A Novel Distributed ADMM Method in Large-Scale Applications

Heric Tsang[1]
[1] Zhengzhou Institute, China University of Geosciences (Beijing), Zhengzhou, China.

ABSTRACT Stochastic alternating direction method of multiplier (ADMM) has shown great promise in distributed environments, and improvements in algorithmic flexibility are expected to result in significant advantages. In this paper, we provide Flex-SADMM, a novel stochastic optimization method based on distributed ADMM. To address the subproblems of ADMM, we specifically integrate variance-reduced first-order information and approximate second-order information in order to achieve steady convergence and improve the search direction's precision. Moreover, in contrast to other ADMM-based approaches that require updates from every computational node in each iteration, our framework only requires nodes to update once, at a predetermined iteration interval, greatly increasing flexibility. Experiments validate the effectiveness and improved flexibility of our suggested approach.

INDEX TERMS distributed optimization, ADMM, variance reduction, Hessian approximation, flexibility

I. INTRODUCTION

Numerous fields, including wireless communication [1][2][3][4][5], and machine learning [6][7][8][49][50][51], have made extensive use of mathematical optimization. The theories of deterministic optimization serve as the foundation for stochastic optimization. Its primary component, known as the stochastic gradient descent (SGD), can be traced back to the seminal work [9]. It approximates the exact gradient for the search direction by using a mini-batch of randomly picked data points from the entire dataset.

Its high variance, however, causes a sluggish convergence [10]. As a result, there are numerous techniques aimed at managing and diminishing variance throughout mini-batch optimization. In particular, the variance reduction approaches [11] have been developed to address the aforementioned problem and significantly speed up SGD's convergence [12]. Stochastic average gradient technique (SAG) [13], its extension, the stochastic variance reduction gradient method (SVRG), are exemplar algorithms. Specifically, SAG and SAGA [14] make use of full gradient, updating with the most recent data at each iteration rather than performing a direct calculation. They share SGD's quick convergence time and little processing overhead. Moreover, second-order information—such as the Hessian matrix—is more precise and can result in a higher rate of convergence. To be exact, numerous recursive update strategies for the estimated Hessian matrix have been presented including symmetric rank-one (SR1) update [15][16][17] and rank-two variations such as the Davidon-Fletcher-Powell (DFP) scheme [18][19] and the BFGS update scheme [20][21].

Specifically, under the stochastic regime, [22] has investigated the online BFGS and online limited memory BFGS (oLBFGS). The primary component involves utilising the stochastic gradient to convert the deterministic BFGS and limited memory BFGS (LBFGS) into a stochastic version. The convergence qualities of oLBFGS have been further explored under the strong convexity assumption in [24]. A new stochastic quasi-Newton technique for support vector machine (SVM) issues has been proposed by [25]. It uses term-by-term equalities to approximate the diagonal elements. Since it only involves scalar multiplications - the approximated diagonal elements of the rescaling matrix, this method is highly computationally efficient. Nevertheless, noise may be introduced by directly using the stochastic gradient for the Hessian approximations, which could compromise the convergence's resilience. Hence, it is a challenge for controlling the quality of the curvature approximations in stochastic optimization.

One computational node is used to implement the aforementioned optimization techniques. However, in the big data era, solving a large-scale optimization problem in machine learning with a single computing node is often prohibitive. This is owing to the reason that the quantities of realistic data for training a machine learning model can be very enormous and often vary from 1TB to 1PB.

2

Corresponding author: Fable Tsang (e-mail: 18601024657@163.com).

Because of this, distributed optimisation has emerged as a key technique, and it is now critical to create algorithms that are scalable and efficient enough to handle large datasets in a parallelized or totally decentralised manner. The alternating direction method of multipliers, or ADMM, is ideally suited to solve large-scale machine learning issues and distributed convex optimisation in general. The ADMM method is easy to use and works by breaking down problems into smaller, more manageable tasks that can be completed in parallel. As a result, it is well-suited for a variety of large-scale applications, such as distributed compressive sensing, massive MIMO wireless communication systems [29][30], and machine learning with large-scale data distributed systems [26][27][28]. It has garnered a great deal of attention in both theoretical and practical domains over decades because it penalizes the constraint equality with a quadratic term in order to solve optimisation problems using the augmented Lagrangian function, which is advantageous in numerical stabability [31][32][33][34][35].

ADMM method is simple and its main ingredient is to decompose the problems into subproblems, which can be individually solved in parallel, thus it turns out to be a natural fit in a wide class of large-scale applications, e.g., machine learning with large-scale data-distributed systems [26], distributed compresive sensing [27][28], and massive MIMO wireless communication systems [29][30]. Moreover, since it solves optimization problems via augmented Lagragian function, which penalizes the contraint equality with a quadratic term, it is advantageous in numerical stabability, and hence it has attracted a considerable amount of attention in both practical and theoretical aspects over decades [31][32][33][34][35].

In particular, [31] has offered a thorough analysis of the ADMM framework. A class of strongly convex problems with a linear equality constraint have had their convergence analysis examined. The convergence rate of the ADMM approach with multiple separable blocks of variables in strongly convex optimisation problems has been further explored in the extensive work [38]. The global R-linear convergence rate has been determined under the assumptions of a given error bound condition and a sufficiently small dual stepsize. A decentralised ADMM (DADMM) technique is put out in [39] in an effort to increase computing efficiency. The key component is to fully account for each agent's network architecture, after which each agent can modify its variable to limit its communication to its neighbors. An further linearized version of DADMM (DLM) has been proposed by [40]. It combines the quick convergence rate of ADMM with the computational economy of the distributed gradient method. Both DADMM and DLM's linear convergence rates are further shown under the same strong convexity assumption in the objective function. A class of nonconvex problems with numerous separable blocks of variables has been studied in [41], where the ADMM approach is applied. It is assumed that the penalty value is large enough to make the subproblem strongly convex and easily solved.

The deterministic regime was used to study the aforementioned ADMM approaches, which were predicated on complete accessibility to the entire distributed dataset. However, in practical applications, a big dispersed dataset may still result in a high processing burden. Stochastic settings are added to ADMM as a more natural approach to solving such a problem, which involves using the sampled mini-batch of the dataset for unbiased gradient estimates rather than the entire distributed dataset. A stochastic ADMM approach has been put forth in [42] to address a category of nonsmooth convex problems. Furthermore, given particular assumptions, the convergence rates for convex and strongly convex functions have been studied. [43] have looked into the popular SVRG technique used in stochastic ADMM. For nonconvex situations, [44] has investigated stochastic ADMM in conjunction with a number of variance reduction techniques in great detail. To be more precise, the ADMM approach has been extended to include SVRG, stochastic average gradient (SAG), and the extension of SAG (SAGA). This has led to the creation of SVRG-ADMM, SAG-ADMM, and SAGA-ADMM, respectively. It has been shown that every resulting ADMM approach converges to an expected ϵ-stationary point. In [43], SVRG-ADMM has been further investigated for solving convex optimization problems with composite objective functions, which has achieved the convergence rates of $O(\log S/S)$ for strongly convex and Lipschitz smooth objective functions, and $O(1/\sqrt{S})$ for convex objective functions without Lipschitz smoothness. In [45], a novel stochastic ADMM has been proposed, its main ingredient is to combine classical stochastic ADMM with gradient free and variance reduction strategy.

All of the aforementioned techniques, however, cannot be used directly in a system with several agents, where some agents are only accessible to update variables during each iteration while the other agents remain mute. Even though [38] has increased the adaptability of ADMM, using the stochastic gradient may cause a huge variance in the subproblem. Consequently, to increase the adaptability of the traditional stochastic ADMM approach, we suggest a novel approach. Specifically, the unique approach just calls for each agent to update its variables on a regular basis—at least once. We apply the variance reduction strategy while maintaining the increased flexibility. The variance reduced stochastic gradient is therefore computed at the first stage and employed in the second stage iterations of our proposed technique, which splits the standard stochastic ADMM method procedure into two phases. Note that our proposed technique becomes SVRG-ADMM when the periodicity is set to one, meaning that every agent updates its corresponding variable at every iteration. As a result, our suggested approach generalises the SVRG-ADMM in use today. In brief, the contributions we have made are as follows:

- A new approach to stochastic ADMM is put forth. Since the convexity of the objective function f has not been assumed, our suggested approach may be used to resolve a particular class of nonconvex problems. Furthermore, we include Sd-REG-LBFGS to guarantee the positive definiteness of the Hessian approximation because the subproblem might not be convex. Moreover, it is not necessary for every agent to update its variable throughout every iteration. Rather, we mandate that every agent modifies its variable at least once on a regular basis.
- We use the SVRG approach in our suggested stochastic ADMM technique. As a result, the SVRG approach is likewise divided into two steps in our suggested ADMM method. Although SVRG has been frequently utilised, its application to the stochastic ADMM framework above is not straightforward since the direct combination may not ensure convergence. We suggest that the periodicity be the number of iterations in the second stage of the SVRG technique, given that every agent modifies its variable at least once every period.

II. THE PROPOSED ALGORITHM

In this section, we start with the review of basic theory in ADMM. This part is based on [31]. Then, we will introduce the application of SRL in the subproblem of stochastic ADMM, which is quite straightforward. Specifically, suppose that there are K agents working in parallel (Here, each agent corresponds to a computation node [47]), and the large dataset $\mathcal{D}$ is divided into K subsets $\mathcal{D}_k$. Moreover, $\mathcal{D}_k$ is allocated to the kth agent. Consider the following consensus optimization problem with equality constraint:

$$\begin{array}{ll} \min\limits_{x_k, k=0,\dots,K;} & \sum_{k=1}^{K} f_k(x_k;\mathcal{D}_k) + g(x_0), \\ \text{s.t.} & x_k - x_0 = 0, k = 1,\dots,K \end{array} \tag{1}$$

where $x_k \in \mathbb{R}^d$, $f_k: \mathbb{R}^d \to \mathbb{R}$ is possibly a nonconvex function with respect to the kth agent, and $g: \mathbb{R}^d \to \mathbb{R}$ is assumed to be a convex function with respect to x_0 . Moreover, the single function with respect to the data point $d_i \in \mathcal{D}_k$ is assumed to be $f_{k,i}(x_k; d_i)$. With these settings, we have $f_k(x_k) = \sum_{d_i\in\mathcal{D}_k} f_{k,i}(x_k; d_i)$. Here, we use the notation $f_k(x_k)$ for $f_k(x_k;\mathcal{D}_k)$ for simplicity, i.e., $f_k(x_k) := f_k(x_k;\mathcal{D}_k)$. The problem (1) can be solved through augmented Larangian function as follows:

$$L(\{x_k\},\{\lambda_k\},x_0) = \sum_{k=1}^{K} f_k(x_k;\mathcal{D}_k) + g(x_0) + \sum_{k=1}^{K} \langle \lambda_k, x_k - x_0\rangle + \sum_{k=1}^{K} \frac{\rho_k}{2}\|x_k - x_0\|^2. \tag{2}$$

In this section, we consider the large dataset cases and the subset $\mathcal{D}_k$ is also large. Hence, the traditional stochastic ADMM method exists the following three problems:

(1). In linearized ADMM, the subproblem $x_k^{t+1} = \text{argmin}_{x_k} f_k(x_k^t) + \langle \nabla f_k(x_k^t), x_k - x_k^t\rangle + \frac{\beta}{2}\|x_k - x_k^t\|^2 + \langle\lambda_k, x_k - x_0\rangle + \frac{\rho_k}{2}\|x_k - x_0\|^2$ may be solved more accurately by incorporating Hessian approximation B_k. However, if $f_k(\cdot)$ is nonconvex, the subproblem can be difficult to solve since the reduction in the objective function of the subproblem may not be ensured at each iteration.

(2). Moreover, directly sampling a small subset of $\mathcal{D}_k$ to form the stochastic gradient for optimization may lead to large variance, this may further slow down the convergence speed.

(3). All the K agents are available to implement the update of each $x_k, k = 1,\dots,K$ in parallel, traditional stochastic ADMM method can be employed. However, a more general case is when there are less than K agents available for the implementation of the update at each iteration and thus traditional stochastic ADMM cannot be directly applied.

A. SVRG STRATEGY

The reduction of the variance of stochastic gradient can be realized by two stage optimization. In the first stage, the full gradient is evaluated at the newly updated variable. Then it is stored and used for the calculation of stochastic gradient in next stage. Specifically, denote the iteration number as $(s+1, t+1)$, where $s+1$ is iteration number of the first stage, and $t+1$ is the second stage iteration number, then the stochastic gradient proposed with respect to the kth variable x_k in [12] is given as follows:

$$v_{k,t}^{s+1} = \frac{1}{b_k^t}\sum_{d_i\in\widetilde{\mathcal{D}}_k^t} [\nabla f_{k,i}(x_{k,t}^s; d_i) - \nabla f_{k,i}(\tilde{x}_k^s; d_i)] + \nabla f_k(\tilde{x}_k^s). \tag{3}$$

where a subset of data points $\widetilde{\mathcal{D}}_k^t$ are randomly sampled with the batch size $b_k^t \ll N_k$. It can be easily verified that the stochastic gradient is unbiased, i.e., $\mathbb{E}[v_{k,t}^{s+1}|x_{k,t}^s] = \nabla f_k(x_{k,t}^{s+1})$. As the full gradient $\nabla f_k(\tilde{x}_k^s)$ is only calculated at the first stage and maintained for the second stage, SVRG strategy is computationally efficient. Moreover, it is verified in Lemma 1 that as the algorithm converges, the variance of $v_{k,t}^{s+1}$ will be progressively reduced to null. SVRG is first proposed in [12], and soon it has been widely applied [44][43][45] and shown effectiveness and fast convergence. Considering the above benefits, we incorporate SVRG method in developing our proposed stochastic ADMM method.

As not all agents update their corresponding variables at each iteration, developing an ADMM based algorithm that guarantees convergence can be difficult. However, in the convergence analysis, by requiring that each agent updates its variable at least once every specific period T , the stochastic algorithm is expected to convergence. Specifically, at each iteration $t+1$, define an index set $\mathcal{J}^{t+1} \subseteq \{0,1,\dots,K\}$, if an agent index $k \in \mathcal{J}^{t+1}$, then the agent is required to update the variable x_k, otherwise, the agent keeps the variable x_k using the previous iterate. For the requirement of convergence that each variable must be updated at least once for every T iterations, we have,

$$\bigcup_{t=1}^{T} \mathcal{J}_t^s = \{0,1,\ldots,K\} \quad (4).$$

B. Hessian Approximation Scheme

For dealing the problem 1, let us consider the quadratic approximation of the objective function in ADMM subproblem, specifically, we have:

$$\begin{aligned} x_k^{t+1} = \mathrm{argmin}_{x_k} & f_k(x_k^t) + \langle \nabla f_k(x_k^t), x_k - x_k^t \rangle \\ & + \frac{1}{2}\langle B_k^t(x_k - x_k^t), x_k - x_k^t \rangle \quad (5) \\ & + \langle \lambda_k, x_k - x_0 \rangle + \frac{\rho_k}{2}\|x_k - x_0\|^2 . \end{aligned}$$

Popular recursive update schemes for the approximated Hessian matrix may be used, and these include symmetric rank-one (SR1) update [15][16][17] and rank-two variants such as the Davidon-Fletcher-Powell (DFP) scheme [18][19] and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update scheme [20][21]. In this subsection, we mainly consider BFGS method as it is one of the most popular quasi-Newton algorithms:

$$B_{k+1} = B_k + \frac{y_k y_k^T}{s_k^T y_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}, \quad (6)$$

where the correction pairs are $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ respectively. However, there exists another problem of numerical stability. Especially for nonconvex objective functions, the positive definiteness of the Hessian update is difficult to maintain. Moreover, for small dataset, it may result in the ill-conditioning problem, which harms the convergence. To address these concerns, we adopt Sd-REG-LBFGS update scheme. The main ingredient is to incorporate both regularization scheme and damped parameter, where the former can ensure numerical stability and the latter can keep the positive definiteness of the Hessian approximation matrix. Specifically, Sd-REG-LBFGS updates the Hessian approximation matrix via:

$$B_{k+1} = B_k + \frac{\tilde{\hat{y}}_k \tilde{\hat{y}}_k^T}{s_k^T \tilde{\hat{y}}_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \gamma \quad (7)$$

with the modified gradient difference:

$$\hat{y}_k = \bar{\theta}_k y_k + (1 - \bar{\theta}_k)(B_k + \delta I)s_k, \quad (8)$$

where δ is a given positive constant that satisfies specific condition (See Lemma 1), and the damped parameter is given as:

$$\bar{\theta}_k = \begin{cases} \frac{0.8 s_k^T(B_k+\delta I)s_k - \gamma s_k^T s_k}{s_k^T(B_k+\delta I)s_k - s_k^T y_k}, & \text{if} \quad s_k^T y_k \le 0.2 s_k^T(B_k + \delta I)s_k + \gamma s_k^T s_k, \\ 1, & \text{otherwise}. \end{cases}$$

(9)

Moreover, the extensions to stochastic regime and limited memory version are straightforward.

Algorithm 1 Flex-SADMM

```
Input: initialize $x_k$ for $k = 0, \ldots, K$ and memory size $M$. Choose
  the constant $\delta$ and $\gamma$ satisfying $0.8\delta > \gamma$.
1: for $s = 0,1,\ldots$ do
2:     Set $\tilde{x}_k \leftarrow x_k$, and calculate $\nabla f_k(x_k)$, for $k = 1, \ldots, K$.
3:     for $t = 0,1,\ldots,T-1$ do
4:         if $0 \in \mathcal{J}$ then
5:             $x_0 \leftarrow \mathrm{argmin}_{x_0} L(\{x_k\},\{\lambda_k\},x_0)$
6:         else $x_0 \leftarrow x_0$
8:         end if
9:         if $k \in \mathcal{J}$ then
10:            Update $x_k$ via (5) and update $\lambda_k$ via
               $\lambda_k \leftarrow \lambda_k + \rho_k(x_k - x_0)$
11:        else
12:            $x_k \leftarrow x_k$, $\lambda_k \leftarrow \lambda_k$
13:        end if
14:    end for
15: end for
```

C. FLEX-SADMM

In light of the aforementioned considerations, we put forth a brand-new stochastic ADMM technique that only necessitates that each agent update its matching variable once every T iterations. In order to incorporate SVRG, we split the ADMM technique into two parts. In the first stage, the whole gradient is calculated, and in the second stage, the number of iterations needed is precisely set to T. Each agent can change its variable in this manner at least once in the second stage. We summarise our suggested stochastic ADMM as in Algorithm 1 based on the considerations above.

In Algorithm 1, since all agents shall update their corresponding variables at least once within the T iterations at the second stage, the full gradient $\nabla f_k(\tilde{x}_k^s)$ is calculated in step 2 for all agents $k = 1, \ldots, K$. Then $v_{k,t}^{s+1}$ is further computed for $k \in \mathcal{J}$, which is used for updating the corresponding variable in step 10. It should be noted that the steps 2-14 can be carried out in parallel. Specifically, the chosen agents in the index set $\mathcal{J}$ can update the variables x_k and λ_k in a distributed manner. In step 10, the Hessian approximation by Sd-REG-LBFGS is maintained positive definite. As mentioned above, the strategy can avoid the ill-conditioning problem and thus our proposed algorithm can perform robustly under small samples. Moreover, the subproblem in step 10 is a quadratic convex problem and can be solved by directly nulling the first-order derivative

Remark: We suggest using our proposed stochastic ADMM approach in a parameter server. Moreover, asychronous and synchronous algorithms have been thoroughly studied for a popular distributed gradient descent technique. Asynchronous techniques gather worker data partially, whereas synchronous algorithms need the server to wait for updates from every worker. While the ADMM approach can boost fault tolerance, distributed gradient descent (DGD) has a comparatively lower fault tolerance. As a result, the ADMM approach can be modified for an asynchronous algorithm. Our suggested approach provides a framework of adaptation to asynchronous algorithms, since each worker only needs to update its variables once every T iterations.

Convergence Analysis: For notational simplicity, let us denote $x_{k,t}^{s+1}$ as x_k^t, the same strategy is used for other variables. Subsequently, the following lemma shows the variance of the stochastic gradient is progressively reduced to zero when the algorithm converges.

Theorem 1. For each $k \in \{1, ..., K\}$ the sequence $\{x_{k,t}^s, \lambda_{k,t}^s, x_{0,t}^s\}$ generated by the proposed Algorithm 1 is expected to converge to a limit point $\{x_k^*, \lambda_k^*, x_0^*\}$ which satisfies:

$$\mathbb{E}\left\|\nabla f_k(x_k^*) + \lambda_k^*\right\| = 0;$$
$$\mathbb{E}\|x_k^* - x_0^*\| = 0;$$
$$\mathbb{E}[d(\textstyle\sum_{k=1}^K \lambda_k^*, \partial g(x_0^*))] = 0,$$

where the metric $d(x, \mathcal{Y})$ is the minimal distance between the vector x and the set $\mathcal{Y}$, i.e.,

$$d(x, \mathcal{Y}) = \min_{y \in \mathcal{Y}} \|x - y\|. \tag{41}$$

Moreover, given any sufficiently small positive value $\varepsilon > 0$, the iteration number S needed to achieve the following ε-stationary point

$$\mathbb{E}\left\|\nabla f_k(x_{k,t}^s) + \lambda_{k,t}^s\right\| \le \varepsilon;$$
$$\mathbb{E}\left\|x_{k,t}^s - x_{0,t}^s\right\| \le \varepsilon;$$
$$\mathbb{E}\left[d\left(\textstyle\sum_{k=1}^K \lambda_{k,t}^s, \partial g\left(x_{0,t}^s\right)\right)\right] \le \varepsilon.$$

satisfies $S \ge \frac{\eta(\zeta_0^2 - \zeta^*)}{\tau\varepsilon}$ for some positive constant $\eta > 0$ and $\tau > 0$.

IV. NUMERICAL EXPERIMENTS

We have studied the theoretical convergence properties of our proposed Flex-SADMM. For the convergence speed in the *Theorem 1*, we have shown that it is closely related to the first stage iteration number S. However, it is also related to the second stage iteration number T. Let us illustrate this briefly. For the convergence speed $O(1/S)$ in *Theorem 1*, it has been derived under the assumption that each agent k has updated its variables at least one time within the second stage. In fact, if we assume that each agent k is required to update its variables r times within the second stage, it can be straightforwardly derived that the convergence speed will be $O(\frac{1}{r \cdot S})$. Hence, in this section, we will study the effect of the update times r for each agent at the second stage. Here, five scenarios based on ADMM are applied for studying the effectiveness of using our proposed Hessian approximation scheme:

1). Flex-SADMM: it will be implemented according to Algorithm 1;

2). Flex-SADMM without BFGS: the only difference to scenario 1) is that identity matrix will be utilized instead of updating the Hessian approximation matrix;

3). Flex-SADMM without SVRG: to reflect the tremendous performance improvement brought by SVRG, we also implemented Flex-SADMM with no SVRG to show its importance and efficiency;

4). Flex-SADMM without SVRG or BFGS: it will be seen that SVRG has brought large performance improvement, which even overwhelms the BFGS. Hence, Flex-SADMM with no SVRG or BFGS is implemented.

5). Fast ADMM: the momentum based algorithm [48];

For the dataset, we choose *scene* dataset [46] (available at http://mulan.sourceforge.net /datasets-mlc.html), since it is simple to study the effect of the parameters. Furthermore, we consider the above 5 scenarios applying to solve logistic regression for binary classification problem due to the simplicity of logistic regression. For the performance evaluation, we choose the norm of gradient (NOG) such that it can reflect how close the iteration point is to the optimal point, i.e.,

$$\text{NOG} = \left\|\frac{1}{N}\sum_{n=1}^N [z_n - \sigma(\theta^T x_n)]x_n\right\| \tag{57}$$

where $\boldsymbol{\theta}$ is the optimization variable, $\boldsymbol{z_n}$ is the class label and $\boldsymbol{x_n}$ is the feature vector.

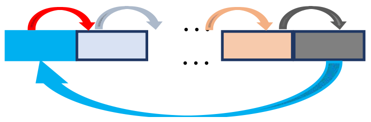


Figure 1 The arrangement of the available agents at each iteration in the second stage.

B. THE IMPACT OF FIRST STAGE

Recall that with the assumption that each agent should update its variables at least one time with the second stage, the convergence speed will become $O(1/S)$. Moreover, the number of the iteration in the first stage determines the times of calculating the full gradient. Therefore, with the above discussions, we continue to study the effect of the iteration number in first stage, namely, S. Here, the total iteration number is set to 1, 000 to ensure fair comparison. Hence, the iteration number in second stage is $1000/S$. We choose $S = [10, 20, 50, 100]$. Moreover, for simplicity, we arrange the available agents as one block, and each block has no common agent. According to this, we set the available agent number to be $AG = 5$. Moreover, the dataset is divided into $K = 10$. Subsequently, we have each agent will update its variables $r = [50,\ 25,\ 10,\ 5]$ times respectively.

Figure 3 shows the results of different iteration number settings in the first stage. It can be seen that convergence speed is nearly the same. This is because for the four scenarios of $r = [50, 25, 10, 5]$, since the convergence speed is $O(\frac{1}{r \cdot S})$ as aforementioned discussions, the four scenarios share nearly the same convergence speed.

This matches theory well. In particular, the zigzags along the line shows the regularity, which is an interesting phenomenon. To be specific, taking Figure 3 (a) as the example, there are in general 10 zigzags and each is with smaller zigzags oscillating, we call the larger zigzag as global zigzag and the smaller as local zigzag. For the Figure 3 (a), the number of global zigzags equals to the first stage iteration number $S = 10$. This is due to the reason that the full gradient is calculated at each start of the first stage, and then it has been passed to the second stage. Hence, with the newly calculated full gradient, it will drop more obviously at the start of the second stage. For the local zigzags, it is obvious that it lies within second stage. It oscillates because only one block of agents updates its variables at each iteration while the others remain silent, and then next block is chosen for the updating. Hence, larger S will lead to dense zigzags and this matches well as the Figure 3 shows.

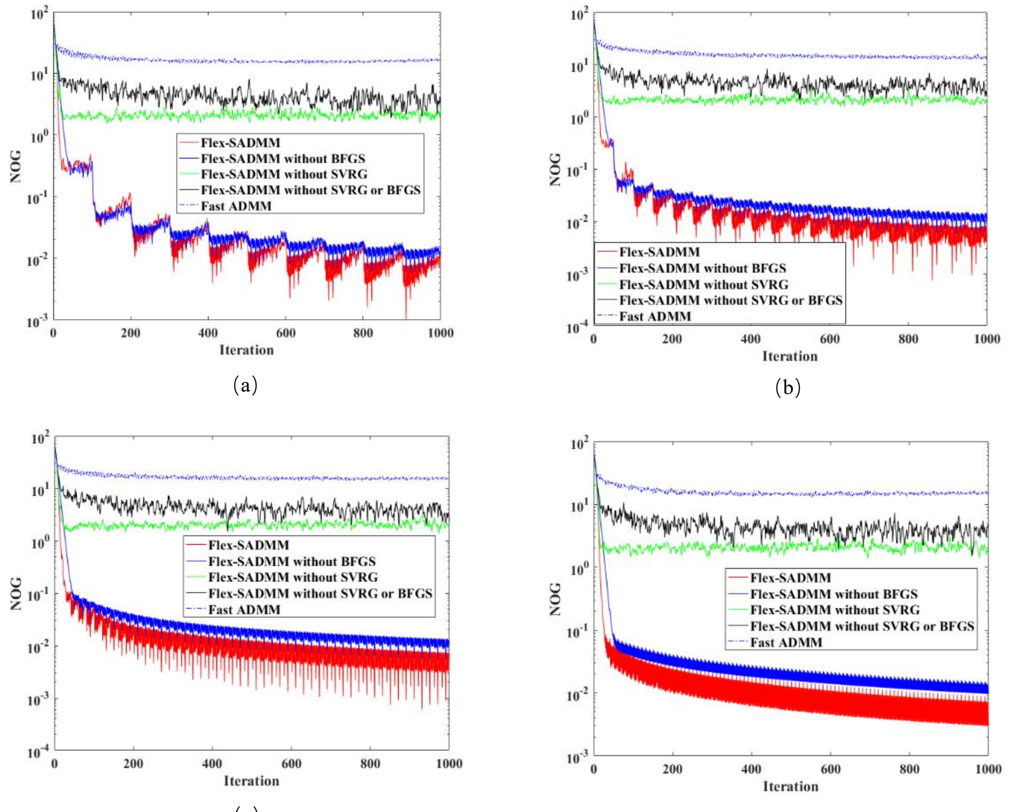


Figure 3 The effect of the iterations number in the first stage, to be specific, (a). $S = 10$; (b). $S = 20$; (c). $S = 50$; (d). $S = 100$.

C. THE IMPACT OF AVAILABLE AGENTS

According to the above experimental results, we notice that the updating times of each agent within the second stage have a significant impact on the performance of our proposed method. Hence, we continue to study the impact of different number of the available agents at each iteration, which can determine the update times of each agent. Specifically, the total iteration number is set to 1, 000, and the first stage iteration number is set to $S = 50$, such that the update times of each agent can be varied more flexibly. For the data division, it will be set to $K = 20$. Furthermore, we set $AG = [1, 3, 5, 7, 9, 20]$. Subsequently, the corresponding update times for each agent are $[1, 3, 5, 7, 9, 20]$ respectively. It should be noted that when $AG = 20$, it means all agents update their variables at each iteration.

As Figure 4 illustrates, it can be seen that in general larger update times for each agent lead to a better performance of NOG and faster convergent speed. This matches well with the convergent theory $O(\frac{1}{r \cdot S})$. Note that the curves in Figure 4 (a) and (f) are much smoother. For Figure 4 (f), the reason is that all the agents update their variables and thus there will be no zigzags. In Figure 4 (a), it can be seen that at each start of the second stage, the curve drops more sharply than the cases in the iterations of the second stage. However, compare to other cases that multiple agents update their variables at each iteration, this decreasing

is much slight. This is because only one agent performs to decrease NOG. For other cases that have multiple agents available at each iteration, it can be seen that the performance improvement becomes smaller as the number of available agents increases. Hence, our proposed method has strong ability in fault tolerance, which also means that one can choose fewer available agents at each iteration for the update, because the very slight performance sacrificing is worth trading for saving hardware resources.

D. THE STUDY OF THE CONVERGENCE

Next, we continue to study when the condition that each agent updates its variables at least once is not satisfied. To achieve this scenario, the dataset is divided into $K = 60$ parts. The total iteration number is set to 1, 000 with $S = 50$. Hence, under the condition that adopts the available agent scenario according to Figure 1, there should be at least a block of 3 agents updating the variables at each iteration. To see if it is not satisfied, we choose to set to be $AG = [1, 2, 3, 4]$. Obviously, we have set three groups for studying, where the first group does not satisfy the condition, the second group just meets the condition and the third group just surpasses the condition. Specifically, $AG = [1, 2]$ does not satisfy the convergence condition. $AG = 3$ just meets the condition while $AG = 4$ just exceeds the condition.

It can be seen from Figure 5 that when the condition that each agent should update its variables at least one time (here, $AG \geq 3$) is not satisfied, all algorithms perform poorly. $AG = 3$ seems to be a water-shed, which means beyond $AG = 3$ it can reach a better and satisfactory performance, while below $AG = 3$, the performance is poor. Note that when $AG = 4$, the performances of all algorithms are substantially improved compared to $AG \leq 3$. Therefore, it matches *Theorem 1* well.

V. CONCLUSION

As there exists different issues for classical stochastic ADMM methods, we have proposed a novel stochastic ADMM method to address these concerns. Specifically, we first incorporate SVRG strategy and divide the ADMM procedure into two stages. At the second stage, the agents work for updating their corresponding variables in parallel. However, we only require each of them updates its variable at least once at the second stage. In the comparisons with

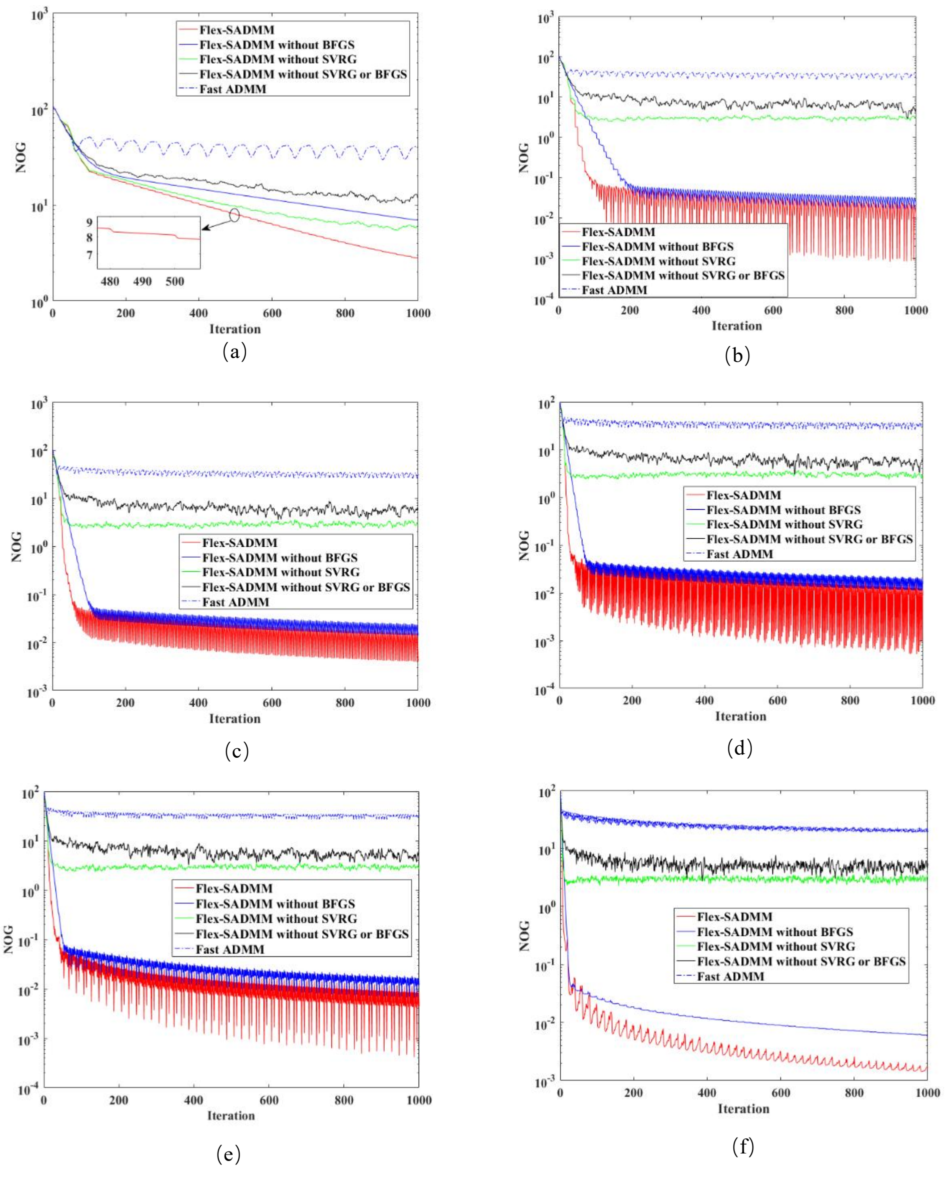
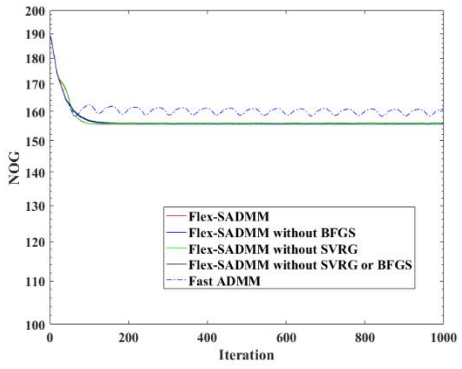
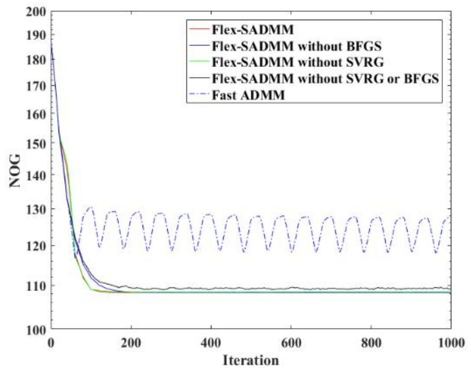


Figure 4 The effect of the number of available agents at each iteration, to be specific, (a). $AG = 1$; (b). $AG = 3$; (c). $AG = 5$; (d). $AG = 7$; (e). $AG = 9$; (f). $AG = 20$.

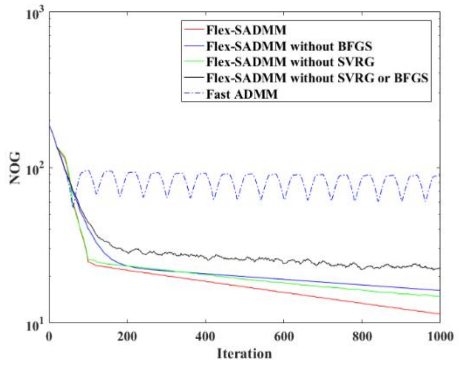
other methods, our proposed method has shown to be advantageous especially in flexibility.



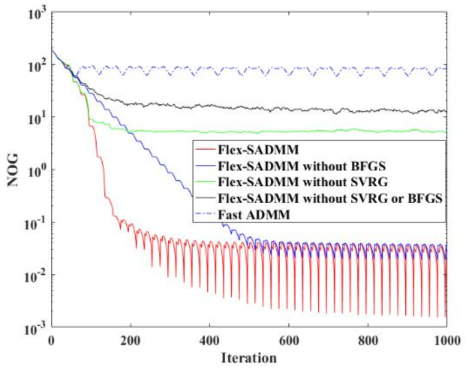
(a)



(b)



(c)



(d)

Figure 5 The study of the case when the condition that each agent should update its variables at least one time is not satisfied, to be specific, (a). $AG = 1$; (b). $AG = 2$; (c). $AG = 3$; (d). $AG = 4$

REFERENCES

[1] Huiming Chen* and W. H. Lam, "Training Based Two-Step Channel Estimation in Two-Way MIMO Relay Systems," IEEE Transactions on Vehicular Technology, vol. 67, no. 3, pp. 2193-2205, March 2018.

[2] Huiming Chen*, W. H. Lam, “Training Design and Two Stage Channel Estimation for Correlated Two-way MIMO Relay Systems Under Colored Disturbance”, Proceedings of the IEEE International Conference on High Performance Computing and Communications, Zhangjiajie, China, Aug., 2019.

[3] Huiming Chen*, W. H. Lam, “Joint LMMSE and MAP Channel Estimation using BCRLB training signals for Correlated Two-way MIMO Relay Systems”, Proceedings of the IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), Hong Kong SAR, China Aug. 2016.

[4] Huiming Chen*, W. H. Lam, “Training Design and Two Stage Channel Estimation for Correlated Two-way MIMO Relay Systems”, *Proceedings of the IEEE International Conference on Ubiquitous and Future Networks (ICUFN)*, Vienna, Austria, Jul. 2016.

[5] P Hansen, B Jaumard, and SH Lu, “Global Optimization of Univariate Lipschitz Functions, Part 1: Survey and Properties”, *Mathematical Programming*, Vol. 55, pp. 251–272, 1992.

[6] Huiming Chen, H. Wang, Y. Li*, Q. Yao, D. Jin and Q. Yang, “LoSAC: An Efficient Stochastic Average Control Method for Federated Optimization”, ACM Transactions on Knowledge Discovery from Data, vol. 17, no. 4, Apr., 2023.

[7] Huiming Chen, H. Wang, Y. Li* and D. Jin,“Advancements in Federated Learning: Models, Methods, and Privacy”, ACM Computing Survey, Dec., 2023.

[8] D. M. Blei, A. Kucukelbir and J. D. McAuliffe, “Variational Inference: A Review for Statisticians,”*J. Am. Statist. Assoc.*, vol. 112, no. 518, pp. 859-877, 2017.

[9] H. Robbins and S. Monro, “A Stochastic Approximation Method,”*Ann. Math. Statist.*, vol. 22, no. 3, pp. 400 - 407, 1951.

[10] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT’2010*. Springer, 177–186.

[11] R. Gower, M. Schmidt, F. Bach, and P. Richtarik. 2020. “Variance-reduced methods for machine learning”, arXiv preprint:arXiv:2010.00892, Oct. 2020.

[12] R. Johnson and T. Zhang. “Accelerating Stochastic Gradient Descent using Predictive Variance Reduction”. *NIPS*, pp. 315–323, 2013.

[13] M. Schmidt, N. Roux, J. Liu, and F. Bach. “Minimizing finite sums with the stochastic average gradient”. *Mathematical programming*, 162, pp. 83–112, 2016.

[14] A. Defazio, F. Bach, and S. Julien, “SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives”. Advances *in Neural Information Processing Systems*, pp. 1646–1654, 2014.

[15] H. Benson, Y. Shanno, “Cubic regularization in symmetric rank-1 quasi-Newton methods. Mathematical Programming Computation,”*Mathematics of Computation*, no. 10, vol.4, pp. 457 - 486, 2018.

[16] H. Khalfan; R. Byrd; R. Schnabel, “A Theoretical and Experimental Study of the Symmetric Rank-One Update,”*SIAM J. Optim.*, vol. 3, no. 1, pp. 1 - 24, Feb. 1993

[17] A. Conn, N. Gould, P. Toint, “Convergence of quasi-Newton matrices generated by the symmetric rank one update,”*Mathematical Programming*, vol. 50, no. 1-3, pp. 177 - 195, Mar. 1991

[18] J. E. Dennis, J. Moré, “A Characterization of Superlinear Convergence and Its Application to Quasi-Newton Methods,”*Mathematics of Computation*, Vol. 28, No. 126, pp. 549 - 560, Apr., 1974

[19] W. C. Davidon, “Variable metric method for minimization,”*SIAM J. Optim.*, Vol. 1, No. 1, pp. 1 - 17, 1991

[20] C. G. Broyden, J. E. Dennis, J. Moré, “On the Local and Superlinear Convergence of Quasi-Newton Methods ,”*IMA J. Appl. Math.*, vol. 13, no. 3, pp. 223 - 245, Dec. 1973

[21] C. G. Broyden, “The convergence of a class of double-rank minimization algorithms 2. the new algorithm ,”*IMA J. Appl. Math.*, vol. 6, no. 1, pp. 222 - 231, 1970.

[22] N. Schraudolph, J. Yu, and S. Gunter, “A stochastic quasi-Newton

[23] method for online convex optimization,”in *Proc. 11th Int. Conf. Artif. Intell. Statist.*, pp. 433 – 440, 2007.

[24] A. Mokhtari, and A. Ribeiro, “Global Convergence of Online Limited Memory BFGS,”*J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 3151 - 3181,
Jan., 2015.

[25] A. Bordes, L. Bottou, and P. Gallinari, “SGD-QN: Careful Quasi-Newton Stochastic Gradient Descent,”*J. Mach. Learn. Res.*, 10, pp. 1737 – 1754, Jul. 2009.

[26] Y. Yang, J. Sun, H. Li and Z. Xu, “ADMM-CSNet: A Deep Learning Approach for Image Compressive Sensing”, to appear in *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[27] H. Liu, B. Song, H. Qin and Z. Qiu, “An Adaptive-ADMM Algorithm With Support and Signal Value Detection for Compressed Sensing”, *IEEE Signal Processing Letters*, vol. 20, no. 4, pp. 315-318, April 2013.

[28] Z. Du, X. Chen, H. Zhang and B. Yang, “Compressed-Sensing-Based Periodic Impulsive Feature Detection for Wind Turbine Systems”, *IEEE Transactions on Industrial Informatics*, vol.13, no.6, pp. 2933 – 2945, Dec. 2017.

[29] E. Vlachos, G. Alexandropoulos and J. Thompson, “Massive MIMO Channel Estimation for Millimeter Wave Systems via Matrix Completion”, *IEEE Signal Processing Letters*, vol. 5, no. 11, pp. 1675 – 1679, Sept. 2018.

[30] Y. Tsai, L. Zheng and X. Wang, “Millimeter-Wave Beamformed Full-Dimensional MIMO Channel Estimation Based on Atomic Norm Minimization”, *IEEE Transactions on Communications*, vol. 66, no. 12, Dec. 2018.

[31] S. Boyd, N. Parikh, E. Chu, B. Peleato and J. Eckstein, “Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers”, *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1-122, Jan. 2011.

[32] 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, April, 2014.

[33] M. Hong, Z. Luo and M. Razaviyayn. “Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems”. *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 337–364, 2016.

[34] 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, Aug. 2015

[35] Mingyi Hong and Zhi-Quan Luo, “On the linear convergence of the alternating direction method of multipliers”, *Mathematical Programming*, vol. 162, no. 1-2, pp. 165-199, Mar. 2017.

[36] Y. Tsai, L. Zheng and X. Wang, “Millimeter-Wave Beamformed Full-Dimensional MIMO Channel Estimation Based on Atomic Norm Minimization”, *IEEE Transactions on Communications*, vol. 66, no. 12, Dec. 2018.

[37] H. Liu, B. Song, H. Qin and Z. Qiu, “An Adaptive-ADMM Algorithm With Support and Signal Value Detection for Compressed Sensing”, *IEEE Signal Processing Letters*, vol. 20, no. 4, pp. 315-318, April 2013.

[38] M. Hong and Z. Luo, “On the linear convergence of the alternating direction method of multipliers”, *Mathematical Programming*, vol. 162, no. 1-2, pp. 165-199, Mar. 2017.

[39] 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, April, 2014.

[40] 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, Aug. 2015.

[41] M. Hong, Z. Luo and M. Razaviyayn. “Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems”. *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 337–364, 2016.

[42] H. Ouyang, N. He, L. Tran and A. Gray, “Stochastic Alternating Direction Method of Multipliers”. *In Proceedings of the 30st International Conference on Machine Learning*, pp. 80–88, 2013.

[43] Y. Yu and L. Huang, “Fast Stochastic Variance Reduced ADMM for Stochastic Composition Optimization”, *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 3364-3370, Aug. 19-25, 2017.

[44] F. Huang, S. Chen and Z. Lu. “Stochastic alternating direction method of multipliers with variance reduction for nonconvex optimization”. *arXiv:1610.02758*, 2016.

[45] F. Huang, S. Gao, S. Chen and H. Huang, “Zeroth-Order Stochastic Alternating Direction Method of Multipliers for Nonconvex Nonsmooth Optimization”, *arXiv:1905.12729*, 2016.

[46] M.R. Boutell, J. Luo, X. Shen, and C.M. Brown, “Learning multi-label scene classiffication,”*Pattern Recognition*, vol. 37, no. 9, pp. 1757-1771, 2004.

[47] T. Chang, M. Hong, and X. Wang, “Multi-Agent Distributed Optimization via Inexact Consensus ADMM”, *IEEE Transactions on Signal Processing*, vol. 63, no. 2, Jan. 15, 2015.

[48] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, “Fast Alternating Direction Optimization Methods”, *SIAM J. Imaging Sci.*, no.7, vol. 3, pp. 1588–1623, 2014.

[49] Liu H, Zhao H, Wang J, et al. LSTM-GAN-AE: A promising approach for fault diagnosis in machine health monitoring[J]. IEEE Transactions on Instrumentation and Measurement, 2021, 71: 1-13.

[50] Liu H, Zong Z, Li Y, et al. NeuroCrossover: An intelligent genetic locus selection scheme for genetic algorithm using reinforcement learning[J]. Applied Soft Computing, 2023, 146: 110680.

[51] Liu H, Zhao H, Geng L, et al. A policy gradient based offloading scheme with dependency guarantees for vehicular networks[C]//2020 IEEE Globecom Workshops (GC Wkshps. IEEE, 2020: 1-6.