Pure and New Mathematics in AI 2024, 1(1), 8499.
https://doi.org/10.24294/pnmai8499

Article

Asynchronous recurrent neural networks with block splitting for distributed partitioned optimization

Jingxin Liu[1,2,*], Jun Peng[1,*], Amin Mansoori[3], Chaoran Zhan[4], Ye Huang[5], Huanbin Wang[6]

[1] Chongqing Research Institute of Intelligent Mathematics and Autonomous AI (RI-IM·AI*), School of Mathematics and Physics Sciences, Chongqing University of Science and Technology, Chongqing 401331, China
[2] Center for Applied Mathematics of Guangxi, Guangxi Minzu University, Guangxi 530006, China
[3] Department of Applied Mathematics, Ferdowsi University of Mashhad, Mashhad 9177948974, Iran
[4] School of Electrical and Electronic Engineering, University of Manchester, Manchester M13 9PL, U.K.
[5] School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore
[6] College of Computing and Data Science, Nanyang Technological University, Singapore 639798, Singapore
*** Corresponding authors:** Jingxin Liu, jingxinliu9@126.com; Jun Peng, jpeng@cqust.edu.cn

CITATION

Liu J, Peng J, Mansoori A, et al. Asynchronous recurrent neural networks with block splitting for distributed partitioned optimization. Pure and New Mathematics in AI. 2024; 1(1): 8499. https://doi.org/10.24294/pnmai8499

ARTICLE INFO

Received: 9 August 2024
Accepted: 20 September 2024
Available online: 8 October 2024

Abstract: This paper presents a class of novel recurrent neural network approaches for a distributed partitioned optimization scenario, where the objective function is separable, strongly convex, and possibly nonsmooth, with the computation of a part of the solution being distributed to a vertex for execution. In our proposed algorithmic framework, the block splitting method allows the solution to be partitioned among vertices according to the divisible structure of the problem, so that each neuron only holds a local memory of the decision variable rather than the memory of the entire decision variable. A local timer is installed for each neuron. If a neuron is triggered by its own timer and a neighbor timer, it will reach an activated state and then update and transmit its own variable information. This asynchronous evolution strategy with time helps to save computational resources. The proposed algorithm is distributed and scalable, with the computation of a single neuron not depending on the size of the vertex network, and the convergence of the algorithm can be guaranteed.

Keywords: recurrent neural network; distributed partitioned optimization; block splitting; asynchronous evolution; local memory

1. Introduction

Distributed optimization is a technique that performs optimization tasks on multiple computing vertices and is designed to handle large-scale datasets and computational problems. In the modern computing environment, especially with the dramatic increase in the amount of data, a single computing unit is often unable to effectively deal with large-scale data or complex computing tasks. Distributed optimization technology makes the overall computing process more efficient and scalable by distributing tasks and data among multiple computing vertices. This technology distributes computing tasks to multiple vertices over the network, which makes the computing process more efficient and scalable. This kind of computing framework is widely used in robotics [1], data processing [2], transportation [3], and other scenarios.

For some parts of the distributed optimization problems, where the information over the network may be redundant or unavailable, such as resource allocation [4], network utility maximization [5], and energy management [6,7], an additional property should be introduced to describe the situation, in which each vertex only performs the computation of a part of the solution, and the whole solver can be constructed by aggregating these local computations. This kind of distributed optimization problem is graphically called distributed partitioned optimization problem, and it is commonly found in modern big data computing applications, such as distributed matrix completion [8], power transfer [9], sparse reconstruction [10]. As far as the standard formalization is concerned, the goal is to minimize a sum of N local objective functions $f_i(x)$ for $x \in \mathbb{R}^n$, and the information of each $f_i(x)$ is stored in a vertex of the network. Different from the standard distributed problems, the decision space is split into $m(m \leq n)$ blocks, and the computation of each $f_i(x)$ may depend on only a subset of blocks. This sparse structure causes previous standard methods [11,12] to appear inefficient, as they require each vertex to store and send the entire decision vector instead of storing and exchanging local decision vector that have an impact on its objective $f_i(x)$ with other vertices.

The theory of distributed algorithms for such additional properties does not seem to have received much attention. Necoara and Clipici [13] introduced a stochastic coordinate descent approach with parallelizability for minimizing the sum of a partially separable smooth convex function and a fully separable nonsmooth convex function and also provided instructions on the algorithm implementation and convergence rate guarantee. Notarnicola et al. [14] designed an asynchronous dual decomposition algorithm to solve such problems, providing a proof of its asymptotic convergence. Bastianello et al. [15] presented a distributed algorithm based on the relaxed ADMM for solving a class of optimization problems with separable convex cost functions. These previous approaches require each vertex not only to store the objective information but also to retain the memory of the corresponding variable block, which can lead to redundant computation when computing gradients for all variables.

At present, the main challenges in solving such problems are that the storage capacity and computation power of vertices depend on the scale of the network, and additionally, all vertices update and transmit data simultaneously, which may lead to unnecessary computing consumption. Recurrent neural networks (RNNs) are widely used in the field of optimization by virtue of their parallel computing and distributed storage and memory functions [16–21]. Especially for large-scale optimization problems, Xia et al. [22] developed an RNN system to address the Clifford-valued distributed optimization with linear equality and inequality constraints. Jia et al. [23] proposed a recurrent neural network (RNN) model with a novel auxiliary function for addressing distributed optimization problems over multi-agent networks. Rajesh et al. [24] presented a hybrid approach of dynamic differential annealed optimization and recalling enhanced RNN. Liu et al. [25] proposed an RNN approach for the minimization of the sum of a set of fuzzy convex functions in the differential inclusion framework. Sherstinsky [26] provided a significant explanation and theoretical derivation of the basic principles of RNN and LSTM networks. Lin et al. [27] presented an efficient route programming for multiple automated guided vehicles by using deep reinforcement learning and RNN. Considering the maturity of RNNs in solving optimization problems, we attempt to apply an RNN model to solve distributed partitioned optimization problems in this paper. RNN can not only predict the future state through the storage memory of historical decisions but also process multiple inputs about local information in time to ensure real-time feedback on optimization performance.

In the algorithm design, we also consider the block splitting method [28]. Conor et al. [29] designed two distributed network optimization algorithms based on the function splitting and the quadratically approximated method, which simplify the local subproblems that must be solved in each update iteration for vertices and improve the computational efficiency on distributed processors. Latafat et al. [30] proposed a triangularly preconditioned primal-dual algorithm to solve the minimization problem of the sum of a Lipschitz-differentiable convex function and two possibly nonsmooth convex functions, one of which consists of a linear map. Based on their research, Li et al. [31] considered the case of a linear function as composite and nonsmooth, providing a distributed primal-dual splitting algorithm along with its asynchronous version. The advantages of block splitting in ensuring convergence and reducing computational load are considerable. To routinizing distributed partitioned optimization problems and their dual forms, we propose an RNN with block splitting. Not only that, the RNN is implanted with an asynchronous update mechanism in the state evolution of neurons, which can improve the computational efficiency of the network, especially in the distributed computing environment. Neurons can be updated independently and in parallel to obtain faster training speed [32]. A considerable conclusion is that asynchronously updated RNNs can dynamically adapt to changing input conditions without requiring the entire network to process each input simultaneously, which will lead to better performance in tasks where input patterns change over time [33].

The main contributions of this paper are as follows. Firstly, the proposed RNN algorithm is full of scalability, in the sense that each neuron processes only a part of the vector of decision information, and neither the data information stored nor the computation performed by each neuron depends on the scale of the distributed network. Second, when a neuron is triggered by its local timer or neighbor timer, it reaches an active state, then updates and transmits its decision information to neighbors. This asynchronous evolution mechanism, based on block splitting, helps reduce communication overhead and save computing resources. Finally, the subgradient update for the dual partitioned problem follows a gradient ascent path, and a reasonable time step ensures that the algorithm finds the optimal decision with a high probability. This paper is organized as follows. In Section 2, we give some prerequisites and a formulation description of the partitioned optimization problem. In Section 3, a novel asynchronous RNN algorithm and its theoretical analysis are mentioned. Finally, two numerical examples about resource allocation and square-root loss elastic-net are dispensed in Section 4, and some conclusions are drawn in Section 5.

2. Preliminaries and problem description

2.1. Notation and definitions

Consider a undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = (1, \cdots, N)$ denotes the set of vertices, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges. The i-th and j-th vertices can exchange information if $(i, j) \in \mathcal{E}$. Denote $\mathcal{N}_i = \{j \in \{1, \cdots, N\}: (i, j) \in \mathcal{E}\}$ by the neighbor vertices of the i-th vertex. For a vector $x \in \mathbb{R}^n$, expand it in the form of N blocks as $x = [x_1^T, \cdots, x_N^T]^T$, in which $x_i \in \mathbb{R}^{n_i}$ satisfies $\sum_{i=1}^N n_i = n$. Divide the identity matrix $I \in \mathbb{R}^{N \times N}$ into N blocks in form $I = [I_1, \cdots, I_N]$, in which $I_i \in \mathbb{R}^{n \times n_i}$, one gets $x_i = I_i^T x$ and $x = \sum_{i=1}^N I_i^T x_i$. For a nonsmooth function $f: \mathbb{R}^n \mapsto \mathbb{R}$, its subdifferential at x is denoted by $\partial f(x)$, its partitioned subdifferential with respect to x_i is denoted by $\partial_{x_i} f(x)$ and $\partial_{x_i} f(x) = U_i^T \partial f(x)$. Denote $\mu(x) \in \partial f(x)$ and $\mu_i(x) \in \partial_{x_i} f(x)$ as a subgradient of f at x and the corresponding partitioned subgradient with respect tox_i, respectively. The optimal value of a function f over its domain $dom f$ is denoted by f_{opt}.

Definition 1 [33]. For a function $f: \mathbb{R}^n \mapsto \mathbb{R}$, its conjugate function $f^*: \mathbb{R}^n \mapsto \mathbb{R}$ is defined by $f^*(y) = \sup_{x \in dom f} \left(y^T x - f(x)\right)$.

Definition 2 [33]. Consider an optimization model:

$$\begin{aligned} &\text{minmize} \quad f(x) \\ &\text{subjectto} \quad h_k(x) = 0,\ k \in \{1, \cdots, m\} \\ &\qquad\qquad\quad g_l(x) \leq 0,\ l \in \{1, \cdots, n\}. \end{aligned} \tag{1}$$

The Lagrangian function is constructed as

$$L(x, \lambda, v) = f(x) + \sum_{k=1}^{m} \lambda_k h_k(x) + \sum_{l=1}^{n} v_l g_l(x), v_l \geq 0 \tag{2}$$

where $\lambda = [\lambda_1, \cdots, \lambda_m]^T$ and $v = [v_1, \cdots, v_n]^T$ denote the Lagrangian multiplier vector. The Lagrangian dual function is defined by

$$\psi_d(\lambda, v) = \inf_x L(x, \lambda, v) \tag{3}$$

which is independent *of* x.

2.2. Problem model

We consider the minimization of a class of distributed partitioned optimization problems subject to local constraints, where the structure of the objective function is separable. The specific form is as follows:

$$\begin{aligned} &\text{minmize} \quad \sum_{i=1}^{N} f_i(x) \\ &\text{subjectto} \quad x \in X_i \subset \mathbb{R}^n,\ i \in \{1, \cdots, N\} \end{aligned} \tag{4}$$

where the local objective function $f_i: \mathbb{R}^n \mapsto \mathbb{R}$ is continuous, convex and possibly nonsmooth. The local objective f_i and constraint X_i are accessible only to the i-th vertex. The decision variable $x \in \mathbb{R}^n$ can be partitioned by $x = [x_1^T, \cdots, x_N^T]^T$, where $x_i \in \mathbb{R}^{n_i}$ satisfies $\sum_{i=1}^N n_i = n$. Since the restricted relationship $x \in X_i$ applies to any $i \in \{1, \cdots, N\}$, then $x \in X \triangleq \bigcap_{i=1}^N X_i$. We introduce block set $\mathcal{X}_i \subset \mathbb{R}^{n_i}$ to implement block decomposition on the constraint set X, i.e., $X=\prod_{i=1}^{N}\mathcal{X}_i=\mathcal{X}_1\times\cdots\times\mathcal{X}_N$, where $\mathcal{X}_i$ can be understood as the restriction set of the i -th decision block and $\times$denotes the Cartesian product. Additionally, the local objective functions and constraints are consistent with the sparsity of the communication topology, that is, f_i and $\mathcal{X}_i$ are rely only on the decision block of the i -th vertex and its neighbors. This problem can be represented in the following distributed way:

$$\begin{aligned} &\text{minmize} \quad \sum_{i=1}^{N} g_i\left(x_i,[x_j]_{j\in\mathcal{N}_i}\right) \\ &\text{subjectto} \quad \left(x_i,[x_j]_{j\in\mathcal{N}_i}\right)\in\mathcal{X}_i\times\prod_{j\in\mathcal{N}_i}\mathcal{X}_j\,,\ i\in\{1,\cdots,N\} \end{aligned} \tag{5}$$

where g_i: $\mathbb{R}^{n_i+\sum_{j\in\mathcal{N}_i}n_j}\mapsto\mathbb{R}$ is viewed as the local objective function determined by the variables x_i and $[x_j]_{j\in\mathcal{N}_i}$. It can be seen that the constraint is actually applied to the i -th vertex and its neighbors, not just the i -th vertex. This seemingly trivial feature actually brings more generality to the problem model and adds significant challenges. The following assumptions about objective function and constraints are applicable to the research work in this paper.

Assumption 1. Each local objective functiong_i: $\mathbb{R}^{n_i+\sum_{j\in\mathcal{N}_i}n_j}\mapsto\mathbb{R}$ is strongly convex with parameter $\delta_i>0$.

Assumption 2. The constraint block set $\mathcal{X}_i\subset\mathbb{R}^{n_i}$ is nonempty compact and convex, and the relatively feasible interior $rint(\mathcal{X}_i)$ is nonempty.

Remark 1. Assumptions 1 and 2 ensure the solvability of distributed partitioned optimization problem (5), and it is determined that the sum of local objective function g_i obtains a unique optimal solution at some $\left(x_i^*,[x_j^*]_{j\in\mathcal{N}_i}\right)$ within the feasible set. The nonemptiness of the relative interior of $rint(\mathcal{X}_i)$ ensures the strong duality of the dual method.

Considering a meticulous treatment of problem (4) in a distributed manner, the notation $R_ix\in\mathbb{R}^n$ is introduced, which is regarded as a local memory of the overall decision information stored at the i -th vertex, and all the memories should be equal. Then, problem (4) can be equivalently presented by the following distributed form:

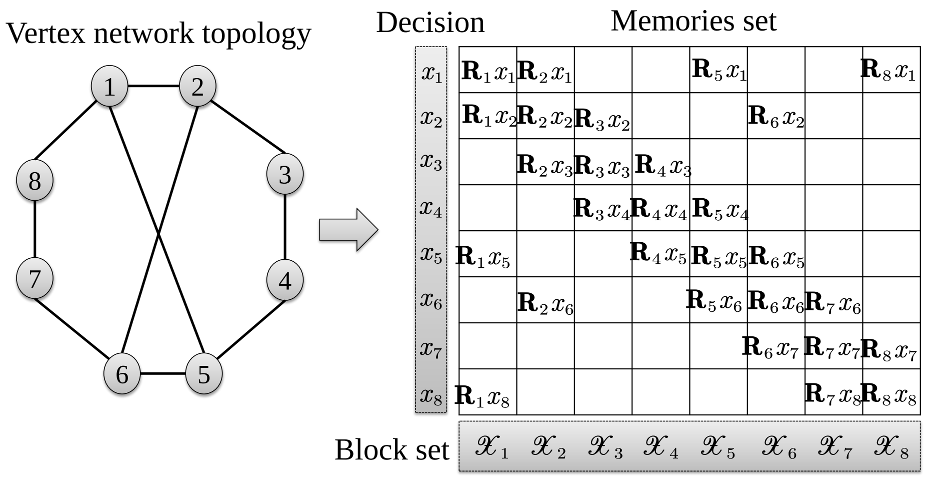
$$\begin{aligned} &\text{minmize} \quad \sum_{i=1}^{N}\tilde{f}_i(R_ix) \\ &\text{subjectto} \quad \mathrm{R}_ix\in X_i,\ i\in\{1,\cdots,N\} \\ &\qquad\qquad\quad \mathrm{R}_ix=R_jx,\ (i,j)\in\mathcal{E}. \end{aligned} \tag{6}$$

If the variable partition details are performed directly according to the idea of problem (5), due to the special structure of f_i and X_i, problem (4) appears rather unwieldy to solve. Our idea is to modify Equation (6) with the help of the partitioned structure to strictly grasp the range of equivalence relations between the decision variables, and then limit their diffusion in the communication graph.

The local memory repository with respect to the overall decision information is created for the i -th vertex, and the partitioned decision variable x_i corresponding to the i -th vertex is stipulated to be consistent only with its neighbors. Thus, problem (5) can be rewritten as follows:

$$\begin{aligned}
&\text{minmize} \quad \sum_{i=1}^{N} \tilde{g}_i\left(R_i x_i, \left[R_i x_j\right]_{j\in\mathcal{N}_i}\right) \\
&\text{subjectto} \quad \left(R_i x_i, \left[R_i x_j\right]_{j\in\mathcal{N}_i}\right) \in \mathcal{X}_i \times \prod_{j\in\mathcal{N}_i} \mathcal{X}_j\,,\ i \in \{1,\cdots,N\} \\
&\qquad \mathrm{R}_i x_i = R_j x_i,\ R_i x_j = R_j x_j,\ j \in \mathcal{N}_i
\end{aligned} \tag{7}$$

in which $\left[R_i x_j\right]_{j\in\mathcal{N}_i} \in \mathbb{R}^{\sum_{j\in\mathcal{N}_i} n_j}$ represents the local memory of the partitioned variable $x_j (j \in \mathcal{N}_i)$ of the j -th vertex stored in the i -th vertex, and $R_i x_i \in \mathbb{R}^{n_i}$ represents the memory of the i -th vertex for its own decision. **Figure 1** illustrates the vertex network topology and the distribution of memories of its partitioned decision variables. It can be seen from the vertical column that the memory information $R_i x_i \cup \left\{R_j x_i : j \in \mathcal{N}_i\right\}$ depends on the i -th vertex, and along the horizontal row it can be found that the memory of individual decision $R_i x_j$ or $R_i x_j (j \in \mathcal{N}_i)$ is related to the corresponding block set $\mathcal{X}_i$.



Vertex network topology
1
2
3
4
5
6
7
8
Decision
Memories set
x_1 $\mathbf{R}_1x_1$ $\mathbf{R}_2x_1$ $\mathbf{R}_5x_1$ $\mathbf{R}_8x_1$
x_2 $\mathbf{R}_1x_2$ $\mathbf{R}_2x_2$ $\mathbf{R}_3x_2$ $\mathbf{R}_6x_2$
x_3 $\mathbf{R}_2x_3$ $\mathbf{R}_3x_3$ $\mathbf{R}_4x_3$
x_4 $\mathbf{R}_3x_4$ $\mathbf{R}_4x_4$ $\mathbf{R}_5x_4$
x_5 $\mathbf{R}_1x_5$ $\mathbf{R}_4x_5$ $\mathbf{R}_5x_5$ $\mathbf{R}_6x_5$
x_6 $\mathbf{R}_2x_6$ $\mathbf{R}_5x_6$ $\mathbf{R}_6x_6$ $\mathbf{R}_7x_6$
x_7 $\mathbf{R}_6x_7$ $\mathbf{R}_7x_7$ $\mathbf{R}_8x_7$
x_8 $\mathbf{R}_1x_8$ $\mathbf{R}_7x_8$ $\mathbf{R}_8x_8$
Block set
$\mathcal{X}_1$ $\mathcal{X}_2$ $\mathcal{X}_3$ $\mathcal{X}_4$ $\mathcal{X}_5$ $\mathcal{X}_6$ $\mathcal{X}_7$ $\mathcal{X}_8$

Figure 1. Vertex network topology and partitioned memories.

Remark 2. Equation (7) intuitively presents the blockized configuration of distributed partitioned optimization problem (4). For a pair of vertices $(i,j) \in \mathcal{E}$, $R_i x_i = R_j x_i$ and $R_i x_j = R_j x_j$ need to be satisfied simultaneously, and this redundant structure is necessary, which provides inspiration for constructing block splitting optimization algorithms.

The notation $R_i \bar{x}$ is introduced to represent the aggregated form of memory stored by the i -th vertex about the decision information, both the neighbor vertices and itself, i.e., $R_i \bar{x} = \left(R_i x_i, \left[R_i x_j\right]_{j\in\mathcal{N}_i}\right)$, which makes the design and analysis of the algorithm more compact. Based on this representation, the form of problem (4) can be rewritten as

$$\begin{aligned} &\text{minmize} \quad \sum_{i=1}^{N} \tilde{g}_i(R_i\bar{x}) \\ &\text{subjectto} \quad \mathrm{R}_i\bar{x} \in \mathcal{X}_i \times \prod_{j\in\mathcal{N}_i} \mathcal{X}_j\,,\ i \in \{1,\cdots,N\} \\ &\qquad\qquad \mathrm{R}_j\bar{x} = R_i\bar{x}, \quad j \in \mathcal{N}_i \end{aligned} \tag{8}$$

where $R_j\bar{x} = \left(R_jx_i, \left[R_jx_j\right]_{j\in\mathcal{N}_i}\right)$. We adopt a strategy of dual decomposition by letting $\mathbf{\Theta}$ denote the stacking of all dual variables in the communication topology. Construct the following Lagrangian function with block splitting structure:

$$\begin{aligned} L(x,\Theta) = \sum_{i=1}^{N} \Bigg[&\tilde{g}_i\left(R_ix_i, \left[R_ix_j\right]_{j\in\mathcal{N}_i}\right) + \sum_{j\in\mathcal{N}_i} \Big[\left(R_{ij}\theta_i\right)^T\left(R_ix_i - R_jx_i\right) \\ &+ \left(R_{ij}\theta_j\right)^T\left(R_ix_j - R_jx_j\right)\Big]\Bigg] \end{aligned} \tag{9}$$

in which $\Theta = [\Theta_1^T,\cdots,\Theta_N^T]^T$ with each chunking $\Theta_i = \left(\left[R_{ij}\theta_i\right]_{j\in\mathcal{N}_i}, \left[R_{ij}\theta_j\right]_{j\in\mathcal{N}_i}\right)$. Based on the undirected and connectivity properties of communication graph, the split Lagrangian function (9) can be converted as follows:

$$\begin{aligned} L(x,\Theta) = \sum_{i=1}^{N} \Bigg[&\tilde{g}_i\left(R_ix_i, \left[R_ix_j\right]_{j\in\mathcal{N}_i}\right) + (R_ix_i)^T \sum_{j\in\mathcal{N}_i} \left(R_{ij}\theta_i - R_{ji}\theta_i\right) \\ &+ \sum_{j\in\mathcal{N}_i} \left(R_ix_j\right)^T\left(R_{ij}\theta_j - R_{ji}\theta_j\right)\Bigg] \end{aligned} \tag{10}$$

the local constraints $\left(R_ix_i, \left[R_ix_j\right]_{j\in\mathcal{N}_i}\right) \in \mathcal{X}_i \times \prod_{j\in\mathcal{N}_i} \mathcal{X}_j$ will not dualized in the split Lagrangian functions (9) and (10), they will be handed in the local optimization problem corresponding to the vertex. We define the Lagrangian dual function ψ_d to be the infimum of the Lagrangian function $L(x,\Theta)$ on the set $x \in X$, i.e.,

$$\psi_d(\Theta) = \inf_{x\in X} L(x,\Theta) = \sum_{i=1}^{N} \psi_{di}\left(\left[R_{ij}\theta_i, R_{ji}\theta_i, R_{ij}\theta_j, R_{ji}\theta_j\right]_{j\in\mathcal{N}_i}\right) \tag{11}$$

in which

$$\begin{aligned} &\psi_{di}\left(\left[R_{ij}\theta_i, R_{ji}\theta_i, R_{ij}\theta_j, R_{ji}\theta_j\right]_{j\in\mathcal{N}_i}\right) \\ &= \inf_{\left(R_ix_i,\left[R_ix_j\right]_{j\in\mathcal{N}_i}\right)} \Bigg[\tilde{g}_i\left(R_ix_i, \left[R_ix_j\right]_{j\in\mathcal{N}_i}\right) + (R_ix_i)^T \sum_{j\in\mathcal{N}_i} \left(R_{ij}\theta_i - R_{ji}\theta_i\right) \\ &\quad + \sum_{j\in\mathcal{N}_i} \left(R_ix_j\right)^T\left(R_{ij}\theta_j - R_{ji}\theta_j\right)\Bigg]. \end{aligned} \tag{12}$$

Since $\left(R_i x_i, \left[R_i x_j\right]_{j\in\mathcal{N}_i}\right) \in \mathcal{X}_i \times \prod_{j\in\mathcal{N}_i} \mathcal{X}_j$ and each $\mathcal{X}_i$ is compact and nonempty, the minimum of in (12) can be uniquely determined, and ψ_{di} is always finite. Then, the dual problem of (11) can be transformed into an unconstrained optimization problem as follows:

$$\text{maximize } \sum_{i=1}^{N} \psi_{di}\left(\left[R_{ij}\theta_i, R_{ji}\theta_i, R_{ij}\theta_j, R_{ji}\theta_j\right]_{j\in\mathcal{N}_i}\right). \tag{13}$$

Coming back to distributed optimization problem (7), we know that primal variables always follow

$$\begin{aligned}\left(R_i x_i, \left[R_i x_j\right]_{j\in\mathcal{N}_i}\right) \in -\underset{\left(x_i,[x_j]_{j\in\mathcal{N}_i}\right)}{argmin} \Bigg(g_i\left(x_i, \left[x_j\right]_{j\in\mathcal{N}_i}\right) + (x_i)^T \sum_{j\in\mathcal{N}_i} \big(R_{ij}\theta_i(t) \\ -R_{ji}\theta_i(t)\big) + \sum_{j\in\mathcal{N}_i} \left(x_j\right)^T \left(R_{ij}\theta_i(t) - R_{ji}\theta_i(t)\right)\Bigg)\end{aligned} \tag{14}$$

in which $\left(x_i, \left[x_j\right]_{j\in\mathcal{N}_i}\right) \in \mathcal{X}_i \times \prod_{j\in\mathcal{N}_i} \mathcal{X}_j$. Let $\tilde{g}_i^*$ be a conjugate of $\tilde{g}_i$, then

$$\begin{aligned}&\psi_{di}\left(\left[R_{ij}\theta_i, R_{ji}\theta_i, R_{ij}\theta_j, R_{ji}\theta_j\right]_{j\in\mathcal{N}_i}\right) \\ &= -\tilde{g}_i^*\left(\sum_{j\in\mathcal{N}_i} \left(R_{ij}\theta_i - R_{ji}\theta_i\right), \left[R_{ij}\theta_j - R_{ji}\theta_j\right]_{j\in\mathcal{N}_i}\right).\end{aligned} \tag{15}$$

It can be seen that the split Lagrangian dual function ψ_{di} does not depend on the whole set of dual variables Θ, but exhibits some sparsity and is only related to the dual variables of neighbor vertices.

3. Recurrent neural network approach

3.1. Basic framework

In the light of the analysis in Section 2, a science basic framework of RNN with block splitting and dual subgradient is proposed to solve distributed partitioned optimization problem (4). In the communication graph $\mathcal{G}$, each vertex performs the update and storage of the primal and dual variables driven by the recurrent neural network. Taking the i -th vertex as a reference, along with the activation and memory of the i -th neuron in neural network, it will wake up and update its own decision x_i and the decisions $\left[x_j\right]_{j\in\mathcal{N}_i}$ of its neighboring vertices, as well as the corresponding dual variables $R_{ij}\theta_i, \left[R_{ij}\theta_j\right]_{j\in\mathcal{N}_i}$. For any pair of vertices $(i,j) \in \mathcal{E}$, we design the following distributed algorithm based on the neuronal memory evolution function of RNN:

$$\left(R_i x_i(t+1), \left[R_i x_j(t+1)\right]_{j\in\mathcal{N}_i}\right) \in \left(R_i x_i(t), \left[R_i x_j(t)\right]_{j\in\mathcal{N}_i}\right) \tag{16}$$

$$-\partial_{\left(x_i,[x_j]_{j\in\mathcal{N}_i}\right)}\left(f_i\left(x_i,[x_j]_{j\in\mathcal{N}_i}\right)+(x_i)^T\sum_{j\in\mathcal{N}_i}\left(R_{ij}\theta_i(t)-R_{ji}\theta_i(t)\right)\right.$$
$$\left.+\sum_{j\in\mathcal{N}_i}(x_j)^T\left(R_{ij}\theta_i(t)-R_{ji}\theta_i(t)\right)\right)$$

$$R_{ij}\theta_i(t+1)=R_{ij}\theta_i(t)+\gamma_i\left(R_ix_i(t+1)-\ R_jx_i(t+1)\right) \tag{17}$$

$$R_{ij}\theta_j(t+1)=R_{ij}\theta_j(t)+\gamma_i\left(R_ix_j(t+1)-\ R_jx_j(t+1)\right) \tag{18}$$

where γ_i is the time step in RNN, which can be viewed as the interval between adjacent moments of the i -th neuron. The computational framework of RNN (16)–(18) is applicable to primal variables $\left(R_ix_i,[R_ix_j]_{j\in\mathcal{N}_i}\right)\in\mathbb{R}^{n_i}\times\mathbb{R}^{\sum_{j\in\mathcal{N}_i}n_j}$ and dual variables $\left(R_{ij}\theta_i,[R_{ij}\theta_j]_{j\in\mathcal{N}_i}\right)\in\mathbb{R}^{n_i}\times\mathbb{R}^{\sum_{j\in\mathcal{N}_i}n_j}$ with any initial states, whose evolution follows neuronal activity. **Figure 2** illustrates the architecture of the RNN (16)–(18), where the top half shows the evolution of the primal variable space under neuron induction, and the bottom half shows the evolutionary behavior of the dual variable space induced by neurons. It can be seen that the output state of the neuron at the current time is determined by the output state at the previous time and the input state at the current time. In this optimization model, each neuron corresponds to each vertex and has the ability to store objective and decision information.

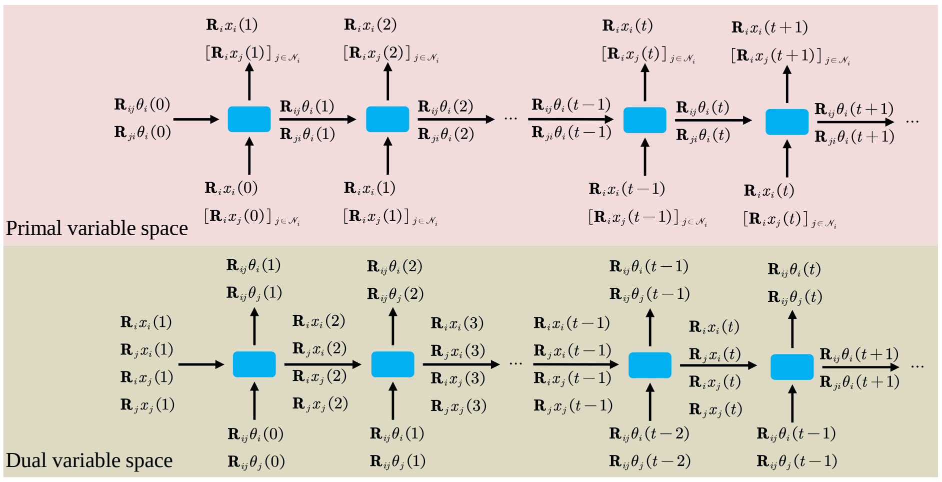

Primal variable space
$\mathbf{R}_{ij}\theta_i(0)$ $\mathbf{R}_{ji}\theta_i(0)$
$\mathbf{R}_ix_i(0)$ $[\mathbf{R}_ix_j(0)]_{j\in\mathcal{N}_i}$
$\mathbf{R}_ix_i(1)$ $[\mathbf{R}_ix_j(1)]_{j\in\mathcal{N}_i}$
$\mathbf{R}_{ij}\theta_i(1)$ $\mathbf{R}_{ji}\theta_i(1)$
$\mathbf{R}_ix_i(2)$ $[\mathbf{R}_ix_j(2)]_{j\in\mathcal{N}_i}$
$\mathbf{R}_{ij}\theta_i(2)$ $\mathbf{R}_{ji}\theta_i(2)$
$\mathbf{R}_{ij}\theta_i(t-1)$ $\mathbf{R}_{ji}\theta_i(t-1)$
$\mathbf{R}_ix_i(t-1)$ $[\mathbf{R}_ix_j(t-1)]_{j\in\mathcal{N}_i}$
$\mathbf{R}_ix_i(t)$ $[\mathbf{R}_ix_j(t)]_{j\in\mathcal{N}_i}$
$\mathbf{R}_{ij}\theta_i(t)$ $\mathbf{R}_{ji}\theta_i(t)$
$\mathbf{R}_ix_i(t+1)$ $[\mathbf{R}_ix_j(t+1)]_{j\in\mathcal{N}_i}$
$\mathbf{R}_{ij}\theta_i(t+1)$ $\mathbf{R}_{ji}\theta_i(t+1)$
Dual variable space
$\mathbf{R}_ix_i(1)$ $\mathbf{R}_jx_i(1)$ $\mathbf{R}_ix_j(1)$ $\mathbf{R}_jx_j(1)$
$\mathbf{R}_{ij}\theta_i(0)$ $\mathbf{R}_{ij}\theta_j(0)$
$\mathbf{R}_{ij}\theta_i(1)$ $\mathbf{R}_{ij}\theta_j(1)$
$\mathbf{R}_ix_i(2)$ $\mathbf{R}_jx_i(2)$ $\mathbf{R}_ix_j(2)$ $\mathbf{R}_jx_j(2)$
$\mathbf{R}_{ij}\theta_i(2)$ $\mathbf{R}_{ij}\theta_j(2)$
$\mathbf{R}_ix_i(3)$ $\mathbf{R}_jx_i(3)$ $\mathbf{R}_ix_j(3)$ $\mathbf{R}_jx_j(3)$
$\mathbf{R}_ix_i(t-1)$ $\mathbf{R}_jx_i(t-1)$ $\mathbf{R}_ix_j(t-1)$ $\mathbf{R}_jx_j(t-1)$
$\mathbf{R}_{ij}\theta_i(t-2)$ $\mathbf{R}_{ij}\theta_j(t-2)$
$\mathbf{R}_{ij}\theta_i(t-1)$ $\mathbf{R}_{ij}\theta_j(t-1)$
$\mathbf{R}_ix_i(t)$ $\mathbf{R}_jx_i(t)$ $\mathbf{R}_ix_j(t)$ $\mathbf{R}_jx_j(t)$
$\mathbf{R}_{ij}\theta_i(t)$ $\mathbf{R}_{ij}\theta_j(t)$
$\mathbf{R}_{ij}\theta_i(t+1)$ $\mathbf{R}_{ji}\theta_i(t+1)$

Figure 2. Architecture of RNN (16)–(18).

Remark 3. The design philosophy of RNN (16)–(18) does not impose strict requirements on the symmetry of the dual variables $R_{ij}\theta_i(t)=-R_{ji}\theta_i(t)$. When the time step size γ_i is chosen reasonably, the RNN algorithm can be simplified to only require one round of communication to perform local minimization and the update of the subgradient, which means that RNN (16)–(18) has better generality in solving distributed partitioned optimization problems (4).

Theorem 1. Assumptions 1–2 hold. If time stepγ_i in (17)–(18) is a constant satisfying $\gamma_i \in 0,1/N\xi_i$, with $\xi_i = \sqrt{2\sum_{j\in\mathcal{N}_i}\left(\frac{1}{\delta_i}+\frac{1}{\delta_j}\right)^2}$, the objective function g_i in problem (5) can converge to the optimal value $g_{i\text{opt}}$ by following the dual variable sequence $\{\Theta_1(t),\cdots,\Theta_N(t)\}$ induced by RNN (16)–(18), and the recurrent primal variable sequence $\{R_k x_i(t): k \in \{i\}\cup\mathcal{N}_i\}$ motivated by Equation (16) satisfies $\underset{t\to\infty}{lim}\|R_k x_i(t) - x_i^*\| = 0$ for any $i \in \{1,\cdots,N\}$, in which $x_i^*$is the i -th block of the unique optimal solution $x^* = [x_1^{*T},\cdots,x_N^{*T}]^T$.

Proof of Theorem 1. Based on Assumption 1, there exists the Lipschitz continuous subgradient $\frac{\partial\psi_d(\Theta)}{\partial R_{ij}\theta_i},\frac{\partial\psi_d(\Theta)}{\partial R_{ij}\theta_j}$ of the Lagrangian dual function ψ_d with splitting slackness. With the help of the conjugate function, for any $j \in \mathcal{N}_i$, there exist $v_i^* \in \partial\tilde{g}_i^*, v_j^* \in \partial\tilde{g}_j^*$ such that

$$\frac{\partial\psi_d(\Theta)}{\partial R_{ij}\theta_i} = [v_i^*]_i - [v_j^*]_i \tag{19}$$

$$\frac{\partial\psi_d(\Theta)}{\partial R_{ij}\theta_j} = [v_i^*]_j - [v_j^*]_j \tag{20}$$

where $[v_i^*]_i$ the i -th component of the subgradient v_i^*. It follows from the strong convexity of g_i that the subgradient of its conjugate function ∂g_i^* is Lipschitz continuous with $\frac{1}{\delta_i}$ [34] (Theorem 4.2.2). Combining Equations (19) and (20), we can obtain $\xi_i = \sqrt{2\sum_{j\in\mathcal{N}_i}\left(\frac{1}{\delta_i}+\frac{1}{\delta_j}\right)^2}$. Next, define the diagonal positive definite matrix $\Upsilon \triangleq \text{diag}\{\gamma_1,\cdots,\gamma_N\}$ with respect to the time step, where $\gamma_i \le \frac{1}{N\xi_i}$ for any $i \in \{1,\cdots,N\}$, then the neuronal groups evolving behavior with respect to the combination of dual variables can be described as

$$\Theta(t+1) = \Theta(t) + \Upsilon\eta\big(\Theta(t)\big) \tag{21}$$

where $\eta\big(\Theta(t)\big) \in \partial\psi_d\big(\Theta(t)\big)$. Since all diagonal elements satisfy $\gamma_i \in 0,\frac{1}{N\xi_i}$, then for any disturbance ζ, we have

$$\psi_d(\Theta(t)+\zeta) \ge \psi_d\big(\Theta(t)\big) + \eta\big(\Theta(t)\big)\zeta - \frac{1}{2}N\zeta^T\text{diag}\{\xi_1,\cdots,\xi_N\}\zeta. \tag{22}$$

On the basis of the proof of the optimality of the nonsmooth optimization problem and its dual form [35], (Theorem 2), it is known that a finite sequence $\{\Theta(t)\}$ will be generated to converge to the optimal solution Θ^* of problem (11). Meanwhile, Equation (21) can be split in the following components:

$$\Theta_i(t+1) \in \Theta_i(t) + \gamma_i\partial_{\Theta_i}\psi_d\big(\Theta(t)\big),\ \text{for } i \in \{1,\cdots,N\}. \tag{23}$$

According to the method of conjugate functions, the solution of the primal minimization problem (2) can be obtained by computing $\partial\tilde{g}_i^*$ at the point $\left(\sum_{j\in\mathcal{N}_i}\left(R_{ij}\theta_i - R_{ji}\theta_i\right), \left[R_{ij}\theta_j - R_{ji}\theta_j\right]_{j\in\mathcal{N}_i}\right)$, and for any $j \in \mathcal{N}_i$, one obtains

$$\frac{\partial\psi_d\left(\Theta(t)\right)}{\partial R_{ij}\theta_i(t)} = R_i x_i(t+1) - R_j x_i(t+1) \tag{24}$$

$$\frac{\partial\psi_d\left(\Theta(t)\right)}{\partial R_{ij}\theta_j(t)} = R_i x_j(t+1) - R_j x_j(t+1). \tag{25}$$

Thus, the evolutionary behavior of the dual neuronal groups can be viewed as the scaled gradient ascent (23). We know from Assumption 2 that the strong duality between the optimization problems (7) and (13) holds, and there is an equivalence relation between optimization problems (7) and (5), then the optimal value $\psi_{di_{\text{opt}}} = g_{i\text{opt}}$ holds. Hence, the objective function g_i in problem (5) can converge to the optimal value $g_{i\text{opt}}$ by following the dual variable sequence $\{\Theta_1(t), \cdots, \Theta_N(t)\}$ driven by RNN (16)–(18).

By Assumption 1, we know that distributed optimization problem (5) has a unique optimal solution $x^* = [x_1^{*T}, \cdots, x_N^{*T}]^T$, and it follows from the equivalence that x^* is also the unique optimal solution to problem (7). Moreover, the first-order optimality condition for the dual problem is given by $0 \in \partial\psi_d(\Theta^*)$, in which $\Theta^* = \left[\Theta_1^{*T}, \cdots, \Theta_N^{*T}\right]^T$ denotes a limit point of the sequence $\{\Theta(t)\}$, and Θ_i^* can be regarded as an equilibrium state of system (23). It obtains from (24) and (25) that there exists a limit point $\left(R_i x_i^*, \left[R_i x_j^*\right]_{j\in\mathcal{N}_i}\right)$ of the neural memory sequence of the primal variable, and according to the evolution behavior of this sequence, the neuron state x_i can search for the unique optimal solution x_i^* in problem (5). This completes the proof.

3.2. Asynchronous RNN (AsyRNN)

In terms of the execution power of the RNN algorithm (16)–(18), we incorporate the asynchronous evolution mechanism. The activated neurons perform information updating and transmission during each evolution of the RNN, whereas the dormant neurons wait to be triggered by a local timer or neighborly information, and finally, the decision information of each vertex on the time series of the RNN can reach the optimal state. Specifically, each neuron in the RNN is assigned a local timer, and the timer of each neuron is random and independent. When the i -th neuron is dormant, it continues to receive information from its neighbors until it is activated by its local timer or by information from neighboring neurons. After being activated, the i -th neuron updates its local variables and transmits the updated information to other neighbors. The timer is established on a local timer $\upsilon_i \in \mathbb{R}_+$ and a randomly generated waiting time T_i. The i -th timer activates the i -th neuron when $\upsilon_i = T_i$, so that the i -th neuron enters the activated mode. After performing the local evolution, the local time resets $\upsilon_i = 0$ and the next waiting time T_i is randomly generated. The waiting times between consecutive triggers are independent and identically distributed random variables that follow the same exponential distribution.

Specifically, when the i -th neuron is dormant, it continuously receives information from its activated neighbors. If the local timer υ_i is triggered or receives new dual information $R_{ji}\theta_i$ and $R_{ji}\theta_j$, then the i-th neuron is activated to perform the update and transmission of the primal variable $R_i\bar{x} = \left(R_i x_i, \left[R_i x_j\right]_{j\in\mathcal{N}_i}\right)$. If the activation state is caused by the local timer triggering of the i -th neuron itself, the update and transmission work is performed on the local dual information $R_{ij}\theta_i$ and $R_{ij}\theta_j$. Considering that there is no global timer, denote by $R_i x_i^{\blacktriangle}$, $R_{ij}\theta_i^{\blacktriangle}$ the variables for which the i -th neuron performs an update in the activated state, while by default denote by $R_i x_i$, $R_{ij}\theta_i$ the variables whose updates are not performed. We assign a state $\left(R_i x_i, \left[R_i x_j\right]_{j\in\mathcal{N}_i}\right)$ and a dual state $\left(\left[R_{ij}\theta_i\right]_{j\in\mathcal{N}_i}, \left[R_{ij}\theta_j\right]_{j\in\mathcal{N}_i}\right)$ to each neuron, and design the asynchronous RNN (AsyRNN) algorithm as follows:

Algorithm 1 Asynchronous RNN (AsyRNN)

```
1:  Preset. Set $\upsilon_i = 0$ and randomly generate a waiting time $T_i$.
2:  Evolution. Dormancy Phase: While $\upsilon_i < T_i$ do
3:  Receive $R_i x_i, R_i x_j$ and $R_{ij}\theta_i, R_{ij}\theta_j$ from the $j(j \in \mathcal{N}_i)$-th neuron.
4:  If $R_{ij}\theta_i, R_{ij}\theta_j$ are received switch to Activation Phase.
    Activation Phase: Calculation and transinformation
5:  (26)
6:  If $\upsilon_i = T_i$
7:  Then calculation and transinformation
    (27), (28)
8:  Set $\upsilon_i = 0$ and randomly generate a new waiting time $T_i$.
9:  Switch to Dormancy Phase.
```

$$\begin{aligned}\left(R_i x_i^{\blacktriangle}, \left[R_i x_j^{\blacktriangle}\right]_{j\in\mathcal{N}_i}\right) \in \left(R_i x_i^{\blacktriangle}, \left[R_i x_j^{\blacktriangle}\right]_{j\in\mathcal{N}_i}\right) \\ - \partial_{\left(x_i, \left[x_j\right]_{j\in\mathcal{N}_i}\right)} \Bigg(f_i\left(x_i, \left[x_j\right]_{j\in\mathcal{N}_i}\right) + (x_i)^T \sum_{j\in\mathcal{N}_i} \left(R_{ij}\theta_i^{\blacktriangle}(t) - R_{ji}\theta_i(t)\right) \\ + \sum_{j\in\mathcal{N}_i} \left(x_j\right)^T \left(R_{ij}\theta_i^{\blacktriangle}(t) - R_{ji}\theta_i(t)\right)\Bigg)\end{aligned} \tag{26}$$

$$R_{ij}\theta_i^{\blacktriangle} = R_{ij}\theta_i + \gamma_i\left(R_i x_i^{\blacktriangle} - R_j x_i\right) \tag{27}$$

$$R_{ij}\theta_j^{\blacktriangle} = R_{ij}\theta_j + \gamma_i\left(R_i x_j^{\blacktriangle} - R_j x_j\right) \tag{28}$$

Figure 3 shows the evolution mechanism of AsyRNN (26)–(28) in the primal and dual variable spaces under the dormant and activation phases. The dormancy and activation states of the i -th neuron are switched by local timers and local information without any central timer. The time taken for the computation of the dormancy phase is negligible compared to that of the activation phase. Besides, a constant local step γ_i is employed in the ascent step, which can be initialized by local information exchange between neighboring neurons. Each neuron is performing each computation by the latest value available locally.

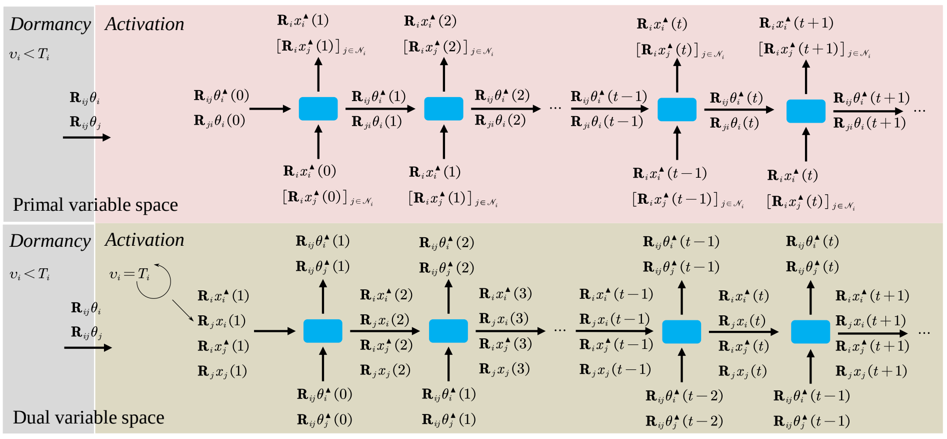


Figure 3. Architecture of AsyRNN (26)–(28).

Remark 4. Relying on the designed asynchronous update strategy triggered by local timers, AsyRNN (26)–(28) can better achieve steady state of gradient flow during training and prevent the vanishing and exploding gradient problems, which is due to the uncoordinated and more gradual propagation of neuron evolution throughout the network. Neurons adapt more flexibly to large or highly dynamic network environments based on their sensitivity to current inputs and local information.

Remark 5. For randomly generated updates, the time-sensitive performance feedback of AsyRNN (26)–(28) can adjust the local firing frequency and memory of the neurons to better adapt to the task requirements. Moreover, the inherent predictive and approximation capabilities of RNN enable it to hold a coexistence attitude towards time-sensitive decisions and effectively handle synchronization among neurons.

Theorem 2. Assumptions 1–2 hold. The local timer v_i is constructed following the principle of AsyRNN (26)–(28), and the time stepγ_i in (27)–(28) is a constant satisfying $\gamma_i \in 0{,}1/\xi_i$, in which ξ_i is given in Theorem 1. Then, the objective function g_i in problem (5) can converge to the optimal value $g_{i\mathrm{opt}}$ by following the dual sequence $\{\Theta(t)\}$ generated by AsyRNN (26)–(28). Moreover, for target confidence $\rho \in (0,1)$ and $\epsilon \in (0, \psi_{d0})$ with$\psi_{d0} \triangleq \psi_d\big(\Theta(0)\big)$, there exists a relevant time parameter $\tau(\epsilon,\rho)$ such that for any $t \geq \tau(\epsilon,\rho)$, the Lagrangian dual function $\psi_d\big(\Theta(t)\big)$ satisfies $\Pr\Big(\big|\psi_d\big(\Theta(t)\big) - \sum_{i=1}^{N} g_{i\mathrm{opt}}\big| \leq \epsilon\Big) \geq 1-\rho$.

Proof of Theorem 2. The dual variable Θ is split and assigned to N neurons to perform the update, and the variable component driven by the i-th neuron at time l is denoted as Θ_{i_l}. The evolution of AsyRNN (26)–(28) at each moment involves only one variable component, that is, Θ_{i_l} is updated at time l, while all other components $\Theta_j (j \neq i_l)$ maintain their state at the previous time. Here the evolution of the dual information is as follows:

$$\Theta_{i_l}(t+1) \in \Theta_{i_l}(t) + \partial_{\Theta_{i_l}} \psi_d\big(\Theta(t)\big) \tag{29}$$

$$\Theta_j(t+1) = \Theta_j(t),\ j \neq i_l. \tag{30}$$

Given that the timers υ_i are triggered independently following the same exponential distribution, from a global view, each evolution of the AsyRNN (26)–(28) involves only one neuron being randomly, uniformly, and independently awoke from its previous state. Therefore, each triggering induces an evolution of the AsyRNN (26)–(28), labeled by l, while prompting the communication vertices in the corresponding distributed optimization problem perform the update and transmission tasks.

After the i-th neuron is activated, it will perform the update process with its own primal variable information $R_i x_i$ and $R_i x_j (j \in \mathcal{N}_i)$, which are explicitly updated because the index i is the variable that updated them. Meanwhile, the i -th neuron will also store and use the variable information $R_j x_i$ and $R_j x_j (j \in \mathcal{N}_i)$ received from its neighbors $j \in \mathcal{N}_i$. These variables perform the update process with the index j if the j -th neuron itself or one of its neighbors is activated, and then the j -th neuron sends the updated variable to its neighbors, which include the i -th neuron. The update process for the dual variables follows a similar pattern.

Since $R_{i_l j} \theta_{i_l}$ and $R_{i_l j} \theta_j \left(j \in \mathcal{N}_{i_l}\right)$ are the components of Θ_{i_l}, and the index i_l follows a random uniform distribution, it follows from the modification of the index i_l the evolution Equations (29) and (30) correspond to the updating Equations (27) and (28). On the basis of Theorem 1, for the unconstrained optimization problem (13), it follows from block splitting method that there exists the Lipschitz continuous subgradient of the objective function ψ_d with respect to block Θ_{i_l}. By invoking [36] (Theorem 5), it can be known that the evolutionary behavior of neurons following (26)–(28) can search for the optimal value $\psi_{di\mathrm{opt}}$ of the local objective function in problem (13) with a high probability. Next, $\psi_{di\mathrm{opt}} = g_{di\mathrm{opt}}$ is known by virtue of the strong duality of problems (5) and (13). This completes the proof.

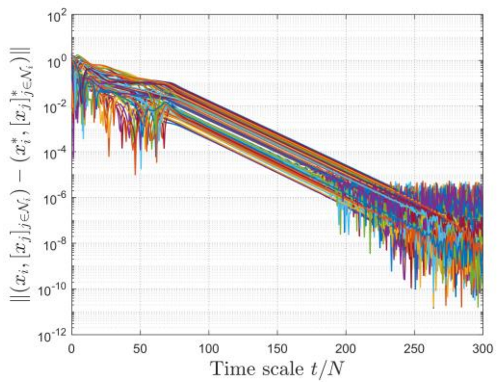
4. Numerical instances

4.1. Distributed network resource allocation

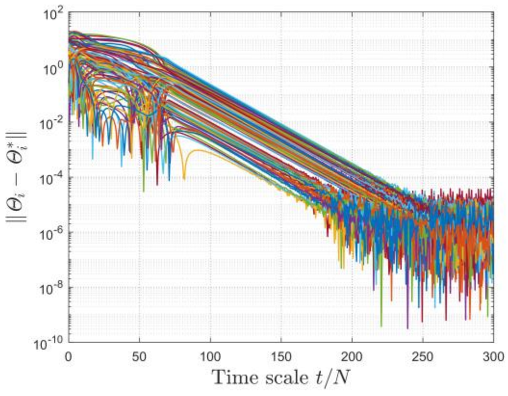
Example 1. Firstly, we consider an instance of resource allocation enjoying the partitioned property over a transmission network $\mathcal{G} = (\mathcal{V}, \mathscr{E})$ with $|\mathcal{V}| = 100$ [37], where the local objective function $g_i\left(x_i, [x_j]_{j\in\mathcal{N}_i}\right)$ is a quadratic function, and the set of local constraints $\left\{\left(x_i, [x_j]_{j\in\mathcal{N}_i}\right) : \left(x_i, [x_j]_{j\in\mathcal{N}_i}\right) \in \mathcal{X}_i \times \prod_{j\in\mathcal{N}_i} \mathcal{X}_j\right\}$ is expressed by linearity. The distributed partitioned optimization is formally described as follows:

$$\begin{aligned} \text{minmize} \quad & \sum_{i=1}^{N} \left(x_i, [x_j]_{j\in\mathcal{N}_i}\right)^T P_i \left(x_i, [x_j]_{j\in\mathcal{N}_i}\right) + \alpha_i \| \left(x_i, [x_j]_{j\in\mathcal{N}_i}\right) - \varpi_i \| \\ \text{subjectto} \quad & A_i \left(x_i, [x_j]_{j\in\mathcal{N}_i}\right) \preccurlyeq b_i, i \in \{1, \cdots, N\} \end{aligned} \tag{31}$$

where the decision variable $x_i \in \mathbb{R}^{n_i}$ and n_i is uniformly selected from $\{1,2,3,4\}$, P_i is a positive definite matrix with eigenvalues uniformly generated by $[1,5]$, α_i is uniformly generated in $[0,1]$, and each entry of the vector ϖ_i is chosen randomly from $[0,10]$. Each pair (A_i, b_i) characterizes a linear constraint with the number of rows selected uniformly from $\{1,2\}$. The entries of each A_i follow a standard Gaussian distribution $\mathcal{N}(0,1)$, while b_i are generated in accordance with feasible linear constraints. In the algorithm execution, each time step $\gamma_i = \frac{1}{\xi_i}$. The initial states of all dual variables are set to zero vectors. Given the asynchronous nature of the algorithm, we normalize the time t with respect to the number of vertices N and obtain the time scale $\frac{t}{N}$. **Figure 4a**,**b** show the error evolution curves of primal and dual variables for each vertex driven by AsyRNN (26)–(28), respectively. In **Figure 5a**, the convergence of the AsyRNN algorithm can be obtained from the difference between the dual functions ψ_{id} at time scale $\frac{t}{N}$ and the optimal value $\psi_{di\text{opt}} = g_{di\text{opt}}$ of problem (5). To show the value of the proposed AsyRNN (26)–(28) in solving distributed partitioned optimization, we introduce two other RNN models [21,23] with dynamic evolution capability as a comparison and perform technical block splitting on the neuronal states inside them in order to adapt to problem (5). **Figure 5b** shows that our proposed AsyRNN algorithm possesses better optimization convergence performance on the same time scale.

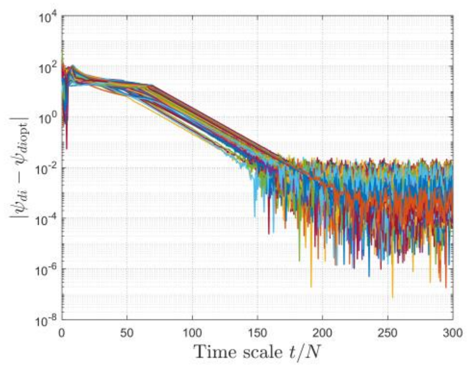


(a)

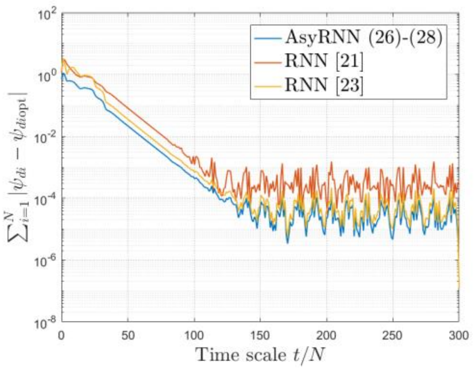


(b)

Figure 4. Error evolution of variables in Example 1. **(a)** Error evolution curves of primal variables guided by AsyRNN (26)–(28) in Example 1; **(b)** error evolution curves of dual variables guided by AsyRNN (26)–(28) in Example 1.



(a)



(b)

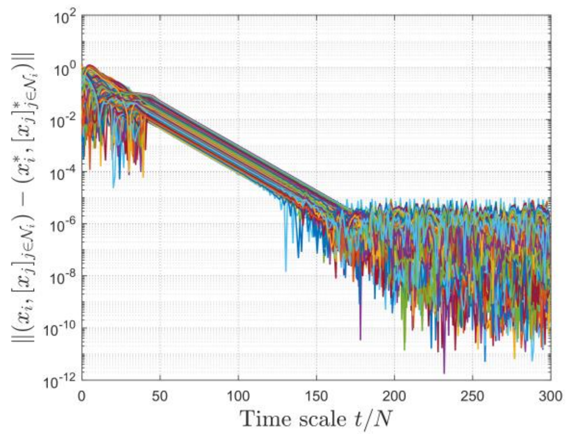
Figure 5. Error evolution and comparison of average convergence rates on dual functions in Example 1. **(a)** Error evolution curves of dual functions guided by AsyRNN (26)–(28) in Example 1; **(b)** Average convergence rates of dual functions driven by AsyRNN (26)–(28), RNN [21] and RNN [23] in Example 1.

4.2. Distributed elastic-net problem with square-root loss

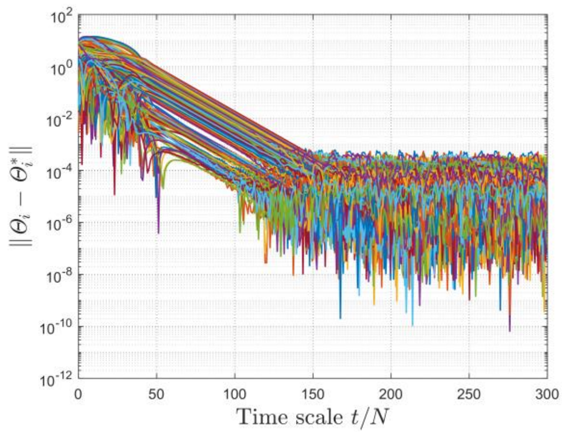
Example 2. This example presents a distributed elastic-net problem with square-root loss and partition [38], reified as follows:

$$\begin{aligned} \text{minmize} \quad & \sum_{i=1}^{N} \Big(\| B_i \left(x_i, [x_j]_{j \in \mathcal{N}_i} \right) - c_i \| + \frac{\sigma_1}{2} \| \left(x_i, [x_j]_{j \in \mathcal{N}_i} \right) \|^2 \\ & + \sigma_2 \| \left(x_i, [x_j]_{j \in \mathcal{N}_i} \right) \|_1 \Big) \end{aligned} \tag{32}$$

where $\left\| B_i \left(x_i, [x_j]_{j \in \mathcal{N}_i} \right) - c_i \right\|$ is a nonsmooth term, the previous algorithms such as FISTA [39] are not applicable, and $\frac{\sigma_1}{2} \left\| \left(x_i, [x_j]_{j \in \mathcal{N}_i} \right) \right\|^2 + \sigma_2 \left\| \left(x_i, [x_j]_{j \in \mathcal{N}_i} \right) \right\|_1$ is a strongly convex term. In this example, the number of vertices in distributed elastic-net is N=300. The entries of each $B_i \in \mathbb{R}^{p_i \times \left(n_i + \sum_{j \in \mathcal{N}_i} n_j \right)}$ follow a standard Gaussian distribution $\mathcal{N}(0,1)$, and then it is normalized by $\frac{1}{\sqrt{p_i}}$, where p_i and n_i are uniformly selected from $\{1,2,3,4\}$. Given a sparse vector $\left(x_i, [x_j]_{j \in \mathcal{N}_i} \right)^{\#}$ involving nonzero entries sampling from the standard Gaussian distribution $\mathcal{N}(0,1)$ as the true parameter vector, then the observation measurement can be expressed as $c_i = B_i \left(x_i, [x_j]_{j \in \mathcal{N}_i} \right)^{\#} + \varsigma \mathcal{N}(0,1)$, in which $\varsigma = 10^{-3}$ denotes the parameter in the noisy environment. $\sigma_1 > 0$ and $\sigma_2 > 0$ are two regularization parameters, which are set to r_1=0.1,r_2=0.01 in this example. In the AsyRNN (26)–(28), the time step of the i -th neuron is fixed as $\gamma_i = \frac{1}{\xi_i}$, and the initial states of all dual variables are set to zero vectors. **Figure 6a,b** show the error evolution curves of primal and dual variables over time scale $\frac{t}{N}$ guided by AsyRNN (26)–(28), respectively. We can intuitively see that the convergence accuracy of the primal and dual variables of each vertex under the guidance of the neuron is not the same, and the convergence rate shows the fusion of linear and sublinear. **Figure 7a** displays the error evolution on the dual function over time scale. It can be seen that the error on the dual function corresponding to some vertices can reach below 10^{-10} at a certain time scale, which is due to the asynchronous transmission mechanism that makes the activation time and frequency of some vertices different. Besides, in order to reflect the advancement of the designed AsyRNN model, we use the previous two RNN models [21,23] for comparison experiments, where the evolution of neuronal groups should be divided into blocks to match the partition structure of the distributed optimization problem. The results in **Figure 7b** show that the proposed AsyRNN algorithm has a better average convergence performance for solving problem (32).

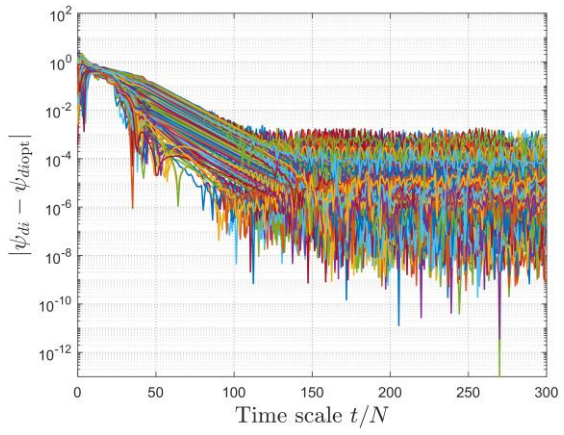


(a)

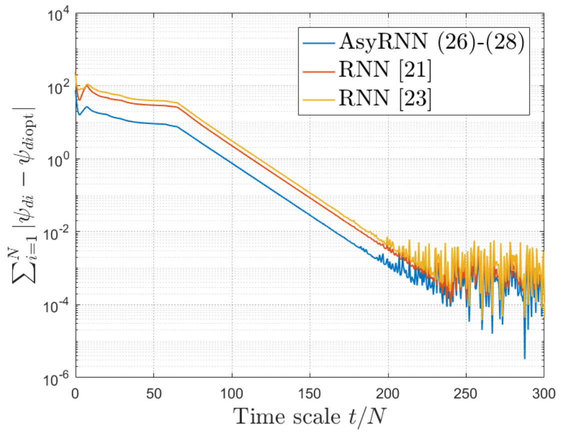


(b)

Figure 6. Error evolution of variables in Example 2. **(a)** Error evolution curves of primal variables guided by AsyRNN (26)–(28) in Example 2; **(b)** error evolution curves of dual variables guided by AsyRNN (26)–(28) in Example 2.



(a)



(b)

Figure 7. Error evolution and comparison of average convergence rates on dual functions in Example 2. **(a)** Error evolution curves of dual functions guided by AsyRNN (26)–(28) in Example 2; **(b)** average convergence rates of dual functions driven by AsyRNN (26)–(28), RNN [21] and RNN [23] in Example 2.

Remark 6. The simulation results of the above two examples show that the proposed AsyRNN (26)–(28) algorithm is feasible and effective for solving some specific distributed partitioned optimization problems. Neurons are allocated to all vertices one by one for optimization guidance, and the calculation and memory storage of a single neuron do not depend on the scale of the entire vertex network. The probability conclusion of the Lagrangian function is transplanted by the results in [36], and the verification on specific performance needs to be further analyzed, which will be the future research work.

5. Conclusion

In this paper, a recurrent neural network algorithm based on dual subgradient and block splitting has been presented for a class of distributed partitioned optimization problems. We have constructed a Lagrangian dual function enjoying block structure and make all primal and dual variables follow the guidance of the neuron. In the switching mechanism of activation and dormancy state of neurons, we have introduced local timers for asynchronous responses, which are of great benefit to save computing resources. Finally, the convergence of the proposed AsyRNN has been proved theoretically and verified by numerical simulations. The future work is to continue to explore the probability of convergence accuracy while considering the application of the proposed AsyRNN model in big data optimization.

Author contributions: Conceptualization, JL and JP; methodology, JL; software, JL; validation, CZ, YH and HW; formal analysis, CZ; investigation, YH; resources, JL; data curation, HW; writing—original draft preparation, JL; writing—review and editing, AM; visualization, CZ; supervision, JP; project administration, JL; funding acquisition, JL. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Natural Science Foundation of China under Grants 62406044 and 61932006; in part by the Guangxi Science and Technology Program under Grant AD23023001; in part by the Natural Science Foundation of Chongqing under Grant CSTB2024NSCQ-MSX1087; in part by the Postdoctoral Fellowship Program of CPSF under Grant GZB20230092; in part by the China Postdoctoral Science Foundation under Grant 2023M740383; and in part by the Research Foundation of Chongqing University of Science and Technology under Grants ckrc20231224 and 20231217. This research was partially supported by the Natural Science Foundation of China under Grants 62302068 and 62202071.

Acknowledgments: The authors would like to thank Jin-Song Dong of the School of Computing of NUS for helpful discussions on topics related to this work.

Conflict of interest: The authors declare no conflict of interest.

References

1. Best G, Cliff OM, Patten T, et al. Dec-MCTS: Decentralized planning for multi-robot active perception. The International Journal of Robotics Research. 2019; 38(2-3): 316-337.
2. Li B, Cen S, Chen Y, et al. Communication-efficient distributed optimization in networks with gradient tracking and variance reduction. Journal of Machine Learning Research. 2020; 21(180): 1-51.
3. Leng K, Li S. Distribution path optimization for intelligent logistics vehicles of urban rail transportation using VRP optimization model. IEEE Transactions on Intelligent Transportation Systems. 2021; 23(2): 1661-1669.
4. Wang X, Yang S, Guo Z, et al. A distributed dynamical system for optimal resource allocation over state-dependent networks. IEEE Transactions on Network Science and Engineering. 2022; 9(4): 2940-2951.
5. Cao Y, Sun B, Tsang DHK. Online network utility maximization: Algorithm, competitive analysis, and applications. IEEE Transactions on Control of Network Systems. 2022; 10(1): 274-284.
6. Romijn TCJ, Donkers MCF, Kessels JTBA, et al. A distributed optimization approach for complete vehicle energy management. IEEE Transactions on Control Systems Technology. 2018; 27(3): 964-980.
7. Li Q, Liao Y, Wu K, et al. Parallel and distributed optimization method with constraint decomposition for energy management of microgrids. IEEE Transactions on Smart Grid. 2021; 12(6): 4627-4640.
8. Cheng J, Liu Y, Ye Q, et al. DISCS: A distributed coordinate system based on robust nonnegative matrix completion. IEEE/ACM Transactions on Networking. 2016; 25(2): 934-947.
9. Choi KW, Aziz AA, Setiawan D, et al. Distributed wireless power transfer system for Internet of Things devices. IEEE Internet of Things Journal. 2018; 5(4): 2657-2671.

10. Zhang M, Zhang H, Yuan D, et al. Learning-based sparse data reconstruction for compressed data aggregation in IoT networks. IEEE Internet of Things Journal. 2021; 8(14): 11732-11742.
11. Scaman K, Bach F, Bubeck S, et al. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In: Proceedings of the 34th International Conference on Machine Learning (ICML'17); 6–11 August 2017; Sydney, Australia.
12. Kovalev D, Salim A, Richtárik P. Optimal and practical algorithms for smooth and strongly convex decentralized optimization. In: Proceedings of: Advances in Neural Information Processing Systems 33 (NeurIPS'20); 6-12 December 2020.
13. Necoara I, Clipici D. Parallel random coordinate descent method for composite minimization: Convergence analysis and error bounds. SIAM Journal on Optimization. 2016; 26(1): 197-226.
14. Notarnicola I, Carli R, Notarstefano G. Distributed partitioned big-data optimization via asynchronous dual decomposition. IEEE Transactions on Control of Network Systems. 2018; 5(4): 1910-1919.
15. Bastianello N, Carli R, Schenato L, et al. A partition-based implementation of the relaxed ADMM for distributed convex optimization over lossy networks. In: Proceedings of: 2018 IEEE Conference on Decision and Control (CDC); 17-19 December 2018; Miami, USA.
16. Liang XB, Wang J. A recurrent neural network for nonlinear optimization with a continuously differentiable objective function and bound constraints. IEEE Transactions on Neural Networks. 2000; 11(6): 1251-1262.
17. Xia Y, Feng G, Wang J. A novel recurrent neural network for solving nonlinear optimization problems with inequality constraints. IEEE Transactions on Neural Networks. 2008; 19(8): 1340-1353.
18. Liu Q, Wang J. A one-layer recurrent neural network for constrained nonsmooth optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics). 2011; 41(5): 1323-1333.
19. Cheng L, Hou ZG, Lin Y, et al. Recurrent neural network for non-smooth convex optimization problems with application to the identification of genetic regulatory networks. IEEE Transactions on Neural Networks. 2011; 22(5): 714-726.
20. Qin S, Xue X. A two-layer recurrent neural network for nonsmooth convex optimization problems. IEEE Transactions on Neural Networks and Learning Systems. 2014; 26(6): 1149-1160.
21. Li G, Yan Z, Wang J. A one-layer recurrent neural network for constrained nonconvex optimization. Neural Networks. 2015; 61: 10-21.
22. Xia Z, Liu Y, Kou KI, et al. Clifford-valued distributed optimization based on recurrent neural networks. IEEE Transactions on Neural Networks and Learning Systems. 2023; 34(10): 7248-7259.
23. Jia W, Qin S, Xue X. A generalized neural network for distributed nonsmooth optimization with inequality constraint. Neural Networks. 2019; 119: 46-56.
24. Rajesh P, Muthubalaji S, Srinivasan S, et al. Leveraging a dynamic differential annealed optimization and recalling enhanced recurrent neural network for maximum power point tracking in wind energy conversion system. Technology and Economics of Smart Grids and Sustainable Energy. 2022; 7(1): 19.
25. Liu J, Liao X, Dong JS. A recurrent neural network approach for constrained distributed fuzzy convex optimization. IEEE Transactions on Neural Networks and Learning Systems. 2024; 35(7): 9743-9757.
26. Sherstinsky A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena. 2020; 404: 132306.
27. Lin Y, Hu G, Wang L, et al. A multi-AGV routing planning method based on deep reinforcement learning and recurrent neural network. IEEE/CAA Journal of Automatica Sinica. 2024; 11(7): 1720-1722.
28. Parikh N, Boyd S. Block splitting for large-scale distributed learning. In: Proceedings of: Advances in Neural Information Processing Systems 24 (NeurIPS'11); 12-14 December 2011; Granada, Spain.
29. O'Connor M, Zhang G, Kleijn WB, et al. Function splitting and quadratic approximation of the primal-dual method of multipliers for distributed optimization over graphs. IEEE Transactions on Signal and Information Processing over Networks. 2018; 4(4): 656-666.
30. Latafat P, Freris NM, Patrinos P. A new randomized block-coordinate primal-dual proximal algorithm for distributed optimization. IEEE Transactions on Automatic Control. 2019; 64(10): 4050-4065.
31. Li H, Wu X, Wang Z, et al. Distributed primal-dual splitting algorithm for multiblock separable optimization problems. IEEE Transactions on Automatic Control. 2021; 67(8): 4264-4271.

32. Moradi S, Indiveri G. An event-based neural network architecture with an asynchronous programmable synaptic memory. IEEE Transactions on Biomedical Circuits and Systems. 2013; 8(1): 98-107.
33. Gaunt AL, Johnson MA, Riechert M, et al. AMPNet: Asynchronous model-parallel training for dynamic neural networks. arXiv preprint arXiv:1705.09786. 2017.
34. Urruty JBH, Lemaréchal C. Conjugacy in Convex Analysis. In: Convex analysis and minimization algorithms II: Advanced theory and bundle methods. Springer Science & Business Media; 1993. pp. 35-82.
35. Sen S, Sherali HD. A class of convergent primal-dual subgradient algorithms for decomposable convex programs. Mathematical Programming. 1986; 35: 279-297.
36. Richtárik P, Takáč M. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. Mathematical Programming. 2014; 144(1): 1-38.
37. Li JY, Du KJ, Zhan ZH, et al. Distributed differential evolution with adaptive resource allocation. IEEE Transactions on Cybernetics. 2022; 53(5): 2791-2804.
38. Tang P, Wang C, Sun D, et al. A sparse semismooth Newton based proximal majorization-minimization algorithm for nonconvex square-root-loss regression problems. Journal of Machine Learning Research. 2020; 21(226): 1-38.
39. Jiang H, Luo S, Dong Y. Simultaneous feature selection and clustering based on square root optimization. European Journal of Operational Research. 2021; 289(1): 214-231.