An RNN-Based Algorithm for Decentralized-Partial-Consensus Constrained Optimization

Zicong Xia, Graduate Student Member, IEEE, Yang Liu, Member, IEEE, Jianlong Qiu, Member, IEEE, Qihua Ruan, and Jinde Cao, Fellow, IEEE

Abstract—This technical note proposes a decentralized-partial-consensus optimization (DPCO) problem with inequality constraints. The partial-consensus matrix originating from the Laplacian matrix is constructed to tackle the partial-consensus constraints. A continuous-time algorithm based on multiple interconnected recurrent neural networks (RNNs) is derived to solve the optimization problem. In addition, based on nonsmooth analysis and Lyapunov theory, the convergence of continuous-time algorithm is further proved. Finally, several examples demonstrate the effectiveness of main results.

Index Terms—Decentralized-partial-consensus optimization (DPCO), nonsmooth analysis, partial-consensus matrix, recurrent neural networks (RNNs).

1. INTRODUCTION

In recent decades, distributed optimization has captured a major number of attention [1]–[5] due to its great potential applications [6]–[9]. The chief objective of distributed optimization is to design an algorithm to derive the optimal points to the optimization problem, in which each agent has personal information that is only known by itself. Moreover, as a new stage of system science development, the theory and method of complexity science provide a new idea, new method, and new way for the development of human beings, which has a good application prospect [10]. In this background, the methods for distributed optimization from multiagent theory and interconnected recurrent neural networks (RNNs) appeal tremendous interests on account of the theoretic significance related to many kinds of fields and applications: resource assignment in communication networks [8], [11], [12], multiairnet networks [3], [6], [13], multirobot motion planning [14], and machine learning [5], [9], [15], [16].

There are a large number of different kinds of methods to solve distributed problems (listed in Table I in Section IV). Since the mid-1980s, the neurodynamic optimization method based on RNNs has been widely investigated [17]. Furthermore, with the development of long short-term memory (LSTM) and gated recurrent units (GRUs) in the theory of RNNs, RNNs have become popular models that have shown great promise in many natural language processing tasks [18], [19]. Meanwhile, one of the possible and very promising approaches for real-time optimization is to apply RNNs based on circuit implementation. As parallel computational models for solving constrained optimization problems, RNNs have received a great deal of attention and brought a wide range of applications over the past few decades. Many existing RNN-based optimization methods with centralized cost functions have been developed [20]–[24]. Recent years have also witnessed the great development and improvement in RNNs for solving distributed optimization problems. For instance, Liu et al. [25] presented a collective neurodynamic approach with multiple interconnected RNNs for distributed optimization. Fan and Wang [26] gave a collective neurodynamic optimization method to nonnegative matrix factorization. Xia et al. [5] proposed the RNN-based algorithms to solve distributed optimization in quaternion field.

Various kinds of constraints have been investigated for optimization problems while considering different practical cases in real life. For example, for the resource allocation problem, the choice of each agent localizes in a certain range, while the agents may not want to share their private information with others [2]. Thus, the local constraints are considered. In social networks, the limitations of communication capacities for each agent should also cause the constraints. In addition, some engineering tasks involving construction ability, technical restrictions, and time limitations have more complex constraints [27]. In last few decades, several different constraints are considered: inequality constraints [28]–[30], equality constraints [12], [23], [30], [31], bound constraints [1], [23], local bound constraints [2], [4], and approximate constraints [32].

In large-scale optimization problems, the computation capacity of an agent is not enough to handle all the constraints of the agents because of the performance limitations of the agents in communication capacities as well as task requirements of privacy and security. Thus, distributed optimization gets compelling attention due to its larger capacities. Distributed optimization design is often necessary to be solved by the identical optimal point with the identical dimension for each cost function. However, in many cases, the minimizers of different cost functions may not share the same dimension. For instance, in a simple transportation problem, the configuration of different conveyances could be different, and hence, the optimal transport volume for each conveyance is not identical and is determined by distinct functions with respect to diverse variables. In this case, some variables should be identical such as the distance. The case mentioned above can be described as a decentralized-partial-consensus optimization (DPCO) problem, which has the same form as cost function in distributed optimization but optimal points with different dimensions and partial-consensus components. In order to be visual, Fig. 1 shows the core concept of three types of mentioned optimization. Due to the difference for the dimensions of optimal points, DPCO problems are more flexible and
suit for more practical problems. However, the existing works, based on RNNs, can not deal with DPCO problems, and this brief focuses on solving DPCO problems.

This brief aims at solving the DPCO problems by a continuous-time algorithm based on RNNs. The contributions of this brief are made as summarized in the following.

1) The model of DPCO problems is constructed, and a partial-consensus matrix is proposed to handling the partial-consensus constraints.

2) In order to solve DPCO problems, a continuous-time algorithm is designed based on multiple interconnected RNNs, and its convergence is proved by nonsmooth analysis and Lyapunov theory.

The remainder of this brief is arranged as follows. Section II introduces some necessary notations, definitions, and lemmas, including serial number subset and partial-consensus matrix. In Section III, the DPCO problem is formulated. A distributed continuous-time algorithm based on RNNs is presented. Section IV presents an example to demonstrate the efficiency of this brief. Finally, Section V marks a brief conclusion and proposes some directions for the research in the future.

II. PRELIMINARIES

A. Notations

$\mathbf{R}$ and $\mathbf{N}_+$ denote the real number set and the positive integer set, respectively. $\text{int}(\Omega)$ denotes the interior of the set $\Omega$. Define $A$ a matrix while $\delta_{\min}(A)$ and $\delta_{\max}(A)$ denote its smallest and largest eigenvalue, respectively. $A(i,j)$ is the $(i,j)$th element of $A$. $L_N$ denotes the $N$-dimensional Laplacian matrix. $\otimes$ denotes the Kronecker product. $|||·|||$ denotes the 2-norm. $\omega \subset \mathbf{N}_+$ is a finite set, $|\omega|$ denotes the cardinality of $\omega$, and $\partial \omega$ denotes the ordered set of $\omega$. Different from normal set, the order of ordered set cannot be changed. $\bar{\omega}^p$ is the $p$th entry of $\bar{\omega}$, $\bar{\omega}_i = \{i_1, \ldots, i_p\} \subset \mathbf{N}_+$ and $\bar{\omega}_j = \{j_1, \ldots, j_{m-p}\} \subset \mathbf{N}_+$ are two different ordered sets and all the entries are distinguished. The operator $\bigcup$ leads that $\bar{\omega}_i \bigcup \bar{\omega}_j = \{i_1, \ldots, i_p, j_1, \ldots, j_{m-p}\}$. For column vectors $x_1, \ldots, x_n$, define $\text{col}[x_1, \ldots, x_n] = [x_1^T, \ldots, x_n^T]^T$.

B. Serial Number Subset and Partial-Consensus Matrix

$v = \{1, \ldots, m\}$ denotes the serial number set of the components of an $m$-dimensional vector. $\nu_n \subset v$ denotes the serial number subset for the partial components of a vector. When we set $\nu_n = \{i_1, \ldots, i_n\}$ and $x = \text{col}[x_1, \ldots, x_m] \in \mathbf{R}^m (m \geq n)$, then vector $x^{(\nu_n)} := \text{col}[x_{i_1}, \ldots, x_{i_n}] \in \mathbf{R}^n$. Without loss of generality, we set $\nu_n = \{1, \ldots, n\}$ and $x^{(\nu_n)} = \text{col}[x_1, \ldots, x_n] \in \mathbf{R}^n$. Actually, the serial number subset is aimed at selecting some components to consist a new vector from a vector. Then, we give a set of vectors $\{x_i \in \mathbf{R}^n | i = 1, \ldots, N\}$ in which the dimensions of the vectors need not be identical. Let $x = \text{col}[x_1, \ldots, x_N]$ as superposition of $x_i (i = 1, \ldots, N)$. We set $n_{\min} = \min(n_i)$, and the serial number subsets with $n \leq n_{\min}$ are denoted as $\nu_n$. $N = \{1, \ldots, \sum_{i=1}^{N} n_i\}$ denotes the serial number set of the components of $x$. For convenience, we set $N = \sum_{i=1}^{N} n_i$.

Now, we propose a novel matrix, called partial-consensus matrix, to handle the partial-consensus constraints. First, we propose a matrix dimension extension method by the operator $E_{\bar{\omega}} : M^{p \times p} \rightarrow M^{(p+|\omega|) \times (p+|\omega|)}$. When assume a singleton set $\omega = \{t\}$ with $t \in \{1, \ldots, p\}$, $E_{\bar{\omega}}(M)$ denotes the matrix after adding a zero row and a zero column behind the $(t-1)$th row and column (when $t = 1$, add a zero row and a zero column in the first row and column). When assuming that $|\omega| > 1$, $E_{\bar{\omega}}(M) = \left[\begin{array}{c|c} E_{\bar{\omega}(\{t\})}(M_{\{t\}} & \cdots \cdots (E_{\bar{\omega}(\{t\})}(M_{\{t+\omega\}})) \end{array}\right]$. For the sake of understanding $E_{\bar{\omega}}$, we give a simple example: set $\bar{\omega} = \{1, 3\}$ and $M = \left[\begin{array}{cc} 2 & -1 \\ -1 & 2 \end{array}\right]$

then we have

$E_{\bar{\omega}}(M) = E_{\bar{\omega}(\{1\})} \left(E_{\bar{\omega}(\{3\})}(M) \right) = E_{\bar{\omega}(\{1\})} \left(\begin{array}{cc} 0 & 0 \\ 0 & -1 \end{array}\right)$

Assume that $\bar{V}_n = \bigcup_{j=1}^{N} N_{j} \setminus \bar{\nu}_n$ and $\nabla \nabla n = \bar{V} \setminus \bar{V}_n$. Now, we construct the partial-consensus matrix $K[\bar{\Omega}] = E_{\bigcup_{j=1}^{N} \bar{V}_n} \left(L_N \otimes I_n\right) \in M^{N \times N}$, where $\bar{\Omega} = \{x_i \in \mathbf{R}^n | i = 1, \ldots, N\}$. For saving notations, $\bar{\Omega}$ is adopted to represent $K[\bar{\Omega}]$. However, $\bar{\Omega}$ should be always considered for $K[\bar{\Omega}]$.

Lemma 1: Set $n_0 = 0$. If $K_n(i,j) \neq 0$, then $N_n \subset N_{n+1} \subset \mathbf{N}_+$ and $N_1 \subseteq N_2 \subseteq N$ such that $i = j = N_3 n$. According to the operator $E_{\nu_n}$, for $\sum_{i=1}^{N_3} n_i \leq i \leq \sum_{i=1}^{N_3} n_i$ and $\sum_{i=1}^{N_3} n_i \leq j \leq N_2 n$, $K_n(i,j) = L_N \otimes I_n \{i_0, j_0\}$, where $i_0 = i - \sum_{t=1}^{N_3} (n_t - n)$ and $j_0 = j - \sum_{t=1}^{N_3} (n_t - n)$. We can get the range of $i_0$ and $j_0$. Then, considering the range of $i_0$ and $j_0$, one can find that $N_1 = N_2 + N_3$. Next, $i_0 - j_0 = i - \sum_{t=1}^{N_3} (n_t - n) - (j - \sum_{t=1}^{N_3} (n_t - n)) = N_3 n$, and combining with $N_1 = N_2 + N_3$, we have $i - j = \sum_{t=1}^{N_3} n_t - \sum_{t=0}^{N_3} n_t = \sum_{t=0}^{N_3} n_t$. Then, we can derive a corollary from Lemma 1.

Corollary 1: Set $n_0 = 0$. If $\forall \{n_1, n_2 \subseteq N$ and $N_1 \subseteq N_2 \subseteq \nu_n$ such that $i \neq j \neq \sum_{i=1}^{N_3} n_i$, then $K_n(i,j) = 0$.

In order to deepen the understanding of Lemma 1 and Corollary 1, we give some notes and remarks.

Remark 1: 1) Corollary 1 is the converse negative proposition of Lemma 1. The function of Corollary 1 is to judge whether $K_n(i,j) = 0$. In addition, Corollary 1 indicates that when the difference of $i$ and $j$ is not the term like $\sum_{t=1}^{N_3} n_t - \sum_{t=0}^{N_3} n_t$, $K_n(i,j) = 0$. 

Fig. 1. Distinctions among three types of optimization. (a) Centralized optimization case. (b) Distributed optimization case. (c) DPCO case.
2) It should be noticed that $L_N \otimes I_n(i,j)$ denotes the $(i,j)$th element of matrix $L_N \otimes I_n$ and does not denote the Kronecker product of $L_N$ and $I_n(i,j)$.

Now, we give a lemma to illustrate the function of the partial-consensus matrix $K_n$.

**Lemma 2:** $K_n x = 0$ if and only if $x_i^{(\nu_k)} = x_j^{(\nu_k)}$, $i, j = 1, \ldots, N$.

**Proof:** Considering the equation $K_n x = 0$, it can be easily derived that $\sum_{i=1}^{N} K_n(i,j) x_j = 0$ for $i = 1, \ldots, N$. According to the properties of $L_N \otimes I_n$, we find that $K_n(i,i) = -\sum_{i=1}^{N} K_n(i,j)$. On account of Corollary 1, considering the case in which $i = \bar{V}_n^1$, we have

$$K_n \left( \bar{V}_n^1 \right) x_{\bar{V}_n^1} + K_n \left( \bar{V}_n^1, n_1 + \bar{V}_n^1 \right) x_{\bar{V}_n^{1+n_1}} + \cdots + K_n \left( \bar{V}_n^1, \sum_{t=1}^{N-1} n_t + \bar{V}_n^1 \right) x_{\bar{V}_n^{1+\sum_{t=1}^{N-1} n_t}} = 0.$$  \hspace{1cm} (1)

Then, choose $i = \bar{V}_n^{1+n_1}, \ldots, \bar{V}_n^{1+\sum_{t=1}^{N-1} n_t}$, and it leads to a group of equations from (1)

$$\sum_{j \neq 1}^{N-1} K_n \left( \bar{V}_n^1, \sum_{t=1}^{j-1} n_t + \bar{V}_n^1 \right) \left( \begin{array}{c} \bar{V}_n^1 - x_{\bar{V}_n^1} \end{array} \right) = 0 \hspace{1cm} \text{for } j = 1, \ldots, N-1.$$  \hspace{1cm} (2)

It is easy to observe that (2) guarantees $x_{\bar{V}_n^1} = \cdots = x_{\bar{V}_n^{1+\sum_{t=1}^{N-1} n_t}}$. Due to the construction of $\bar{V}_n$ and with the further derivation, one can find $\bar{V}_n^{1+\sum_{t=1}^{N-1} n_t} = \bar{V}_n^{1+\sum_{t=1}^{N-1} n_t} = \bar{V}_n^{1+\sum_{t=1}^{N-1} n_t}$, and then,

$$x_{\bar{V}_n^{1+\sum_{t=1}^{N-1} n_t}} = x_1^{(\nu_k)} = x_j^{(\nu_k)}.$$

Similarly, $x_1^{(P)} = \cdots = x_j^{(P)}$ for $p = 1, \ldots, n$, which indicates the result: $x_i^{(\nu_k)} = x_j^{(\nu_k)}$. On the contrary, when $x_i^{(\nu_k)} = x_j^{(\nu_k)}$, one can have that (2) holds, which can lead to $K_n x = 0$. ■

From Lemma 2, we obtain that the main function of $K_n$ is to make some parts of several vectors identical. Then, its properties originating from the Laplacian matrix are proved as follows.

**Lemma 3:**

1) $K_n$ is positive semidefinite.

2) $K_n$ has $nN$ nonnegative eigenvalues. $\delta_{\text{min}}(K_n) = \delta_{\text{min}}(L_N \otimes I_n) = 0$ and its multiplicity is $N - nN + 1$. $\delta_{\text{max}}(K_n) = \delta_{\text{max}}(L_N \otimes I_n)$.

**Proof:** The proof can be obtained by the characteristic polynomial of $K_n$, which is similar to one of $L_N \otimes I_n$. Thus, the proof is omitted. ■

To make the concept of $K_n$ clear, we give a simple example to explain how to construct a partial-consensus matrix from a group of vectors.

**Example 1:** We give a set of vectors

$$x_1 = \{x_{11}, x_{12}, x_{13}\}, \quad x_2 = \{x_{21}, x_{22}, x_{23}, x_{24}\}, \quad x_3 = \{x_{31}, x_{32}, x_{33}, x_{34}, x_{35}\}.$$  

We need to construct a partial-consensus matrix to achieve that $x_{1i} = x_{2j} = x_{3k}$ with $i = 1, 2, 3$. Define $\Omega = \{x_{11}, x_{12}, x_{13}\}$. Let $n_{\text{min}} = 3$ and $n = 3$, and hence, $\nu_3 = \{1, 2, 3\}$. $x = \text{col}[x_{11}, x_{12}, x_{13}]$ and $V = \{1, \ldots, 12\}$. Then, $V_3 = \{1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12\}$ and

$$x(V_3) = \{x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{31}, x_{32}, x_{33}\}.$$  

Select the Laplacian matrix

$$L_3 = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}.$$  

Then, $K_3$ can be constructed as follows, as shown at the bottom of the next page.

By solving the equation $K_3 x = 0$, we can have $x_{1i} = x_{2j} = x_{3k}$ with $i = 1, 2, 3$, and 3, which illustrates the effectiveness of the partial-consensus matrix.

From the perspective of topological structure diagram, we give more details for $K_n$. The topological structure of $L_3$ and $K_3$ is shown in Fig. 2. Fig. 2 shows the transformation process from Laplacian matrix to partial-consensus matrix from the perspective of graph theory. We regard the Laplacian matrix as an adjoin matrix of the graph (the state node graph at the top of Fig. 2). In addition, the state node graph at the bottom of Fig. 2 is a connected graph of the components of state vectors for $n$ agents. The partial-consensus matrix, after transforming from the Laplacian matrix, is the adjoin matrix of the connected graph. By using the partial-consensus matrix, the connected components will be consistent.

In Section II, let $\nu_\alpha \subset \nu$ denote the serial number subset of the partial components of a vector. Correspondingly, for a set of vectors $\Omega = \{x_{11}, x_{12}, \ldots, x_N\}$, the consensus parts are $x_{1j} = x_{2j} = \ldots, = \ldots$.
In essence, we can also choose any $n$ components of each vector $\in \Omega$ and just move the selected components to the top $n$ positions of each vector $\in \Omega$. To achieve the mentioned movement, we adopt the more general form of $v_n = [i_1, \ldots, i_n]$ and propose the definition of permutation matrix $A[v \setminus v_0]$, which can change $x \in \mathbb{R}^n$ into $\text{col}\{v^{(n)}, x^{(v \setminus v_0)}\}$. Moreover, in the remainder of this note, we just the case in $v = [1, \ldots, m]$. Thus, permutation matrix $A[v \setminus v_0]$ is not used, but it really plays the role of random selection of consensus parts.

**Lemma 4**: $\text{col}\{v^{(n)}, x^{(v \setminus v_0)}\} = A[v \setminus v_0]x$, where

$$A[v \setminus v_n]_{i,j} = \begin{cases} 1 & (i, j) = (\tilde{v}_p, \tilde{v}_p), p = 1, \ldots, m \\ 0 & \text{others} \end{cases}$$

with $\tilde{v} = v_n \setminus v_n$.

**Proof**: The construction of the permutation matrix is similar to the normal permutation matrix, and the proof is direct and hence is omitted.

A simple example for Lemma 4 is given: set $x = [x_1, x_2, x_3, x_4]$, $v = [1, 2, 3, 4]$, and $\tilde{v} = [2, 3, 4]$. Then, $\text{col}\{v^{(n)}, x^{(v \setminus v_0)}\} = [x_2, x_3, x_4, x_1]$. By Definition 4, we have

$$A[v \setminus \tilde{v}] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

Then, we calculate $A[v \setminus v_n]x = [x_2, x_3, x_4, x_1] = \text{col}\{v^{(n)}, x^{(v \setminus v_0)}\}$.

C. Projection and Nonsmooth Analysis in Convex Optimization

Let $\Omega$ be a convex set and $P_\Omega(x)$ denote the projection of $x$ onto $\Omega$, i.e., $P_\Omega(x) = \text{argmin}_{y \in \Omega} \|y - x\|$. The property holds [33]

$$\left( P_\Omega(x) - x \right)^T \left( P_\Omega(x) - x \right) \leq 0 \quad \forall x \in \mathbb{R}^n \forall x' \in \Omega. \quad (3)$$

From [33], cone($\Omega$) = $\{y : x \in \Omega, \gamma \geq 0\}$ denotes the convex cone of $\Omega$. cone($\Omega$)$^o$ is the polar cone of cone($\Omega$). $N_\Omega(x) = [\text{cone}(\Omega - x)]^o$ is the normal cone of $\Omega$ at $x \in \Omega$.

**Lemma 5** [33]: Let $\Omega$ be a closed convex set and let $x \in \Omega$. Then

$$N_\Omega(x) = \{x' \in \mathbb{R}^n : P_\Omega(x + x') = x\} \quad \text{(4)}$$

**Definition 1** [34] Generalized Gradient): A function $f : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuous in $X$, $v$ is a vector in $X$ and $\hat{f}(x) = \{\zeta \in \mathbb{R}^n : \zeta \in X : f^0(x; v) \geq \zeta, v\}$ denotes the generalized gradient of $f$ at $x$, where $f^0(x; v)$ is finite and well defined and $f^0(x; v) = \lim_{t \to 0} \text{sup}((f(x + tv) - f(x))/t)$. $\hat{f}_\mu(x)\mu$ is partial of $x$ denotes the subgradient of the partial derivatives, specifically defined as: $\hat{f}_\mu(x\mu) = \{\zeta_1 \in \mathbb{R}^n : f(\mu, y) - f(\mu, 0) \leq \zeta_1, \mu = 0\}$.

**Lemma 6** [34]: If $F : \mathbb{R}^n \to \mathbb{R}$ is $L$-Lipschitz continuous, then $\{x_i\} \to F(\xi_i)$ be a sequence in $\mathbb{R}^n$ and $\xi_i \in \partial F(x_i)$. Suppose that $x_i$ converges to $x$ and $\xi_i$ is a cluster point of $\xi_i$, and hence, $\xi \in \partial F(x)$.

**Lemma 7** (35) Invariance Principle): Let $\Omega$ be a compact set such that the system $\dot{x} = f(x)$ is unique and remains in $\Omega$ for all $t \geq t_0$. Let $V : \Omega \to \mathbb{R}$ be a time independent regular function such that $\dot{V} \leq 0$ for all $v \in \Omega$. Define $S = [x \in \Omega : 0 \in V]$. Then, every trajectory in $\Omega$ converges to the largest invariant set, $M$, in the closure of $S$.

**Remark 2**: The concepts related to nonsmooth analysis based on generalized gradient and differential inclusion are analyzed in [2] and [33]. The generalized gradient with respect to $x$ is often a set of vectors whose dimension is the same as $x$. When it comes to the inequality scaling, it is usual to choose a feasible element from the generalized gradient like formula (3a) in [2]. In this brief, for convenience and saving notation, when the inequalities involve the generalized gradient, the process of inequality scaling in the proof just continues without selecting some feasible elements from a generalized gradient.

III. MAIN RESULTS

In this section, we propose the concept of DPCO and give an RNN-based algorithm to solve DPCO problems. Compared with the distributed optimization, the cost function of DPCO has the same form as one in distributed optimization, but its optimal points have different dimensions and partial-consensus components. Specifically, compared with the distributed optimization, there are two main differences in the DPCO:

1) The dimensions of the optimal point for each objective function need not be identical.
2) Only some components of the optimal points for each objective function are identical.

To this end, this note mainly considers the DPCO problem modeled as follows:

$$\min F(x) = \sum_{i=1}^{N} f_i(x_i)$$

s.t. $x_i^{(n)} = x_j^{(n)}$

$$g_i(x_i) \leq 0,$$

$$x_i \in \Omega_i, \quad i, j = 1, \ldots, N \quad (5)$$

where $x_i \in \mathbb{R}^n$, $x = \text{col}\{x_1, \ldots, x_N\}$, $n \leq \min\{n_i\}$, $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$, and $g_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}, i = 1, \ldots, N$.

In this note, we call formula (5) as Problem (5). When we set special $n_i, m_i$, and $v_n$ such that $n_i = n_j, i, j = 1, \ldots, N, v_n = v$ and $m_j = m_j, i, j = 1, \ldots, N$, the DPCO problem becomes a distributed optimization problem. Thus, the DPCO problems are more
general than distributed optimization and decentralized optimization problems. In addition, due to differences, the methods, in existing works about distributed optimization, cannot solve the DPCO problem. Hence, this brief has the contribution on dealing with DPCO problems. In addition, due to differences, the methods, in existing problems. In addition, due to differences, the methods, in existing

\[ \begin{align*}
    \dot{x} &= 2\delta P_{\Omega} (x - \partial f (x) - \partial G (x)) P_{+}^\mu - K_n \lambda \\
    \dot{\lambda} &= K_n x \\
    \dot{\mu} &= \delta (P_{+}^\mu - \mu)
\end{align*} \]

where \( \delta = \delta_{\text{max}} (K_n) + 1 \).

Then, turning Algorithm (6) in compact form into a coupling form to describe the RNNs clearly

\[ \begin{align*}
    \frac{dx_i}{dt} &= 2\delta \left[ P_{\Omega} \left( x_i - \partial f_j (x_i) - \partial g_i (x_i) \right) P_{+}^\mu \right. \\
    & \quad - \left. \sum_{j=1,j \neq i}^{N} K_n (i,j) (x_i - x_j + \lambda_i - \lambda_j) - x_i \right] \\
    \frac{d\lambda_i}{dt} &= \sum_{j=1,j \neq i}^{N} K_n (i,j) (x_i - x_j) \\
    \frac{d\mu_i}{dt} &= \delta (P_{+}^\mu - \mu_i)
\end{align*} \]

where \( \delta = \delta_{\text{max}} (K_n) + 1 \).

The block diagram of the obtained multiple interconnected RNNs in Algorithm (6) is shown in Fig. 3, in which one can find that each RNN has a three-layer structure, and the RNNs, for example, \( i \) and \( j \), are linked by the constructed partial-consensus matrix \( K_n \). Each RNN exchanges the information on \( x_i \) and \( \lambda_i \) with its neighbors determined by \( K_n \). Specifically, in the collective RNNs, \( f_i, g_i \), and \( \Omega_i \) in (6) are only known by RNN \( i \), i.e., each RNN is arranged to minimize a local objective function subject to its local constraints rather than global information. Different from the applications in RNNs such as [5], [25], [26], (7) exchanges information with its neighbors to achieve partial consensus rather than global consensus.

Next, the construction process of Algorithm (6) is given in detail. Before further process, the definition of the equilibrium point of a differential equation (inclusion) is given.

**Definition 2:** [35] For a differential inclusion (equation), \( \dot{x} \in f (x) (\dot{x} = f (x)) \), \( x^* \) is said to be an equilibrium point of the differential inclusion (equation), if \( 0 \in f (x^*) (0 = f (x^*)) \).

Then, the main idea of the construction of RNNs is shown as follows.

**Step 1:** Derive Karush–Kuhn–Tucker (KKT) conditions from the considered problem.

**Step 2:** Construct the RNNs whose equilibrium point exactly satisfies KKT conditions (the construction methods, which can be found in [36], are various and always determined by the constraints and the cost functions).

**Step 3:** Prove that the designed RNNs converge to the equilibrium point.

In the remainder of Section III, three steps for Algorithm (6) are illustrated. Now, the necessary assumption is given, which is well known in convex optimization items to the convex inequality constraints.

**Assumption 1:**

1) **Convexity and Continuity:** \( \Omega_i \) is the convex set, \( f_i \) and \( g_i \) are convex and continuous, \( i = 1, \ldots, N \).

2) **Slater’s Condition:** \( \exists \bar{x} \in \text{int} (\Omega) \) such that \( g (\bar{x}) < 0 \).

Assumption 1 is a normal term in optimization problem. Convexity and continuity keep that the optimal points are the global solutions. Slater’s condition guarantees the constraint qualification for optimization problems. In addition, Problem (5) should have at least one optimal solution.

For step 1, it is necessary to propose a lemma under Assumption 1, which is the KKT conditions for Problem (5).

**Lemma 8:** Under Assumption 1, a point \( x^* \) is the optimal solution to Problem (5) if and only if there exist \( \lambda^* \) and \( \mu^* \) such that

\[ \begin{align*}
    0 &\in \partial F (x^*) + \partial G (x^*) \mu^* + K_n \lambda^* + N_{\Omega} (x^*) \\
    g_i (x_i^*) &\in N_{R_1^m} (\mu_i^*) \quad \forall i = 1, \ldots, N. \quad (8)
\end{align*} \]

**Proof:** According to Lemma 2, we can verify that Problem (5) is equivalent to

\[ \begin{align*}
    \min_{x} F (x) &= \sum_{i=1}^{N} f_i (x_i) \\
    \text{s.t.} \quad K_n x &= 0, \ g_i (x_i) \leq 0, \ x_i \in \Omega_i, \quad i = 1, \ldots, N. \quad (9)
\end{align*} \]

Under Assumption 1, there exists an optimal solution \( x^* \), \( \lambda^* = \text{col} [\lambda_1^*, \ldots, \lambda_N^*] \) with \( \lambda_i^* \in R_1^m \), and \( \mu^* = \text{col} [\mu_1^*, \ldots, \mu_N^*] \) with \( \mu_i^* \in R_1^m \) such that

\[ \begin{align*}
    0 &\in \partial F (x^*) + \partial G (x^*) \mu^* + K_n \lambda^* + N_{\Omega} (x^*) \quad (10) \\
    K_n x^* &= 0 \quad (11) \\
    \mu_i^* &\geq 0, \ g_i (x_i) \leq 0, \ (\mu_i^*, g_i (x_i)) = 0 \quad \forall i = 1, \ldots, N. \quad (12)
\end{align*} \]

It is no doubt that (10) amounts to the first formula of (8), whereas (12) amounts to the second formula of (8), which keeps this proof. ■

For step 2, we need to prove the equivalence between the equilibrium point of Algorithm (6) and the optimal solution to Problem (5).

**Theorem 1:** Under Assumption 1, the equilibrium point of Algorithm (6) is an optimal solution to Problem (5).
Proof: Let \((\hat{x}, \hat{\lambda}, \hat{\mu})\) denote the equilibrium point of Algorithm (6), which indicates

\[
0 \in 2\delta P_{\Omega} \left( \hat{x} - \partial F(\hat{x}) - \partial G(\hat{x}) P_{\Omega}^\mu K_{\hat{\lambda}} - K_{\hat{\lambda}} \hat{x} \right) - 2\delta \hat{x}
\]

(13a)

\[
0 = K_{\hat{\lambda}} \hat{x}
\]

(13b)

\[
0 = \delta (P_{\Omega}^\mu - \hat{\mu})
\]

(13c)

Thus, (13c) implies that \(P_{\Omega}^\mu = \hat{\mu}\), which guarantees that \(P_{\Omega}^\mu(\hat{\mu}_0 + g_t(x_t)) = \hat{\mu}_0\). Then, the second formula of (8) holds. To sum up with \(P_{\Omega}^\mu = \hat{\mu}\), (13a) and (13b), the first formula of (8) holds. On account of Lemma 8 and (8), \(\hat{x}\) is an optimal solution to Problem (5).

Lemma 8 and Theorem 1 connect the equilibrium point of Algorithm (6) with the optimal solution to Problem (5) by the tools in KKT condition and equilibrium analysis.

For step 3, the convergence of Algorithm (6) is proved in the next theorem. The convergence of the algorithm can be proved by the approaches of nonsmooth analysis and Lyapunov-based technique. The proof consists of two parts: 1) the boundedness of \(x, \lambda, \mu\) from Algorithm (6) and 2) Algorithm (6) converge to the equilibrium point of Algorithm (6). The proving method is to construct Lyapunov functions and use the properties of coercive functions.

Theorem 2: Under Assumption 1, then the following statements are correct.

1) \(x, \lambda, \mu\) from Algorithm (6) are bounded.
2) The states \(x\) of Algorithm (6) will converge to an optimal solution for Problem (5) from any initial point \((x(0), \lambda(0), \mu(0))\).

Proof: Similar to the proof of Theorem 1, \((\hat{x}, \hat{\lambda}, \hat{\mu})\) denotes the equilibrium of Algorithm (6). The proof begins with constructing the Lyapunov functions

\[
\Phi = \Phi_2 + \Phi_3 + \Phi_4
\]

\[
\Phi_1 = F(x) + \frac{1}{2} \| P_{\Omega}^\mu \|^2 + \frac{1}{2} \| \lambda + \hat{\lambda} \|^2 K_n(x + \hat{x})
\]

\[
\Phi_2 = \Phi_2(1, \lambda, \mu) - \Phi(\hat{x}, \hat{\lambda}, \hat{\mu}) T(x - \hat{x})
\]

\[
\Phi_3 = \frac{1}{2} \| x - \hat{x} \|^2 + \frac{1}{2} \| \lambda - \hat{\lambda} \|^2 (2M_N - K_{\hat{\lambda}})(\lambda - \hat{\lambda})
\]

\[
\Phi_4 = \frac{1}{2} \| \mu - \hat{\mu} \|^2.
\]

(14)

The Lyapunov function \(\Phi\) should satisfy three conditions.

1) \(\Phi > 0\) when \(\col(x, \lambda, \mu) \neq \col(\hat{x}, \hat{\lambda}, \hat{\mu})\).
2) \(\Phi \leq 0\).
3) \(\Phi \to \infty\) whenever \(\|\col(x, \lambda, \mu)\| \to \infty\).

First, we prove that \(\Phi\) satisfies 1). From the convexity of \(F(x)\) and norm operator \(\|\cdot\|\), we have that \(\Phi_1\) is also convex. For convex function \(\Phi_1(\lambda, \mu, \mu)\), the inequality

\[
\Phi_1(\lambda, \mu) - \Phi_1(\hat{x}, \hat{\lambda}, \hat{\mu}) \geq (x - \hat{x})^T \partial \Phi_1(\hat{x}, \hat{\lambda}, \hat{\mu})
\]

\[
+ (\lambda - \hat{\lambda})^T \partial \Phi_1(\hat{x}, \hat{\lambda}, \hat{\mu}) + (\mu - \hat{\mu})^T \partial \mu(\hat{x}, \hat{\lambda}, \hat{\mu})
\]

\[
= (x - \hat{x})^T \partial \Phi_1(\hat{x}, \hat{\lambda}, \hat{\mu}) + (\lambda - \hat{\lambda})^T K_n(\hat{x} + \hat{\lambda})
\]

\[
- (\mu - \hat{\mu})^T \hat{\mu}
\]

(15)

holds. From (15), \(\Phi_2 \geq 0\).

For \(\Phi_3\), \(\delta I_N - K_{\hat{\lambda}}\) is positive definite due to \(\delta = \delta_{\text{max}}(K_n) + 1\), and \(\|x - \hat{x}\|^2 \geq 0\). Thus, \(\Phi_3 \geq 0\). In addition, \(\Phi_4 \geq 0\) due to

\[\|\mu - \hat{\mu}\|^2 \geq 0\]. \(\Phi\) is an appropriate Lyapunov function satisfies that \(\Phi = \Phi_2 + \Phi_3 + \Phi_4 \geq 0\) and \(\Phi = 0\) only if \(\col(x, \lambda, \mu) = \col(\hat{x}, \hat{\lambda}, \hat{\mu})\).

Then, we prove \(\Phi\) satisfying 2), and hence, the derivative of \(\Phi = \Phi_2 + \Phi_3 + \Phi_4\) needs to be given. The derivative of \(\Phi_2\) is calculated

\[
\dot{\Phi}_2 = \partial \Phi_2 x + \partial \Phi_2 \hat{x} + \partial \Phi_2 \hat{\mu} \]

\[
= \partial \Phi_1(x, \lambda, \mu) + \partial \mu(x, \lambda, \mu)
\]

\[
= \frac{1}{2} \| x - \hat{x} \|^2 + \frac{1}{2} \| \lambda - \hat{\lambda} \|^2 (2M_N - K_{\hat{\lambda}})(\lambda - \hat{\lambda})
\]

(16)

Divide \(\dot{\Phi}_2\) into three parts \(\dot{\Phi}_{21}, \dot{\Phi}_{22}, \) and \(\dot{\Phi}_{23}\)

\[
\dot{\Phi}_{21} = \left( \partial \Phi_1(x, \lambda, \mu) - \partial \Phi_1(\hat{x}, \hat{\lambda}, \hat{\mu}) \right) x
\]

\[
\dot{\Phi}_{22} = (x - \hat{x})^T K_{\hat{\lambda}} \hat{\lambda} + (\lambda - \hat{\lambda})^T K_{\hat{\lambda}} \hat{\lambda}
\]

\[
\dot{\Phi}_{23} = (\mu - \hat{\mu})^2 \hat{\mu}
\]

(17)

(18)

(19)

On the purpose of convenience, set \(P_{\Omega} = P_{\Omega}(x - \hat{F}(x) - \partial G(x) P_{\Omega}^\mu - K_{\hat{\lambda}} \hat{\lambda} - K_n \hat{x})\). At this moment, one can get

\[
\dot{\Phi}_3 + \dot{\Phi}_{21} = 2\delta (x - \hat{x})^T (P_{\Omega} - \hat{x}) + 2\delta \left( \partial \Phi_1(x, \lambda, \mu) \right) (P_{\Omega} - \hat{x})
\]

\[= 2\delta (x - \hat{x})^T (P_{\Omega} - \hat{x}) + 2\delta (x - \hat{x})^T (P_{\Omega} - \hat{x})
\]

\[= 2\delta (x - \hat{x})^T (P_{\Omega} - \hat{x}) + (\lambda - \hat{\lambda})^T (2\delta I_N - K_{\hat{\lambda}}) \hat{\lambda}
\]

\[= 2\delta (x - \hat{x})^T (P_{\Omega} - \hat{x}) + (\lambda - \hat{\lambda})^T (2\delta I_N - K_{\hat{\lambda}}) \hat{\lambda}
\]

\[= 2\delta (x - \hat{x})^T (P_{\Omega} - \hat{x}) + (\lambda - \hat{\lambda})^T (2\delta I_N - K_{\hat{\lambda}}) \hat{\lambda}
\]

(20)

Then, using (3), (4), and (8), one can find that \(\dot{\Phi}_1(\Phi - P_{\Omega} + x)(P_{\Omega} - \hat{x}) \leq 0\) and \(\dot{\Phi}_1(\Phi - P_{\Omega} - \hat{x}) \leq 0\). To proceed, calculating

\[
\dot{\Phi}_3 + \dot{\Phi}_{21} \leq -2\delta (P_{\Omega} - \hat{x})^T (P_{\Omega} - \hat{x}) + 2\delta (x - \hat{x})^T (P_{\Omega} - \hat{x})
\]

\[= 2\delta (x - \hat{x})^T (P_{\Omega} - \hat{x}) + (\lambda - \hat{\lambda})^T (2\delta I_N - K_{\hat{\lambda}}) \hat{\lambda}
\]

\[\leq -2\delta ||P_{\Omega} - \hat{x}||^2 + 2\delta \left( \partial \Phi_1(x, \lambda, \mu) \right) (x - \hat{x})
\]

\[= -2\delta ||P_{\Omega} - \hat{x}||^2 + 2\delta \left( \partial \Phi_1(x, \lambda, \mu) \right) (x - \hat{x})
\]

(21)

(22)

(23)
Furthermore, using the invariance principle (Lemma 7), we can find it is easy to verify that if $F_i = F_{i-1} + \sigma_i\mathcal{E}_i$, then we provide the method to handle the last inequality converges to the largest invariant set in the closure of $S := \{x \in \Phi(0) : \dot{\Phi} = 0\}$. To analyze set $S$, one can obtain from (26) that $\Phi = 0$ results in that the point $(x, \lambda, \mu)$ is actually an equilibrium of (5). Hence, (2) holds.

IV. COMPARISON AND SIMULATION RESULTS

In this section, we list Table I to compare the existing works in references with this brief. Then, consider the following optimization problem.

Example 2:

\[
\min_{x_1, x_2} \sum_{i=1}^{3} f_i(x_i) \\
\text{s.t. } x_i^{(0)} = x_j^{(0)}, \quad g_i(x_i) \\n\end{align*}
\]

where $x_1 \in \mathbb{R}$, $x_2 = \text{col}[x_{31}, x_{32}] \in \mathbb{R}^2$, and $x_3 = \text{col}[x_{13}, x_{23}] \in \mathbb{R}^2$. $\Omega_{1, 2, 3} = [1, 2]$, $x = \text{col}[x_1, x_2, x_3]$, $v_1 = \{1\}$ indicates that $x_1 = x_{21} = x_{31}$.

\[
f_i(x_i) = \begin{cases} 
(x_i - 1.5)^2 + |x_i - 0.5| & i = 1 \\
|x_i - 1| + (x_i - 1.5)^2 & i = 2 \\
|x_i - 1| + |x_i - 1.5| & i = 3 
\end{cases}
\]

\[
g_i(x_i) = \begin{cases} 
x_i - 2 & i = 1 \\
e^{x_i^2} - 5 & i = 2 \\
x_i - x_{i-2} - 0.4 & i = 3 
\end{cases}
\]

We select the Laplacian matrix

\[
L_3 = \begin{bmatrix} 
1 & -1 & 0 \\
-1 & 2 & -1 \\
0 & -1 & 1 
\end{bmatrix}.
\]

Then, $K_1$ can be constructed as follows:

\[
K_1 = \begin{bmatrix} 
1 & -1 & 0 & 0 & 0 \\
-1 & 2 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 
\end{bmatrix}.
\]

The graph of $L_3$ and $K_1$’s adjacency matrix is shown in Fig. 4.

The optimal solution of the problem in Example 2 by Algorithm (6) is shown in Figs. 5 and 6. Fig. 5 shows the trajectories of the components of $x$, and Fig. 6 shows the value of the objective function from different initial states. We can get the optimal solution $x_1 = 1.0536$, $x_2 = [1.0536, 1.5858]^T$, and $x_3 = [1.0536, 1.5]^T$, which can be verified that this point is optimal. The optimal value is 1.4532 from Fig. 6. From Fig. 5, Algorithm (6) makes some components of $x$ identical correspondingly, but the simulation results in relative briefs with distributed optimization can only achieve all components of $x$ identical correspondingly. Besides, we can find that DPCO problem has a more general form than (global) distributed optimization problem. For Problem (5), when we set special $n_1, m_1$ and $v_n$ such that $n_j = n_1, i, j = 1, \ldots, N, v_n = v$, and
In this brief, a continuous-time decentralized algorithm in RNN-based fashion has been proposed for the DPCO under the convex inequality constraints and local set constraints. A novel matrix, called partial-consensus matrix, was presented to deal with the partial-consensus constraints. By the means of nonsmooth analysis and Lyapunov-based technique, the convergence of the continuous-time decentralized algorithm has been proved. Finally, a numerical simulation has also been illustrated to show the algorithm’s performance. In future works, the combination with the matrix graph theory will be considered in more flexible cases for DPCO problems.

### TABLE I

| Ref. | Type of cost function | Domain | Type of constraints | Methods |
|------|-----------------------|--------|---------------------|---------|
| [20] | Centralized, non-smooth | Real   | Linear equalities, inequalities | RNNs    |
| [23] | Centralized, non-smooth | Real   | Linear equalities, bound | RNNs    |
| [31] | Centralized, non-smooth | Quaternion | Linear equalities, inequalities | RNNs    |
| [32] | Centralized, quadratic | Real   | Linear equalities, bound | RNNs    |
| [6]  | Distributed, smooth    | Real   | None                | Multi-agent systems |
| [38] | Distributed, smooth    | Real   | None                | Opinion dynamics |
| [39] | Distributed, smooth    | Real   | Local bound         | Feedback feedforward control |
| [28] | Distributed, non-smooth | Real   | Linear equalities, inequalities | Multi-agent systems |
| [29] | Distributed, non-smooth | Real   | Linear equalities, inequalities | Multi-agent systems |
| [1]  | Distributed, non-smooth | Real   | Local bound         | Penalty, adaptive RNNs |
| [32] | Distributed, non-smooth | Real   | Approximate equalities, inequalities | RNNs    |
| Herein | Decentralized-partial-consensus, non-smooth | Real | Local bound, inequalities | RNNs    |

### References

[1] H. Zhou, X. Zeng, and Y. Hong, “Adaptive exact penalty design for constrained distributed optimization,” IEEE Trans. Autom. Control, vol. 64, no. 11, pp. 4661–4667, Nov. 2019.

[2] X. Zeng, P. Yi, and Y. Hong, “Distributed continuous-time algorithm for constrained convex optimizations via nonsmooth analysis approach,” IEEE Trans. Autom. Control, vol. 62, no. 10, pp. 5227–5233, Oct. 2017.

[3] X. Wang, S. Li, and G. Wang, “Distributed optimization for disturbed second-order multiagent systems based on active antistability control,” IEEE Trans. Neural Netw. Learn. Syst., vol. 31, no. 6, pp. 2104–2117, Jun. 2020.

[4] D. Wang, Z. Wang, M. Chen, and W. Wang, “Distributed optimization for multi-agent systems with constraints set and communication time-delay over a directed graph,” Inf. Sci., vol. 438, pp. 1–14, Apr. 2018.

[5] Z. Xia, Y. Liu, J. Lu, J. Cao, and L. Rutkowski, “Penalty method for constrained distributed quaternion-variable optimization,” IEEE Trans. Cybern., early access, Nov. 18, 2020, doi: 10.1109/TCYB.2020.3031687.

[6] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” IEEE Trans. Autom. Control, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[7] I. Lobel and A. Ozdaglar, “Distributed subgradient methods for convex optimization over random networks,” IEEE Trans. Autom. Control, vol. 56, no. 6, pp. 1291–1306, Jun. 2011.

[8] Z. Deng, S. Liang, and Y. Hong, “Distributed continuous-time algorithms for resource allocation problems over weight-balanced digraphs,” IEEE Trans. Cybern., vol. 48, no. 11, pp. 3116–3125, Nov. 2018.

[9] T. Yang et al., “A survey of distributed optimization,” Annu. Rev. Control, vol. 47, pp. 278–305, Jan. 2019.

[10] D. Helbing et al., “Saving human lives: What complexity science and information systems can contribute,” J. Stat. Phys., vol. 158, no. 3, pp. 735–781, 2015.

[11] X. Wang, G. Wang, and S. Li, “Distributed finite-time optimization for integrator chain multiagent systems with disturbances,” IEEE Trans. Autom. Control, vol. 65, no. 12, pp. 5296–5311, Dec. 2020.

[12] S. Liang, X. Zeng, and Y. Hong, “Distributed sub-optimal resource allocation over weight-balanced graph via singular perturbation,” Automatica, vol. 95, pp. 222–228, Sep. 2018.

[13] X. Wang, G. Wang, and S. Li, “Distributed finite-time optimization for disturbed second-order multiagent systems,” IEEE Trans. Cybern., early access, May 21, 2020, doi: 10.1109/TCYB.2020.2998490.

[14] H. Jaleel and J. S. Shamma, “Distributed optimization for robot networks: From real-time convex optimization to game-theoretic self-organization,” Proc. IEEE, vol. 108, no. 11, pp. 1953–1967, Nov. 2020.
[15] X. Wang, Y. Hong, P. Yi, H. Ji, and Y. Kang, “Distributed optimization design of continuous-time multiagent systems with unknown-frequency disturbances,” *IEEE Trans. Cybern.*, vol. 47, no. 8, pp. 2058–2066, Aug. 2017.

[16] S. Yang, J. Wang, and Q. Liu, “Cooperative–competitive multiagent systems for distributed minimax optimization subject to bounded constraints,” *IEEE Trans. Autom. Control*, vol. 64, no. 4, pp. 1358–1372, Apr. 2019.

[17] D. Tank and J. Hopfield, “Simple ‘neural’ optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit,” *IEEE Trans. Circuits Syst.*, vol. CAS-33, no. 5, pp. 533–541, May 1986.

[18] T. Ergen and S. S. Kozat, “Efficient online learning algorithms based on LSTM neural networks,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp. 3772–3783, Aug. 2018.

[19] B.-J. Hou and Z.-H. Zhou, “Learning with interpretable structure from gated RNN,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2267–2279, Jul. 2020.

[20] Q. Liu and J. Wang, “A one-layer recurrent neural network for constrained nonsmooth optimization,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 5, pp. 1323–1333, Oct. 2011.

[21] Y. Xia and J. Wang, “A general projection neural network for solving monotone variational inequalities and related optimization problems,” *IEEE Trans. Neural Netw.*, vol. 15, no. 2, pp. 318–328, Mar. 2004.

[22] Q. Liu and J. Wang, “A one-layer recurrent neural network with a discontinuous hard-limiting activation function for quadratic programming,” *IEEE Trans. Neural Netw.*, vol. 19, no. 4, pp. 558–570, Apr. 2008.

[23] Q. Liu and J. Wang, “A one-layer projection neural network for nonsmooth optimization subject to linear equalities and bound constraints,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 5, pp. 812–824, May 2013.

[24] Q. Liu, T. Huang, and J. Wang, “One-layer continuous-and discrete-time projection neural networks for solving variational inequalities and related optimization problems,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 7, pp. 1308–1318, Jul. 2014.

[25] Q. Liu, S. Yang, and J. Wang, “A collective neurodynamic approach to distributed constrained optimization,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 8, pp. 1747–1758, Aug. 2017.

[26] J. Fan and J. Wang, “A collective neurodynamic optimization approach to nonnegative matrix factorization,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2344–2356, Oct. 2017.

[27] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[28] S. Liang, X. Zeng, and Y. Hong, “Distributed nonsmooth optimization with coupled inequality constraints via modified Lagrangian function,” *IEEE Trans. Autom. Control*, vol. 63, no. 6, pp. 1753–1759, Jun. 2018.

[29] X. Li, L. Xie, and Y. Hong, “Distributed continuous-time nonsmooth convex optimization with coupled inequality constraints,” *IEEE Trans. Control Netw. Syst.*, vol. 7, no. 1, pp. 74–84, Mar. 2020.

[30] M. Zhu and S. Martinez, “On distributed convex optimization under inequality and equality constraints,” *IEEE Trans. Autom. Control*, vol. 57, no. 1, pp. 151–164, Jan. 2012.

[31] Y.-W. Lv, G.-H. Yang, and C.-X. Shi, “Differentially private distributed optimization for multi-agent systems via the augmented Lagrangian algorithm,” *Inf. Sci.*, vol. 538, pp. 39–53, Oct. 2020.

[32] X. Jiang, S. Qin, and X. Xue, “Continuous-time algorithm for approximate distributed optimization with affine equality and convex inequality constraints,” *IEEE Trans. Syst., Man, Cybern. Syst.*, early access, Dec. 19, 2020, doi: 10.1109/TSMC.2019.2957156.

[33] A. P. Ruszczynski, *Nonlinear Optimization*. Princeton, NJ, USA: Princeton Univ. Press, 2006.

[34] F. Clarke, *Optimization and Nonsmooth Analysis*. Hoboken, NJ, USA: Wiley, 1983.

[35] W. Ren and Y. Cao, *Distributed Coordination of Multi-Agent Networks: Emergent Problems, Models, and Issues*. London, U.K.: Springer-Verlag, 2010.

[36] Y. Xia and J. Wang, “A general methodology for designing globally convergent optimization neural networks,” *IEEE Trans. Neural Netw.*, vol. 9, no. 6, pp. 1331–1343, Nov. 1998.

[37] Y. Liu, Y. Zheng, J. Lu, J. Cao, and L. Rutkowski, “Constrained quaternion-variable convex optimization: A quaternion-valued recurrent neural network approach,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 19, no. 2, pp. 418–427, Feb. 2018.

[38] X. L. Shi, J. Cao, G. Wen, and M. Perc, “Finite-time consensus of opinion dynamics and its applications to distributed optimization over digraph,” *IEEE Trans. Cybern.*, vol. 49, no. 10, pp. 3767–3779, Oct. 2019.

[39] Z. Qiu, L. Xie, and K. You, “Feedback-feedforward control approach to distributed optimization,” in *Proc. Amer. Control Conf. (ACC)*, Jul. 2019, pp. 1412–1417.

[40] Z. Deng, S. Liang, and W. Yu, “Distributed optimal resource allocation of second-order multiagent systems,” *Int. J. Robust Nonlinear Control*, vol. 28, no. 14, pp. 4246–4260, Sep. 2018.