected features are shared between tasks. Combined with Fig. 3, the proposed algorithm can achieve the same performance as the benchmark algorithm based on very few features.

Fig. 5 shows the satisfaction of the consensus constraints and the value of the loss function for the Alg-1-full algorithm and the Alg-1-rand algorithm in solving the ADMM problem. These eight pictures are randomly selected from the training of 200 epochs of reinforcement learning algorithms. It can be seen from the figure that 20 cycles of the ADMM algorithm can ensure convergence.

V. Conclusion

We have proposed a distributed and parallel algorithmic framework for the nonconvex optimization problems, which can find wide applications in modern machine-learning problems. The algorithm framework is based on the popular ADMM technique, while it can handle a more challenging and general case that all terms in the objective function are nonconvex. Theoretical convergence of the proposed algorithm is established only with the assumption that parameter $\{\beta_i\}$ is large enough. The algorithm is evaluated on the popular MTL benchmarks with nonconvex penalties. In fact, our framework can incorporate many existing MTL formulations either in the presence of outlier tasks or not, and can routinely adapt them to the parallel and distributed computing setting. However, our algorithm framework still has some limitations: the combined regularization term should be discussed in detail with the purpose to raise the efficiency; from the perspective of theoretical analysis, the convergence rate should be further discussed. Furthermore, we will leave a more comprehensive empirical study for the parallel case and inexact case to the future work.

Appendix A
Discussions of SCAD-Type and MCP-Type Penalties

We first introduce two seed functions (which play the role as the absolute value function $|\cdot|$ for ℓ_1-norm).

1) *SCAD [36]*

$$g(|\theta|;\lambda,a)=\begin{cases}\lambda|\theta|, & |\theta|\le\lambda\\ \frac{-\theta^2+2a\lambda|\theta|-\lambda^2}{2(a-1)}, & \lambda<|\theta|\le a\lambda\\ \frac{(a+1)\lambda^2}{2}, & |\theta|>a\lambda\end{cases}$$

where $a>2$ and $\lambda>0$.

2) *MCP [37]*

$$g(|\theta|;\lambda,a)=\begin{cases}\lambda|\theta|-\frac{1}{2a}\theta^2, & |\theta|<a\lambda\\ \frac{\lambda^2 a}{2}, & |\theta|\ge a\lambda\end{cases}$$

where $a>0$ and $\lambda>0$.

The SCAD or MCP penalty for $\mathbf{x}\in\mathbb{R}^n$ is defined as $g(\mathbf{x})=\sum_{i=1}^n g(|\mathbf{x}_i|;\lambda,a)$. They can both result in an estimator with three important properties: 1) unbiasedness; 2) sparsity; and 3) continuity. This is an obvious advantage over the convex ℓ_1 penalty which usually leads to biased estimators and the concave ℓ_p penalty with $0\le p<1$ that often fails to satisfy the continuity condition. More algorithms can be developed based on the structure of SCAD and MCP. More discussions can be found in [44]. Based on these seed functions, we define four penalties: group-SCAD, group-MCP, matrix-SCAD, and matrix-MCP for $\mathbf{X}\in\mathbb{R}^{m\times n}$

$$\text{group-SCAD/group-MCP: } h(\mathbf{X})=\sum_{j=1}^{n} g\big(\|\mathbf{X}[:,j]\|_2;\lambda,a\big)$$

$$\text{matrix-SCAD/matrix-MCP: } h(\mathbf{X})=\sum_{j=1}^{r} g\big(\sigma_j;\lambda,a\big)$$

where $\{\sigma_i\}$ is the singular values of matrix $\mathbf{X}$.

By their above definition, it is obvious that group-SCAD and group-MCP have similar structures with Group-Lasso, while matrix-SCAD and matrix-MCP have similar structures with nuclear norm. Due to the oracle properties of SCAD and MCP, the new four penalties also have the oracle properties, that is, unbiasedness, sparsity, and continuity. Hence, group-SCAD and group-MCP have advantages theoretically to Group-Lasso regarding the group sparsity, while matrix-SCAD and matrix-MCP can perform better than nuclear norm for the low-rank result.

However, an important issue is that $g(|\theta|;\lambda,a)$ is not convex with respect to θ; thus, classical algorithmic and theoretical results cannot be directly applied. Fortunately, they have two special properties as which can be used to design the algorithm with sound theoretical results. Recalling the proximal operator of both SCAD and MCP penalties, we can obtain two analytical solutions for SCAD and MCP, respectively, for $\tau<a-1$ and $\tau<a$ accordingly

$$\Big(\text{prox}_g^{\tau}(\mathbf{z})\Big)_i=\begin{cases}\text{sign}(z_i)(|\mathbf{z}_i|-\tau\lambda)_+, & |\mathbf{z}_i|\le(1+\tau)\lambda\\ \frac{(a-1)z_i-\text{sign}(\mathbf{z}_i)\tau a\lambda}{a-1-\tau}, & (1+\tau)\lambda<|\mathbf{z}_i|\le a\lambda\\ z_i, & a\lambda<|z_i|\end{cases}$$

$$\Big(\text{prox}_g^{\tau}(\mathbf{z})\Big)_i=\begin{cases}\text{sign}(z_i)\Big(\frac{a}{a-\tau}|\mathbf{z}_i|-\frac{\tau a\lambda}{a-\tau}\Big)_+, & |\mathbf{z}_i|\le a\lambda\\ \mathbf{z}_i, & |\mathbf{z}_i|>a\lambda.\end{cases}$$

This property is similar to the ℓ_1-norm, whose proximal operator is also called shrinkage. When we compute $\{\mathbf{z}_i^{k+1}\}$ in our algorithm framework, these proximal operators can be efficiently used to solve the associated subproblems.

Appendix B
Proof of Theorem 1

Assumptions:

1) There exists a positive constant $L_i>0$ such that

$$\|\nabla f_i(\mathbf{x}_i)-\nabla f_i(\mathbf{y}_i)\|\le L_i\|\mathbf{x}_i-\mathbf{y}_i\|,\ \forall\mathbf{x}_i,\mathbf{y}_i,\ \forall i. \quad (16)$$

2) The parameter β_i is chosen large enough such that the x_i subproblem (9) is strongly convex with modulus γ_i, $\beta_i\gamma_i\ge 2L_i^2$ and $\beta_i\ge L_i$ for all i.

3) $g(\mathbf{x})$ is bounded from below, that is, $\underline{g} = \min g(\mathbf{x}) > -\infty$.

Theorem 2: Suppose the above assumptions hold and let $\{\{\mathbf{x}_i^k\}, \{\mathbf{z}_i^k\}, \{\boldsymbol{\lambda}_i^k\}\}$ be the sequence generated by Algorithm 1, then we have the following:

1) $\lim_{k\to\infty} \|\mathbf{x}_i^k - \mathbf{z}_i^k\| = 0, \ \forall i$;
2) let $(\{\mathbf{x}_i^\star\}, \{\mathbf{z}_i^\star\}, \{\boldsymbol{\lambda}_i^\star\})$ denote any limit point of the sequence $\{\{\mathbf{x}_i^k\}, \{\mathbf{z}_i^k\}, \{\lambda_i^k\}\}$. Then, the following statement is true:

$$0 = \nabla f_i\left(\mathbf{x}_i^\star\right) - \boldsymbol{\lambda}_i^\star, \quad \forall i \tag{17a}$$

$$\left(\mathbf{z}_1^\star, \ldots, \mathbf{z}_m^\star\right) \in \arg\min \ h(\{\mathbf{z}_i\}) - \sum_{i=1}^{m} \left\langle \boldsymbol{\lambda}_i^\star, \mathbf{x}_i^\star - \mathbf{z}_i \right\rangle \tag{17b}$$

$$\mathbf{x}_i^\star = \mathbf{z}_i^\star, \quad \forall i \tag{17c}$$

which means any limit point of Algorithm 1 is a stationary solution of problem (5).

Before proving this theorem, we first prove some lemmas.

Lemma 1: For Algorithm 1, we have the following sufficient decrease with respect to the augmented Lagrangian function value:

$$\mathcal{L}\left(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^{k+1}\right) - \mathcal{L}\left(\mathbf{x}_i^k, \mathbf{z}_i^k, \boldsymbol{\lambda}_i^k\right) \leq \sum_{i\neq 0, i\in\mathcal{C}^{k+1}} \left(\frac{L_i^2}{\rho_i} - \frac{\gamma_i}{2}\right) \left\|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\right\|^2. \tag{18}$$

Proof: We first split the sufficient decrease with respect to the augmented Lagrangian function value by

$$\begin{aligned}\mathcal{L}\left(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^{k+1}\right) - \mathcal{L}\left(\mathbf{x}_i^k, \mathbf{z}_i^k, \boldsymbol{\lambda}_i^k\right) &= \left[\mathcal{L}\left(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^{k+1}\right) - \mathcal{L}\left(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^k\right)\right] \\ &\quad + \left[\mathcal{L}\left(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^k\right) - \mathcal{L}\left(\mathbf{x}_i^k, \mathbf{z}_i^k, \boldsymbol{\lambda}_i^k\right)\right].\end{aligned} \tag{19}$$

Further, we will try to bound these two terms in (19) and the first term can be bounded by

$$\begin{aligned}\mathcal{L}\left(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^{k+1}\right) - \mathcal{L}\left(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^k\right) &= \sum_{i=1}^{m} \left\langle \boldsymbol{\lambda}_i^k - \boldsymbol{\lambda}_i^{k+1}, \mathbf{x}_i^{k+1} - \mathbf{z}_i^{k+1} \right\rangle \\ &= \sum_{i\neq 0, i\in\mathcal{C}^{k+1}} \frac{1}{\beta_i} \left\|\boldsymbol{\lambda}_i^{k+1} - \boldsymbol{\lambda}_i^k\right\|^2\end{aligned} \tag{20}$$

where the last equation is obtained through (10) and the fact that $\boldsymbol{\lambda}_i^{k+1} = \boldsymbol{\lambda}_i^k$ for all variable block $\mathbf{x}_i$ has not been updated. Then, we consider the second term in (19), which can be bounded by

$$\begin{aligned}&\mathcal{L}\left(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^k\right) - \mathcal{L}\left(\mathbf{x}_i^k, \mathbf{z}_i^k, \boldsymbol{\lambda}_i^k\right) \\ &= \left[\mathcal{L}\left(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^k\right) - \mathcal{L}\left(\mathbf{x}_i^k, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^k\right)\right] + \left[\mathcal{L}\left(\mathbf{x}_i^k, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^k\right) - \mathcal{L}\left(\mathbf{x}_i^k, \mathbf{z}_i^k, \boldsymbol{\lambda}_i^k\right)\right] \\ &\leq \sum_{i=1}^{m} \left[\left\langle \nabla_{\mathbf{x}_i} \mathcal{L}\left(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^k\right), \mathbf{x}_i^{k+1} - \mathbf{x}_i^k \right\rangle - \frac{\gamma_i}{2} \left\|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\right\|^2\right] + \left[\mathcal{L}\left(\mathbf{x}_i^k, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^k\right) - \mathcal{L}\left(\mathbf{x}_i^k, \mathbf{z}_i^k, \boldsymbol{\lambda}_i^k\right)\right] \\ &\leq \sum_{i\neq 0, i\in\mathcal{C}^{k+1}} \left[\left\langle \nabla_{\mathbf{x}_i} \mathcal{L}\left(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^k\right), \mathbf{x}_i^{k+1} - \mathbf{x}_i^k \right\rangle - \frac{\gamma_i}{2} \left\|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\right\|^2\right] \\ &\leq - \sum_{i\neq 0, i\in\mathcal{C}^{k+1}} \frac{\gamma_i}{2} \left\|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\right\|^2\end{aligned} \tag{21}$$

where the first inequality is obtained through that $\mathcal{L}(\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\lambda}_i)$ is strongly convex with respect to $\mathbf{x}_i$; the second inequality is obtained through that $\{\mathbf{z}_i^{k+1}\}$ are the guaranteed global minimizer of subproblem (9), $\mathbf{x}_i^{k+1} = \mathbf{x}_i^k$ for $i \notin \mathcal{C}^{k+1}$; the third inequality is obtained through the optimality condition of subproblem (9) with respect to each $\mathbf{x}_i$. Further, by adding (20) and (21), we have

$$\begin{aligned}&\mathcal{L}\left(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^{k+1}\right) - \mathcal{L}\left(\mathbf{x}_i^k, \mathbf{z}_i^k, \boldsymbol{\lambda}_i^k\right) \\ &= \left[\mathcal{L}\left(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^{k+1}\right) - \mathcal{L}\left(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^k\right)\right] + \left[\mathcal{L}\left(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^k\right) - \mathcal{L}\left(\mathbf{x}_i^k, \mathbf{z}_i^k, \boldsymbol{\lambda}_i^k\right)\right] \\ &\leq \sum_{i\neq 0, i\in\mathcal{C}^{k+1}} \frac{1}{\beta_i} \left\|\boldsymbol{\lambda}_i^{k+1} - \boldsymbol{\lambda}_i^k\right\|^2 - \sum_{i\neq 0, i\in\mathcal{C}^{k+1}} \frac{\gamma_i}{2} \left\|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\right\|^2 \\ &\leq \sum_{i\neq 0, i\in\mathcal{C}^{k+1}} \left(\frac{L_i^2}{\beta_i} - \frac{\gamma_i}{2}\right) \left\|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\right\|^2\end{aligned} \tag{22}$$

where the last inequality is obtained through the fact that

$$\begin{cases} \text{optimality condition of (9)} \Rightarrow \\ \qquad \nabla f_i\left(\mathbf{x}_i^{k+1}\right) + \beta_i\left(\mathbf{x}_i^{k+1} - \mathbf{z}_i^{k+1} - \frac{\boldsymbol{\lambda}_i^k}{\beta_i}\right) = 0 \\ (10) \Rightarrow \beta_i\left(\mathbf{x}_i^{k+1} - \mathbf{z}_i^{k+1} - \frac{\boldsymbol{\lambda}_i^k}{\beta_i}\right) = -\boldsymbol{\lambda}_i^{k+1} \end{cases}$$

then we obtain $\boldsymbol{\lambda}_i^{k+1} = \nabla f_i(\mathbf{x}_i^{k+1})$ and ∇f_i is Lipschitz continuous with constant L_i. Equation (22) exactly implies (18). ∎

It is obvious that (18) implies that if the assumption $\beta_i\gamma_i \geq 2L_i^2$ holds, then the augmented Lagrangian function value will always decrease. This is the key property that we need to prove in order to guarantee the convergence result of Algorithm 1.

Lemma 2: For Algorithm 1, the following limit exists and is lower bounded by $\underline{g}$, that is, $\lim_{k\to\infty} \mathcal{L}(\mathbf{x}_i^k, \mathbf{z}_i^k, \boldsymbol{\lambda}_i^k) \geq \underline{g}$.

Proof: Recall the augmented Lagrangian function and express into

$$\begin{aligned}&\mathcal{L}\left(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^{k+1}\right) \\ &= \sum_{i=1}^{m} f_i\left(\mathbf{x}_i^{k+1}\right) + h\left(\mathbf{z}_1^{k+1}, \ldots, \mathbf{z}_i^{k+1}\right) - \sum_{i=1}^{m} \left[\left\langle \boldsymbol{\lambda}_i^{k+1}, \mathbf{x}_i^{k+1} - \mathbf{z}_i^{k+1} \right\rangle - \frac{\beta_i}{2} \left\|\mathbf{x}_i^{k+1} - \mathbf{z}_i^{k+1}\right\|^2\right] \\ &= h\left(\mathbf{z}_1^{k+1}, \ldots, \mathbf{z}_i^{k+1}\right) + \sum_{i=1}^{m} \left[f_i\left(\mathbf{x}_i^{k+1}\right) + \left\langle \nabla f_i\left(\mathbf{x}_i^{k+1}\right), \mathbf{z}_i^{k+1} - \mathbf{x}_i^{k+1} \right\rangle + \frac{\beta_i}{2} \left\|\mathbf{x}_i^{k+1} - \mathbf{z}_i^{k+1}\right\|^2\right] \\ &\geq h\left(\mathbf{z}_1^{k+1}, \ldots, \mathbf{z}_i^{k+1}\right) + \sum_{i=1}^{m} f_i\left(\mathbf{z}_i^{k+1}\right) = f\left(\mathbf{z}_1^{k+1}, \ldots, \mathbf{z}_i^{k+1}\right)\end{aligned} \tag{23}$$

where the last inequality is obtained through that the Lipschitz continuity of ∇f_i and $\beta_i \geq L_i$ for all i. Recalling the third assumption, we can guarantee that $\mathcal{L}(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^{k+1})$ is lower bounded. This, combined with (18), implies that $\mathcal{L}(\mathbf{x}_i^{k+1}, \mathbf{z}_i^{k+1}, \boldsymbol{\lambda}_i^{k+1})$ is monotonically decreasing and is convergent. This completes the proof of this lemma. ∎

This property ensures the existence of the limit of the augmented Lagrangian function value sequence which will help prove the convergence of Algorithm 1. Next, we move to the proof of Theorem 2.

Proof: At first, with the essentially cyclic update rule, we have

$$\begin{aligned}
&\mathcal{L}\left(\mathbf{x}_i^{k+T}, \mathbf{z}_i^{k+T}, \boldsymbol{\lambda}_i^{k+T}\right) - \mathcal{L}\left(\mathbf{x}_i^{k}, \mathbf{z}_i^{k}, \boldsymbol{\lambda}_i^{k}\right) \\
&\leq \sum_{t=1}^{T}\left\{\sum_{i\neq 0, i\in\mathcal{C}^{k+1}}\left(\frac{L_i^2}{\rho_i}-\frac{\gamma_i}{2}\right)\left\|\mathbf{x}_i^{k+t}-\mathbf{x}_i^{k+t-1}\right\|^2\right. \\
&\qquad \left.-\frac{\gamma}{2}\sum_{i=1}^{m}\left(\left\|\mathbf{z}_i^{k+t}-\mathbf{z}_i^{k+t-1}\right\|^2\right)\right\} \\
&= \sum_{t=1}^{T}\sum_{i=1}^{m}\left[\left(\frac{L_i^2}{\rho_i}-\frac{\gamma_i}{2}\right)\left\|\mathbf{x}_i^{k+t}-\mathbf{x}_i^{k+t-1}\right\|^2\right]
\end{aligned}$$

and by using the fact that each index i has been updated at least once, we have that

$$\left\|\mathbf{x}_i^{k+1}-\mathbf{x}_i^{k(i)}\right\| \to 0, \quad \forall i \tag{24}$$

where $k(i)$ and $k(0)$ denote the last updated iterative numbers before k+1 for $\mathbf{x}_i$ and $(\mathbf{z}_1, \ldots, \mathbf{z}_m)$, respectively. Further, we have

$$\begin{aligned}
\left\|\boldsymbol{\lambda}_i^{k+1}-\boldsymbol{\lambda}_i^{k(i)}\right\| &= \left\|\nabla f_i\left(\mathbf{x}_i^{k+1}\right)-\nabla f_i\left(\mathbf{x}_i^{k(i)}\right)\right\| \\
&\leq L_i\left\|\mathbf{x}_i^{k+1}-\mathbf{x}_i^{k(i)}\right\| \to 0, \quad \forall i
\end{aligned}$$

which further implies $\|\mathbf{x}_i^{k+1}-\mathbf{z}_i^{k+1}\| \to 0$. This completes the first part of this theorem.

Next, we show the second part of this theorem. Recall the optimality conditions with respect to $\mathbf{x}_i$ and $(\mathbf{z}_1, \ldots, \mathbf{z}_m)$, respectively, as follows:

$$\begin{cases}
\nabla f_i\left(\mathbf{x}_i^{k+1}\right)+\beta_i\left(\mathbf{x}_i^{k+1}-\mathbf{z}_i^{k+1}-\dfrac{\boldsymbol{\lambda}_i^k}{\beta_i}\right)=0, \ i\neq 0, i\in\mathcal{C}^{k+1} \\
h(\mathbf{z}_1,\ldots,\mathbf{z}_m)+\sum\limits_{i=1}^{m}\dfrac{\beta_i}{2}\left\|\mathbf{z}_i-\left(\mathbf{x}_i^k-\dfrac{\boldsymbol{\lambda}_i^k}{\beta_i}\right)\right\|^2 \\
\quad \geq h\left(\mathbf{z}_1^{k+1},\ldots,\mathbf{z}_m^{k+1}\right)+\sum\limits_{i=1}^{m}\dfrac{\beta_i}{2}\left\|\mathbf{z}_i^{k+1}-\left(\mathbf{x}_i^k-\dfrac{\boldsymbol{\lambda}_i^k}{\beta_i}\right)\right\|^2 \\
\quad \forall \mathbf{z}_i, 0\in\mathcal{C}^{k+1}.
\end{cases}$$

The first inequality is equivalent with

$$\nabla f_i\left(\mathbf{x}_i^{k+1}\right)-\boldsymbol{\lambda}_i^k+\beta_i\left(\mathbf{x}_i^{k+1}-\mathbf{z}_i^{k+1}\right)=0, \ i\neq 0, i\in\mathcal{C}^{k+1}$$

and the second inequality implies that (convexity without h)

$$\begin{aligned}
&h(\mathbf{z}_1,\ldots,\mathbf{z}_m)-h\left(\mathbf{z}_1^{k+1},\ldots,\mathbf{z}_m^{k+1}\right) \\
&\quad+\sum_{i=1}^{m}\left\langle\beta_i\left[\mathbf{z}_i^{k+1}-\left(\mathbf{x}_i^k-\frac{\boldsymbol{\lambda}_i^k}{\beta_i}\right)\right], \mathbf{z}_i-\mathbf{z}_i^{k+1}\right\rangle\geq 0 \\
&\forall \mathbf{z}_i, 0\in\mathcal{C}^{k+1}.
\end{aligned} \tag{25}$$

Using the definition of the essentially cyclic update rule, we have that for all i

$$\nabla f_i\left(\mathbf{x}_i^{k(i)}\right)-\boldsymbol{\lambda}_i^{k(i)}=0, \ \forall i\neq 0, \ \text{for some } k(i)\in[k,k+T],$$

$$\begin{aligned}
&h(\mathbf{z}_1,\ldots,\mathbf{z}_m)-h\left(\mathbf{z}_1^{k(0)},\ldots,\mathbf{z}_m^{k(0)}\right) \\
&\quad+\sum_{i=1}^{m}\left\langle\boldsymbol{\lambda}_i^{k(0)+1}+\beta_i\left(\mathbf{x}_i^{k(0)}-\mathbf{x}_i^{k(0)-1}\right), \mathbf{z}_i-\mathbf{z}_i^{k(0)}\right\rangle
\end{aligned} \tag{26}$$

$$\geq 0, \ \forall \mathbf{z}_i, \ \text{for some } k(0)\in[k,k+T] \tag{27}$$

and we also have (24). Using this result, taking limit for (26) and (27) and using the fact that $\forall i$

$$\left\|\mathbf{x}_i^{k+1}-\mathbf{x}_i^k\right\|\to 0, \quad \mathbf{x}_i^{k+1}\to\mathbf{x}_i^\star, \quad \mathbf{z}_i^{k+1}\to\mathbf{z}_i^\star, \quad \boldsymbol{\lambda}_i^{k+1}\to\boldsymbol{\lambda}_i^\star.$$

Recall (26) and by taking the limitation, we can obtain that for all i

$$\nabla f_i\left(\mathbf{x}_i^\star\right)-\boldsymbol{\lambda}_i^\star=0$$

which implies (17a). Further, for (27), we can obtain that for all $\mathbf{z}_i$

$$h(\mathbf{z}_1,\ldots,\mathbf{z}_m)-h\left(\mathbf{z}_1^\star,\ldots,\mathbf{z}_m^\star\right)+\sum_{i=1}^{m}\left\langle\boldsymbol{\lambda}_i^\star, \mathbf{z}_i-\mathbf{x}_i^\star\right\rangle\geq 0$$

which can be considered as the optimality condition of (17b) and, as a result, we obtain (17b). Further, (17c) can be obtained due to the fact that $\|\boldsymbol{\lambda}_i^{k+1}-\boldsymbol{\lambda}_i^k\|\to 0$. This completes the second part of this theorem. ∎

REFERENCES

[1] Y. Li, J. Wang, J. Ye, and C. K. Reddy, "A multi-task learning formulation for survival analysis," in *Proc. ACM SIGKDD*, 2016, pp. 1715–1724.

[2] Y. Wang, D. P. Wipf, Q. Ling, W. Chen, and I. J. Wassell, "Multi-task learning for subspace segmentation," in *Proc. ICML*, 2015, pp. 1209–1217.

[3] J. Pu, Y.-G. Jiang, J. Wang, and X. Xue, "Which looks like which: Exploring inter-class relationships in fine-grained visual categorization," in *Proc. ECCV*, 2014, pp. 425–440.

[4] N. S. Rao, C. R. Cox, R. D. Nowak, and T. T. Rogers, "Sparse overlapping sets lasso for multitask learning and its application to fMRI analysis," in *Proc. NeurIPS*, 2013, pp. 2202–2210.

[5] A. Argyriou, T. Evgeniou, and M. Pontil, "Convex multi-task feature learning," *Mach. Learn.*, vol. 73, no. 3, pp. 243–272, 2008.

[6] T. K. Pong, P. Tseng, S. Ji, and J. Ye, "Trace norm regularization: Reformulations, algorithms, and multi-task learning," *SIAM J. Optim.*, vol. 20, no. 6, pp. 3465–3489, 2010.

[7] J. Liu, S. Ji, and J. Ye, "Multi-task feature learning via efficient $\ell_{2,1}$-norm minimization," in *Proc. UAI*, 2009, pp. 339–348.

[8] J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert, "A new approach to collaborative filtering: Operator estimation with spectral regularization," *J. Mach. Learn. Res.*, vol. 10, pp. 803–826, Mar. 2009.

[9] P. Gong, C. Zhang, Z. Lu, J. Huang, and J. Ye, *GIST: General Iterative Shrinkage and Thresholding for Non-Convex Sparse Learning*, Tsinghua Univ., Beijing, China, Mar. 2013.

[10] C. Lu, J. Tang, S. Yan, and Z. Lin, "Nonconvex nonsmooth low rank minimization via iteratively reweighted nuclear norm," *IEEE Trans. Image Process.*, vol. 25, no. 2, pp. 829–839, Feb. 2016.

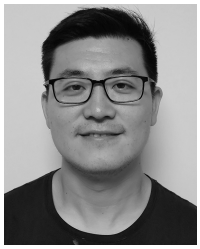
[11] G. Swirszcz and A. C. Lozano, "Multi-level lasso for sparse multi-task regression," in *Proc. ICML*, 2012, pp. 361–368.

[12] A. Jalali, S. Sanghavi, C. Ruan, and P. K. Ravikumar, "A dirty model for multi-task learning," in *Proc. NeurIPS*, 2010, pp. 964–972.

[13] J. Chen, J. Zhou, and J. Ye, "Integrating low-rank and group-sparse structures for robust MTL," in *Proc. ACM SIGKDD*, 2011, pp. 42–50.

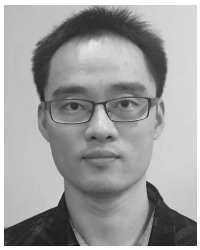
[14] P. Gong, J. Ye, and C. Zhang, "Robust multi-task feature learning," in *Proc. ACM SIGKDD*, 2012, pp. 895–903.

[15] P. Tseng, "Approximation accuracy, gradient methods, and error bound for structured convex optimization," *Math. Program.*, vol. 125, no. 2, pp. 263–295, 2010.
[16] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.
[17] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87. New York, NY, USA: Springer, 2013. [Online]. Available: https://www.springer.com/gp/book/9781402075537
[18] M. W. Schmidt, N. L. Roux, and F. R. Bach, "Convergence rates of inexact proximal-gradient methods for convex optimization," in *Proc. NeurIPS*, 2011, pp. 1458–1466.
[19] S. Tao, D. Boley, and S. Zhang, "Local linear convergence of ISTA and FISTA on the LASSO problem," *SIAM J. Optim.*, vol. 26, no. 1, pp. 313–336, 2016.
[20] H. Attouch, J. Bolte, and B. F. Svaiter, "Convergence of descent methods for semi-algebraic and tame problems: Proximal algorithms, forward-backward splitting, and regularized Gauss–Seidel methods," *Math. Program.*, vol. 137, nos. 1–2, pp. 91–129, 2013.
[21] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Math. Program.*, vol. 146, nos. 1–2, pp. 459–494, 2014.
[22] P. Ochs, Y. Chen, T. Brox, and T. Pock, "iPiano: Inertial proximal algorithm for nonconvex optimization," *SIAM J. Imag. Sci.*, vol. 7, no. 2, pp. 1388–1419, 2014.
[23] Y. Xu and W. Yin, "A globally convergent algorithm for nonconvex optimization based on block coordinate update," *J. Sci. Comput.*, vol. 72, no. 2, pp. 700–734, 2017.
[24] P. Gong, C. Zhang, Z. Lu, J. Huang, and J. Ye, "A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems," in *Proc. ICML*, 2013, pp. 37–45.
[25] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.
[26] R. Glowinski and A. Marroco, "Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problémes de Dirichlet non linéaires," *ESAIM Math. Model. Numer. Anal.*, vol. 9, no. R2, pp. 41–76, 1975.
[27] J. Eckstein and W. Yao, "Understanding the convergence of the ADMM: Theoretical and computational perspectives," *Pac. J. Optim.*, vol. 11, pp. 619–644, Oct. 2015.
[28] B. Ames and M. Hong, "Alternating direction method of multipliers for penalized zero-variance discriminant analysis," *Comput. Optim. Appl.*, vol. 64, no. 3, pp. 725–754, 2016.
[29] Z. Wen, C. Yang, X. Liu, and S. Marchesini, "Alternating direction methods for classical and ptychographic phase retrieval," *Inverse Problems*, vol. 28, no. 11, 2012, Art. no. 115010.
[30] R. Zhang and J. T. Kwok, "Asynchronous distributed ADMM for consensus optimization," in *Proc. ICML*, 2014, pp. 1701–1709.
[31] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM J. Optim.*, vol. 26, no. 1, pp. 337–364, 2016.
[32] M. Hong, X. Wang, M. Razaviyayn, and Z.-Q. Luo, "Iteration complexity analysis of block coordinate descent methods," *Math. Program.*, vol. 163, nos. 1–2, pp. 85–114, 2017.
[33] H. Yue, Q. Yang, X. Wang, and X. Yuan, "Implementing the ADMM to big datasets: A case study of LASSO," *SIAM J. Sci. Comput.*, vol. 40, no. 5, pp. A3121–A3156, 2018.
[34] S. Xiang, X. Shen, and J. Ye, "Efficient sparse group feature selection via nonconvex optimization," in *Proc. ICML*, 2013, pp. 284–292.
[35] Q. Yao, J. T. Kwok, and W. Zhong, "Fast low-rank matrix learning with nonconvex regularization," in *Proc. ICDM*, 2015, pp. 539–548.
[36] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *J. Amer. Stat. Assoc.*, vol. 96, no. 456, pp. 1348–1360, 2001.
[37] C.-H. Zhang, "Nearly unbiased variable selection under minimax concave penalty," *Ann. Stat.*, vol. 38, no. 2, pp. 894–942, 2010.
[38] S. Vijayakumar and S. Schaal, "LWPR: An $O(n)$ algorithm for incremental real time learning in high dimensional space," in *Proc. ICML*, 2000, pp. 288–293.
[39] H. Goldstein, "Multilevel modelling of survey data," *J. Roy. Stat. Soc. D Stat.*, vol. 40, no. 2, pp. 235–244, 1991.
[40] S. Kogan, D. Levin, B. R. Routledge, J. S. Sagi, and N. A. Smith, "Predicting risk from financial reports with regression," in *Proc. NAACL*, 2009, pp. 272–280.
[41] D. Calandriello, A. Lazaric, and M. Restelli, "Sparse multi-task reinforcement learning," in *Proc. NeurIPS*, 2014, pp. 819–827.
[42] G. Brockman *et al.* (2016). *OpenAI Gym*. [Online]. Available: https://arxiv.org/abs/1606.01540
[43] A. Rahimi and B. Recht, "Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning," in *Proc. NeurIPS*, 2009, pp. 1313–1320.
[44] L. Shen, B. W. Suter, and E. E. Tripp, "Structured sparsity promoting functions," *J. Optim. Theory Appl.*, vol. 183, no. 2, pp. 386–421, 2019.



Xiangfeng Wang received the B.S. degree in mathematics and applied mathematics and the Ph.D. degree in computational mathematics from Nanjing University, Nanjing, China, in 2009 and 2014, respectively.

He is currently an Assistant Professor with the School of Computer Science and Technology and the MOE Key Laboratory for Advanced Theory and Application in Statistics and Data Science, East China Normal University, Shanghai, China, and the Shanghai Institute of Intelligent Science and Technology, Tongji University, Shanghai. His research interests lie in the areas of large-scale optimization and applications on machine learning.



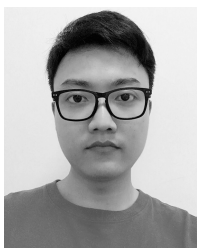
Junchi Yan (M'10) received the Ph.D. degree from the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China.

He has been a Senior Research Staff Member and a Principal Scientist for industrial vision with IBM Research, Yorktown Heights, NY, USA, since April 2011. He is currently an Associate Professor with the Department of Computer Science and Engineering and the MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University. His research interests are machine learning and visual computing.

Dr. Yan received the ACM China Doctoral Dissertation Nomination Award and China Computer Federation Doctoral Dissertation Award. He serves as an Associate Editor for IEEE ACCESS, a Managing Guest Editor for *Pattern Recognition Letters*, and on the executive board of ACM China Multimedia Chapter.

Bo Jin received the B.S. degree in electronic engineering and the Ph.D. degree in control theory and control engineering from Shanghai Jiao Tong University, Shanghai, China, in 2004 and 2014, respectively.

He is currently an Assistant Professor with the School of Computer Science and Technology and the MOE Key Laboratory for Advanced Theory and Application in Statistics and Data Science, East China Normal University, Shanghai, and the Shanghai Institute of Intelligent Science and Technology, Tongji University, Shanghai. His major research interests include machine learning, online learning, reinforcement learning, computer vision, and their applications.



Wenhao Li was born in Jiangxi, China, in 1995. He received the bachelor's degree in engineering from Lanzhou University, Lanzhou, China, in 2016, and the master's degree in engineering from East China Normal University, Shanghai, China, in 2019. He is currently pursuing the Ph.D. degree with East China Normal University and Tongji University, Shanghai.

His main research direction is deep machine learning and multiagent reinforcement learning.