# Convex and Nonconvex Optimization Based on Neurodynamic Method with Zero-Sum Initial Constraint

Yiyang Ge, Zhanshan Wang, and Bibo Zheng

***Abstract*—A neurodynamic method (NdM) for convex optimization is proposed in this paper with an equality constraint. The method utilizes a neurodynamic system (NdS) that converges to the optimal solution of a convex optimization problem in a fixed time. Due to its mathematical simplicity, it can also be combined with reinforcement learning (RL) to solve a class of nonconvex optimization problems. To maintain the mathematical simplicity of NdS, zero-sum initial constraints are introduced to reduce the number of auxiliary multipliers. First, the initial sum of the state variables must satisfy the equality constraint. Second, the sum of their derivatives is designed to remain zero. In order to apply the proposed convex optimization algorithm to nonconvex optimization with mixed constraints, the virtual actions in RL are redefined to avoid the use of NdS inequality constrained multipliers. The proposed NdM plays an effective search tool in constrained nonconvex optimization algorithms. Numerical examples demonstrate the effectiveness of the proposed algorithm.**

***Index Terms*—Neurodynamic method (NdM), zero-sum initial constraint, distribued optimization, convex and nonconvex optimization, reinforcement learning (RL)**

## I. INTRODUCTION

Convex and nonconvex optimization has been widely applied in various fields, such as economic dispatch (ED), signal processing, resource allocation, and energy management [1–4]. Among the optimization algorithms, the neurodynamic method (NdM) based on the ordinary differential equation (ODE) theory has demonstrated unique advantages. It not only has better generalization performance and robustness, but also can control dynamic characteristics during the optimal solution search. This method has different names [5–7], which are unified here as the NdM. The essence of this method is to use ordinary differential equations to correspond to optimal solutions. Based on the simplicity and completeness of the ODE theory, NdM exhibits flexibility and adaptability in handling complex engineering problems.

Combined with distributed methods, NdM utilizes parallel computing to handle optimization problems faster. Therefore, some scholars have designed distributed neurodynamic system (NdS) that converges to the optimal solution of convex optimization problems [8–11]. In these systems, the convergence efficiency is an important factor to measure the optimization performance. The general result of distributed NdS is asymptotic convergence to the optimal solution in current research. In addition, NdM for solving constrained optimization problems has been proposed [10–13]. While the convergence time of these algorithms is only controlled by adjusting the parameters, the effect is limited. In the industrial, achieving the optimal solution within a finite-time is crucial for the reliability and predictability of algorithms. To this end, some scholars have investigated NdS with finite-time stability for optimization [14–17]. However, the settling time $T$ for determining finite-time stability is dependent on the initial state and system parameters. To improve the convergence and avoid the dependence of the settling time on the initial value, the fixed-time stability is proposed. Some results for optimization are given in Refs. [18–20].

There are designed fixed-time stable systems that perform well in handling unconstrained optimization problems [21, 22]. However, it is difficult to obtain a fast system for situations with constraints. There are two reasons. On the one hand, the designed system inevitably contains exponential terms with optimal conditions, such as extremum conditions, complementary relaxation conditions, constraint conditions, etc. They play a role in proving the fixed-time stability of the system. But as constraints increase, it means that more exponential terms and interaction terms are added to the system, which increases the computational burden. On the other hand, it may be necessary to introduce additional variables to handle multiple constraints. These terms, as discussed in Refs. [19, 20], are designed to accelerate the convergence process. However, auxiliary multipliers interact exponentially in communication to satisfy the distribution characteristics, which also imposes computational and communication burdens. The design of systems with fixed-time stability is more complex. It is not suitable for real-time, large-scale, and iterative optimization.

In order to improve search efficiency, a compact NdS is proposed, which asymptotically converges to the optimal solution [23]. The derivative of the state variable is designed to satisfy the equality constraint. A similar approach in Refs.

Manuscript received: 25 July 2024; revised: 20 September 2024; accepted: 23 October 2024. *(Corresponding author: Zhanshan Wang.)*

Citation: Y. Ge, Z. Wang, and B. Zheng, Convex and nonconvex optimization based on neurodynamic method with zero-sum initial constraint, *IJICS*, 2024, 29(4), 184–194.

Yiyang Ge, Zhanshan Wang, and Bibo Zheng are with College of Information Science and Engineering, Northeastern University, Shenyang 110819, China (e-mail: gyyneu2022@163.com; zhanshan_wang@163.com; zbbenergy@163.com).

Digital Object Identifier 10.62678/IJICS202412.10143

[20, 24] introduces initial values in the system. Some scholars have also proposed a simplified system form based on the Newton's unconstrained optimization method [21, 25–27], but it requires second-order information. Gradient explosion may lead to algorithm failure as complexity and dimensionality increase. Alternatively, an NdS with finite/predefined-time consensus is proposed for constraint optimization [15, 28]. The system only reaches consensus on state variables within a finite/predefined-time, and then asymptotically reaches the optimal solution.

Recently, the coordinated Q-learning (CQL) method in Ref. [29] has gained widely attention. It decomposes complex nonconvex optimization problems into a large number of simple convex optimization problems within the reinforcement learning (RL) framework. The reward value of Q-function is accumulated by the convex optimization algorithm. In this way, it not only alleviates the overall communication burden in nonconvex optimization algorithms, but also reduces the memory pressure associated with maintaining Q-value tables. Inspired by this, a fast response NdS with a few variables is proposed to solve convex optimization problems, which can improve the overall computational efficiency of nonconvex optimization algorithms.

To provide an effective search tool for RL, a single variable NdS is constructed to handle constrained optimization problems. The contributions are as follows:

(1) The proposed single variable NdS converges within a fixed time. Compared with recent fixed-time distributed optimization [21, 25, 30], our algorithm introduces equality constraints. Furthermore, in contrast to methods proposed in Refs. [18, 19], our proposed NdM with zero-sum initial constraints reduces the number of auxiliary multipliers.

(2) The algorithm achieves the optimal solution within a fixed time instead of asymptotically converging to the optimal solution after reaching consensus within a fixed time. This is different from Refs. [28, 31].

(3) By integrating inequality constraints into the box constraints of nonconvex ED, the feasible domain of the action space within the RL framework is redefined. This reduces the inequality constraint auxiliary multiplier in convex optimization algorithms. Therefore, the proposed convex constraint optimization algorithm combined with CQL can effectively solve a class of nonconvex problems.

## II. Preliminary

### A. Algebraic Graph Theory

Consider a multi-agent system consisting of $n$ agents with an undirected communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$. The set of nodes is represented by $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$, and each node corresponds to an agent. The symbol $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ denotes the set of edges. The adjacency matrix is represented by $\mathcal{A} = [a_{ij}]_{n\times n}$. Note that $a_{ij} = a_{ji} > 0$, if and only if $(v_i, v_j) \in \mathcal{E}$. An undirected graph is called connected, for any two nodes $v_i$ and $v_j$, if there exists a path $(v_i, v_{k_1}, \ldots, v_{k_m}, v_j)$ that connects them. The degree of graph $\mathcal{G}$ is $d_i = \sum_{j=1}^{n} a_{ij}$. $L = [l_{ij}]_{n\times n}$ is the Laplace matrix, where $l_{ij} = -a_{ij}$ and $l_{ii} = d_i$. The $L_2$-norm of a vector $x(t) \in \mathbb{R}^n$ is denoted by $\|x(t)\|$. $\mathrm{sig}^{\alpha}(x(t)) = \mathrm{col}\{\mathrm{sig}^{\alpha}(x_1(t)), \mathrm{sig}^{\alpha}(x_2(t)), ..., \mathrm{sig}^{\alpha}(x_n(t))\}$, $i = 1, 2, ..., n$, in which $\mathrm{sig}^{\alpha}(\cdot) = \mathrm{sign}(\cdot)|\cdot|^{\alpha}$. The symbol $P_{\Omega}(y)$ represents the projection of point $y$ on set $\Omega$.

### B. Fixed-Time Stability

Consider the nonlinear system

$$\dot{x}(t) = g(x(t)),\ x(0) = x_0 \tag{1}$$

**Definition 1 [32]** The equilibrium of Eq. (1) is finite-time stable, if there exists the settle time $T(x_0)$, such that the Eq. (1) converges to equilibrium point in finite-time.

**Definition 2 [33]** The equilibrium of Eq. (1) is said to be fixed-time stable if it is finite-time stable and there exists a constant $T_c$ such that the settling time $T(x_0)$ satisfies $T(x_0) < T_c$ for any initial state $x_0$, ensuring that Eq. (1) converges to the equilibrium point within a fixed time $T_c$.

**Lemma 1 [34]** The C-regular function $V(x(t)): \mathbb{R}^n \to \mathbb{R}$, satisfies the conditions involving positive constants $a$ and $b$, as well as non-negative constants $\alpha > 1$ and $0 < \beta < 1$, such that

$$\dot{V}(x(t)) = -aV(x(t))^{\alpha} - bV(x(t))^{\beta},\ x(t) \in \mathbb{R}^n \backslash \{0\}.$$

The origin of Eq. (1) is fixed-time stable within time $T$, in which $T(x(t)) \leqslant T_c \triangleq (\pi/(\alpha-\beta)^b)(b/a)^{\epsilon}\csc(\epsilon\pi)$, $\epsilon = (1-\beta)/(\alpha-\beta)$.

### C. Useful Lemma

**Lemma 2 [35]** Assuming $p_i > 0$, for $0 \leqslant \beta \leqslant 1$ and $\alpha > 1$, the following conclusion can be drawn

$$\sum_{n=1}^{N} p_i^{\beta} \geqslant (\sum_{n=1}^{N} p_i)^{\beta},$$
$$\sum_{n=1}^{N} p_i^{\alpha} \geqslant N^{1-\alpha}(\sum_{n=1}^{N} p_i)^{\alpha}.$$

**Lemma 3 [36]** For undirected connected graph $\mathcal{G}$, the eigenvalues of Laplace matrix $L$ are nonnegative real numbers and can be arranged in order as $0 = \lambda_1(L) \leqslant \lambda_2(L) \leqslant \cdots \leqslant \lambda_n(L)$. The second smallest eigenvalue satisfies $\lambda_2(L) = \min_{x \neq 0_N, 1_N^{\mathrm{T}} x = 0} \left(x^{\mathrm{T}} L x\right) / \left(x^{\mathrm{T}} x\right)$.

**Lemma 4 [27]** The function $f: \mathbb{R}^n \to \mathbb{R}$ is $m$-strongly convex, if exists $m > 0$, such that $f(y) - f(x) \geqslant \nabla f(x)^{\mathrm{T}}(y-x) + m\|y-x\|^2/2$, $\forall x, y \in \mathbb{R}^n$.

**Lemma 5 [37]** Let $\Omega$ be a nonempty closed convex set. The squared distance function from a point $x$ to $\Omega$ is defined as $\Theta_{\Omega}(x) \equiv \|x - P_{\Omega}(x)\|^2/2$, which is continuously differentiable in $x$. Furthermore, its gradient $\nabla\Theta_{\Omega}(x)$ is given by $x - P_{\Omega}(x)$.

*D. Assumption*

**Assumption 1** The graph $\mathcal{G}$ is connected and undirected.

**Assumption 2** There exists $x_i \in \mathbb{R}$, such that $\sum_{i=1}^{N} x_i = d$.

**Assumption 3** There exists $x_i \in \Omega_i$, where $\Omega_i = \{x_i | \underline{x}_i \leqslant x_i \leqslant \overline{x}_i, |x_{i,t} - x_{\text{state},i}| \leqslant x_i^{\text{R}}\}$, such that $\sum_{i=1}^{N} x_i = d$.

**Assumption 4** The local function $f(x_i)$ is $m$-strongly convex and differentiable.

Assumptions 1, 2, and 4 are prepared for convex optimization, while Assumptions 1 and 3 are prepared for nonconvex optimization.

**Remark 1** Assumption 1 is a relatively common assumption. It is suitable for multi-agent communication networks [10, 15, 18]. Under this assumption, the Laplacian matrix satisfies $1_N^{\text{T}} L = 0$, which is important for subsequent analysis. Assumptions 2 and 3 are Slater conditions. Specifically, Assumption 2 is utilized for optimization problems involving equality constraints. For ED, ramp rate and box constraints are introduced, necessitating the use of Assumption 3. Assumption 4 relates to the convex optimization problem proposed in this paper. In nonconvex optimization, the form of the ED objective function is predefined, and no additional requirements are imposed.

## III. Distributed Optimization Model

In this section, an NdS is designed to approximate the optimal solution for convex optimization problems with equality constraints. Furthermore, application in RL framework to realize nonconvex ED is also discussed.

*A. Fixed-Time Distributed Optimization with An Equality Constraint*

An optimization problem with equality constraints is considered

$$\begin{aligned} \min_{x} \quad & f(x(t)) = \sum_{i=1}^{N} f_i(x_i(t)), \\ \text{s.t.,} \quad & \sum_{i=1}^{N} x_i(t) = d \end{aligned} \tag{2}$$

The above optimization problem is common in resource allocation and ED. The purpose of equality constraints is usually to maintain supply-demand balance. Neurodynamic methods are efficient in complex optimization problems with constraints. Initially, Hopfield and Tank [38] used Hopfield neural networks to solve the traveling salesman problem (TSP), demonstrating its potential in solving complex optimization problems. They proposed converting the objective function into a network energy function, with variables corresponding to the network state, and obtaining the optimal solution when the energy function converged. Subsequently, scholars have followed this method to study various neurodynamic models for solving optimization problems [8, 11, 39]. Therefore, to solve the optimization problem in Eq. (2) in this paper, the NdS is designed as

$$\begin{cases} \dot{x}_i(t) = -\sum_{j=1}^{N} l_{ij} \text{sig}^{\alpha}(\nabla f_j(x_j(t))) - \\ \qquad \sum_{j=1}^{N} l_{ij} \text{sig}^{\beta}(\nabla f_j(x_j(t))), \\ \sum_{i=1}^{N} x_i(0) = d \end{cases} \tag{3}$$

where $\alpha > 1$ and $0 < \beta < 1$. Each neuron interacts with its neighbors to collaboratively search for the optimal solution. To reduce the auxiliary multipliers in the system and ensure that the trajectories of subsequent state variables remain within the feasible domain, it is necessary to satisfy zero-sum initial constraints. That is

$$\sum_{i=1}^{N} x_i(0) = d,$$
$$\sum_{i=1}^{N} \dot{x}_i(t) = 0.$$

It can be designed by $x(0) = [d/N, d/N, ..., d/N]$. In this way, we define $\nabla F(x(t)) = [\nabla f_1(x_1(t)), \nabla f_2(x_2(t)), ..., \nabla f_N(x_N(t))]^{\text{T}}$. Additionally, it is always guaranteed by Eq. (3) that the equality constraint $\sum_{i=1}^{N} x_i(t) = d$ holds. This is because

$$\begin{aligned} \sum_{i=1}^{N} \dot{x}_i(t) = & -\sum_{i=1}^{N} \sum_{j=1}^{N} l_{ij} \text{sig}^{\alpha}(\nabla f_i(x_j(t))) - \\ & \sum_{i=1}^{N} \sum_{j=1}^{N} l_{ij} \text{sig}^{\beta}(\nabla f_i(x_j(t))) = \\ & -1^{\text{T}} L \text{sig}^{\alpha}(\nabla F(x(t))) - 1^{\text{T}} L \text{sig}^{\beta}(\nabla F(x(t))) = 0 \end{aligned} \tag{4}$$

Therefore, $\sum_{i=1}^{N} x_i(t) = \sum_{i=1}^{N} x_i(0) = d$.

**Remark 2** Currently, several distributed optimization algorithms with equality constraints converge to the optimal solution within a fixed time [19, 20]. However, because of the use of multipliers and auxiliary variables, they involve excessive interactive calculations. Inspired by Ref. [23], a distributed algorithm achieves high efficiency by designing the derivative sum of state variables to always be zero. As long as the initial values satisfy the constraint conditions, these state variables remain in the feasible domain.

**Remark 3** Another form of NdS can also be designed for fixed-time optimization. Similarly to Ref. [27], the system can be modified as

$$\begin{cases} \dot{x}_i(t) = -\sum_{i=1}^{N} l_{ij} \nabla f_j(x_j(t)) \|\nabla f_j(x_j(t))\|^{-\alpha} - \\ \qquad \sum_{i=1}^{N} l_{ij} \nabla f_j(x_j(t)) \|\nabla f_j(x_j(t))\|^{-\beta}, \\ \sum_{i=1}^{N} x_i(0) = d \end{cases} \tag{5}$$

This section proposes two forms of distributed neural dynamic system approximations for convex constraint

optimization. The exponential form can help the system to converge quickly. Combined with optimization conditions, the system can find the optimal solution along the equality constraint conditions.

### *B. Fixed-Time Nonconvex ED Based on CQL*

This section mainly studies a class of nonconvex ED optimization problems, which are characterized by the unknown mathematical form of the objective function and the lack of gradient information. Adopting RL methods, a Q-value table generation learning strategy is established based on the fast search experience of distributed convex optimization to achieve cost minimization.

Consider a static nonconvex optimization problem

$$\begin{aligned} &\min_{x} \sum_{i=1}^{N} f_i(x_i(t)), \\ &\text{s.t.}, \sum_{i=1}^{N} x_i(t) = d, \\ &\quad \underline{x}_i \leqslant x_i(t) \leqslant \overline{x}_i, \\ &\quad |x_i(t) - x_{\text{state},i}| \leqslant x_i^{\text{R}} \end{aligned} \tag{6}$$

where $x(t) = [x_1(t), x_2(t), ..., x_N(t)]$, $x_i(t) \in \mathbb{R}$. $\underline{x}_i$ and $\overline{x}_i$ are the lower and upper bounds on capacity, respectively. $x_i$ represents the variable waiting for decision-making, which is represented as an action within the RL framework. $x_{\text{state},i}$ represents the state value. $x_i^{\text{R}}$ represents the maximum difference between $x_i$ and the previous moment $x_{\text{state},i}$. The above optimization problem is a typical mathematical model for ED of power grids. In the CQL, a Q-value table is established with the value of the objective function as the reward. The meaning of the third constraint in Eq. (6) is that the decision variables need to meet the bounded range of changes in the previous state $x_{\text{state},i}$, in order to ensure electricity safety. The other two constraints represent supply-demand balance and capacity boundary, respectively.

To handle the nonconvex optimization problem, a fixed-time distributed optimization RL algorithm based on the CQL is proposed as follows:

(1) Divide the feasible domain of virtual actions. According to the inequality constraints $\underline{x}_i \leqslant x_i \leqslant \overline{x}_i$ and $|x_i - x_{\text{state},i}| \leqslant x_i^{\text{R}}$, the feasible domain is obtained as $[b_{\min,i}, b_{\max,i}]$, where $b_{\min,i} = \max\{\underline{x}_i, x_{\text{state},i} - x_i^{\text{R}}\}$ and $b_{\max,i} = \min\{\overline{x}_i, x_{\text{state},i} + x_i^{\text{R}}\}$. The above can refer to Fig. 1. The action space is discretized into $M$ parts. $M+1$ virtual actions are generated as $A_i = \{b_{\min,i}, b_{2,i}, ..., b_{M,i}, b_{\max,i}\}$.

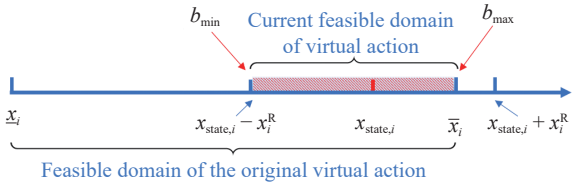



**Figure 1** Schematic diagram of feasible domain for virtual action.

**Remark 4** The feasible domain is redefined. Instead of adding the ramp rate constraint [40] to the distributed optimization algorithms, the algorithm adds it to the RL framework. This reduces the computational complexity of interaction variables and also reduces the search space for feasible domains.

(2) Train Q-value table with greedy strategy. A distributed convex optimization algorithm in Eq. (3) is established to search for optimal values around virtual actions. Note the reward

$$r = K - f(a_{\text{virtual}}) \tag{7}$$

corresponding virtual action $a_{\text{virtual}}$ is recorded and used to generate a Q-value table. The following is abbreviated as $Q_i = Q_i(x_{\text{state},i}, x_{\text{action},i})$, then

$$Q_i \leftarrow Q_i + \alpha(r_i - Q_i) \tag{8}$$

where $\alpha$ is the learning rate.

In a nonconvex optimization problem, the mathematical description of searching for real action $x_i^*$ around virtual action $a_{\text{virtual},i}$ is as follow

$$\begin{aligned} &\min_{x} \ \frac{1}{2} \sum_{i=1}^{N} (x_i(t) - a_{\text{virtual},i})^2, \\ &\text{s.t.}, \sum_{i=1}^{N} x_i(t) = d, \\ &\quad \underline{x}_i \leqslant x_i(t) \leqslant \overline{x}_i, \\ &\quad |x_i(t) - x_{\text{state},i}| \leqslant x_i^{\text{R}} \end{aligned} \tag{9}$$

Based on Eq. (3), the distributed algorithm for solving Eq. (9) is

$$\begin{cases} \dot{x}_i(t) = -\sum_{i=1}^{N} l_{ij} \text{sig}^{\alpha}(x_i(t) - a_{\text{virtual},i}) - \\ \qquad \sum_{i=1}^{N} l_{ij} \text{sig}^{\beta}(x_i(t) - a_{\text{virtual},i}), \\ \sum_{i=1}^{N} x_i(0) = d \end{cases} \tag{10}$$

Then update the reward function and Q-value table based on this result. The greedy strategy gives other nonoptimal virtual actions a chance to be selected.

**Remark 5** Unlike the asymptotic convergence of NdS in Ref. [29], the proposed system converges exponentially to the optimal value, allowing for effective calculation of reward values. In addition, the proposed algorithm converges to the optimal solution in a fixed-time, which is different from the focus of some recent distributed algorithms that achieve consensus in a fixed/predefined-time.

(3) After the training is completed, the optimal virtual action $a_{\text{virtual},i}^*$ is selected in the Q-value table. Then, the distributed convex optimization algorithm in Eq. (10) is used again to calculate the actual actions $x^*$ searched from the surroundings. Finally, the cost function $f(x^*)$ of the actual action $x^*$ is obtained. The specific process is described in Algorithm 1.

**Algorithm 1** Fixed-time distributed optimization for nonconvex ED

**Require:** $M$, $l$, $k_\epsilon$, $x(0)$, $x^{\mathrm{R}}$, $\underline{x}$, $\overline{x}$, and $x_{\mathrm{state},i}$

**Ensure:** $x^*$

Allocate feasible range intervals $[b_{\min,i}, b_{\max,i}]$ for virtual action:

$b_{\min,i} \leftarrow \max\{\underline{x}_i, x_{\mathrm{state},i} - x_i^{\mathrm{R}}\}$;

$b_{\max} \leftarrow \min\{\overline{x}_i, x_{\mathrm{state},i} + x_i^{\mathrm{R}}\}$;

Create Q-value tables with a size of $1 \times (M+1)$;

**for** $k = 1 : l$ **do**

Select the optimal virtual action:

$$a_{\mathrm{virtual},i} = \begin{cases} a^*_{\mathrm{virtual},i}, & \epsilon \mathrm{e}^{-k_\epsilon l}; \\ a_i \in A_i \backslash a^*_{\mathrm{virtual},i}, & 1 - \epsilon \mathrm{e}^{-k_\epsilon l}. \end{cases}$$

Obtain the optimal solution $x^* \leftarrow$ Eq. (8);

Calculate the cost function $f(x^*)$;

Reward can be calculated using Eq. (7) or other forms;

Fill the reward value into the corresponding virtual action position in the Q-value table;

**end for**

Determine the optimal virtual action $a^*_{\mathrm{virtual},i}$ from the Q-value table;

**print** $x^* \leftarrow$ Eq. (8).

An alternative method on the basis of penalty functions is provided below. However, the result is approximate, limited by the penalty coefficient $\theta$ and the number of agents $N$. The detailed proof and error can be seen in Ref. [23].

**Remark 6** The NdM-based distributed algorithm in Eq. (3) is combined with RL method [29] to solve nonconvex optimization. Among them, the original virtual action allocation method has been redefined to meet inequality constraints. Then, the calculation of Q-value is based on the NdS that converges to the optimal solution in a fixed time, rather than asymptotic convergence. It is beneficial for shortening the accumulation time of Q-value.

**Remark 7** Consider a static convex optimization problem in Eq. (6). Two inequality constraints are denoted as

$$g_{i,1}(x_i(t)) = x_i(t) - \min(\overline{x}_i, x_{\mathrm{state},i} + x_i^{\mathrm{R}}),$$
$$g_{i,2}(x_i(t)) = x_i(t) - \max(\underline{x}_i, x_{\mathrm{state},i} - x_i^{\mathrm{R}}).$$

Introduce penalty functions

$$h_i(x_i(t)) = f(x_i(t)) - \frac{1}{\theta}\sum_{k=1}^{2} \ln(g_{i,k}(x_i(t))),$$

or

$$h_i(x_i(t)) = f(x_i(t)) - \frac{1}{\theta}\sum_{k=1}^{2} \|x_i - P_\Omega(x_i)\|^2,$$

where $h_i(x_i(t))$ is a convex function. The distributed optimization algorithm is designed as

$$\begin{cases} \dot{x}_i(t) = -\sum_{i=1}^{N} l_{ij}\mathrm{sig}^{\alpha}(\nabla h_j(x_j(t))) - \\ \qquad \sum_{i=1}^{N} l_{ij}\mathrm{sig}^{\beta}(\nabla h_j(x_j(t))), \\ \sum_{i=1}^{N} x_i(0) = d \end{cases} \tag{11}$$

The provided penalty function method avoids increasing the complexity of the algorithm by dealing with inequality constraints. It is an approximate optimization method.

This section investigates the nonconvex optimization of ED. Equation (3) is combined with reinforcement learning, and two search methods for dealing with inequality constraints are proposed, namely, redefining box constraints into the feasible domain of virtual actions or introducing penalty functions in distributed algorithms. When the latter algorithm approaches the constraint boundary, the value of the logarithmic potential barrier function rapidly increases. Choosing the appropriate penalty coefficient is also a valuable topic.

## IV. Convergence Analysis

In this section, a Lyapunov function is used to analyze the convergence of the proposed NdS.

**Theorem 1** Under Assumptions 1, 2, and 4, the optimization problem in Eq. (2) is solved utilizing algorithm in Eq. (3) within the time $T_1$, where

$$T_1 = \frac{\pi}{\left(\frac{\alpha-\beta}{2}\right)\left(4m^2\lambda_2\left(L_\beta\right)\right)^{\frac{\beta+1}{2}}} \csc\left(\frac{1-\beta}{\alpha-\beta}\pi\right) \times \left(\frac{\left(4m^2\right)^{\frac{\beta+1}{\alpha+1}} \left(\lambda_2\left(L_\beta\right)\right)^{\frac{\beta+1}{2}}}{N^{\frac{1-\alpha}{2}} \left(\lambda_2\left(L_\alpha\right)\right)^{\frac{\alpha+1}{2}}}\right)^{\frac{1-\beta}{\alpha-\beta}}.$$

**Proof** Define the Lyapunov function $V(t) = f(x(t)) - f(x^*)$, then

$$\begin{aligned} &\dot{V}(x(t)) = \\ &\sum_{i=1}^{N} \nabla f_i(x_i(t)) \left( -\sum_{i=1}^{N} l_{ij}\mathrm{sig}^{\alpha}\left(\nabla f_j(x_i(t))\right) - \right. \\ &\left. \sum_{i=1}^{N} l_{ij}\mathrm{sig}^{\beta}\left(\nabla f_j\left(x_j(t)\right)\right) \right) = \\ &-\sum_{i=1}^{N}\sum_{j=1}^{N} \nabla f_i(x(t)) l_{ij}\mathrm{sig}^{\alpha}\left(\nabla f_j\left(x_j(t)\right)\right) - \\ &\sum_{i=1}^{N}\sum_{j=1}^{N} \nabla f_i(x(t)) l_{ij}\mathrm{sig}^{\beta}\left(\nabla f_j\left(x_j(t)\right)\right) = \\ &-\sum_{i=1}^{N}\sum_{j=1}^{N} \left( l_{ij}^{\frac{2}{\alpha+1}} \|\nabla f_j\left(x_j(t)\right)\|^2 \right)^{\frac{\alpha+1}{2}} - \\ &\sum_{i=1}^{N}\sum_{j=1}^{N} \left( l_{ij}^{\frac{2}{\beta+1}} \|\nabla f_j\left(x_j(t)\right)\|^2 \right)^{\frac{\beta+1}{2}} \overset{\text{(a)}}{\leqslant} \\ &-N^{\frac{1-\alpha}{2}} \left( \sum_{i=1}^{N}\sum_{j=1}^{N} l_{ij}^{\frac{2}{\alpha+1}} \|\nabla f_j\left(x_j(t)\right)\|^2 \right)^{\frac{\alpha+1}{2}} - \\ &\left( \sum_{i=1}^{N}\sum_{j=1}^{N} l_{ij}^{\frac{2}{\beta+1}} \|\left(\nabla f_j\left(x_j(t)\right)\right)\|^2 \right)^{\frac{\beta+1}{2}} \end{aligned} \tag{12}$$

By Lemma 2, inequality (a) holds. $L_\alpha(L_\beta)$ represents a Laplace matrix, which is composed of $-a_{ij}^{2/(\alpha+1)}(-a_{ij}^{2/(\beta+1)})$ for each nondiagonal element and $\sum_{i=1,i\neq j}^{N} -a_{ij}^{2/(\alpha+1)}(\sum_{i=1,i\neq j}^{N} -a_{ij}^{2/(\beta+1)})$

for diagonal elements. Then, $1_n^{\mathrm{T}} L_\alpha (1_n^{\mathrm{T}} L_\beta) = 0$. Let $\nabla \bar{f}(x(t)) = ((1_n^{\mathrm{T}} \nabla f(x(t)))/n) 1_n$, we have

$$\begin{aligned}
&(\nabla f(x(t)) - \nabla \bar{f}(x(t)))^{\mathrm{T}} L_\alpha (\nabla f(x(t)) - \nabla \bar{f}(x(t))) = \\
&(\nabla f(x(t)) - \frac{1_n^{\mathrm{T}} \nabla f(x(t))}{n} 1_n)^{\mathrm{T}} L_\alpha \times \\
&(\nabla f(x(t)) - \frac{1_n^{\mathrm{T}} \nabla f(x(t))}{n} 1_n) = \\
&\nabla f(x(t))^{\mathrm{T}} L_\alpha (\nabla f(x(t)) - \frac{1_n^{\mathrm{T}} \nabla f(x(t))}{n} 1_n) - \\
&\frac{1_n^{\mathrm{T}} \nabla f(x(t))}{n} 1_n^{\mathrm{T}} L_\alpha (\nabla f(x(t)) - \frac{1_n^{\mathrm{T}} \nabla f(x(t))}{n} 1_n) = \\
&\nabla f(x(t))^{\mathrm{T}} L_\alpha \nabla f(x(t)) - \frac{1_n^{\mathrm{T}} \nabla f(x(t))}{n} \nabla f(x(t))^{\mathrm{T}} L_\alpha 1_n - \\
&\frac{1_n^{\mathrm{T}} \nabla f(x(t))}{n} 1_n^{\mathrm{T}} L_\alpha (\nabla f(x(t)) - \frac{1_n^{\mathrm{T}} \nabla f(x)}{n} 1_n) = \\
&\nabla f(x(t))^{\mathrm{T}} L_\alpha \nabla f(x(t))
\end{aligned} \tag{13}$$

Note $\phi(t) = \nabla f(x(t)) - \nabla \bar{f}(x(t))$. According to Eq. (13), we have

$$\begin{aligned}
\dot{V} \leqslant & -N^{\frac{1-\alpha}{2}} (\nabla f(x(t))^{\mathrm{T}} L_\alpha \nabla f(x(t)))^{\frac{\alpha+1}{2}} - \\
& (\nabla f(x(t))^{\mathrm{T}} L_\beta \nabla f(x(t)))^{\frac{\beta+1}{2}} = \\
& -N^{\frac{1-\alpha}{2}} ((\nabla f(x(t)) - \nabla \bar{f}(x(t)))^{\mathrm{T}} L_\alpha (\nabla f(x(t)) - \\
& \nabla \bar{f}(x(t))))^{\frac{\alpha+1}{2}} - ((\nabla f(x(t)) - \nabla \bar{f}(x(t)))^{\mathrm{T}} \times \\
& L_\beta (\nabla f(x(t)) - \nabla \bar{f}(x(t))))^{\frac{\beta+1}{2}} = \\
& -N^{\frac{1-\alpha}{2}} (\phi(t)^{\mathrm{T}} L_\alpha \phi(t))^{\frac{\alpha+1}{2}} - (\phi(t)^{\mathrm{T}} L_\beta \phi(t))^{\frac{\beta+1}{2}} \overset{\text{(b)}}{\leqslant} \\
& -N^{\frac{1-\alpha}{2}} (2m\lambda_2(L_\alpha))^{\frac{\alpha+1}{2}} (\phi(t)^{\mathrm{T}} \phi(t))^{\frac{\alpha+1}{2}} - \\
& (2m\lambda_2(L_\beta))^{\frac{\beta+1}{2}} (\phi(t)^{\mathrm{T}} \phi(t))^{\frac{\beta+1}{2}}
\end{aligned} \tag{14}$$

According to Lemma 3, the inequality relationship of (b) holds. Furthermore, from Lemma 4, it can be concluded that

$$\begin{aligned}
& f(x^*) - f(x(t)) \geqslant \\
& \nabla f(x(t))^{\mathrm{T}} (x^* - x(t)) + \frac{m}{2} \|x^* - x(t)\|^2 = \\
& (\nabla f(x(t)) - \nabla \bar{f}(x(t)))^{\mathrm{T}} (x^* - x(t)) + \frac{m}{2} \|x^* - x(t)\|^2 \overset{\text{(c)}}{\geqslant} \\
& -\frac{1}{2m} \|\nabla f(x(t)) - \nabla \bar{f}(x(t))\|^2
\end{aligned} \tag{15}$$

According to the Young's inequality, we have

$$\begin{aligned}
& \frac{m}{2} \|x(t) - x^*\|^2 + \frac{1}{2m} \|\nabla f(x(t)) - \nabla \bar{f}(x(t))\| \geqslant \\
& (\nabla f(x(t)) - \nabla \bar{f}(x(t)))^{\mathrm{T}} (x(t) - x^*).
\end{aligned}$$

Then, we have

$$\begin{aligned}
& (\nabla f(x(t)) - \nabla \bar{f}(x(t)))^{\mathrm{T}} (x^* - x(t)) + \frac{m}{2} \|x(t) - x^*\|^2 \geqslant \\
& -\frac{1}{2m} \|\nabla f(x(t)) - \nabla \bar{f}(x(t))\|^2.
\end{aligned}$$

The inequality relationship of (c) holds. Thus, $2mV(t) \leqslant \phi^{\mathrm{T}}(t)\phi(t)$. By substituting Eq. (13), we obtain

$$\begin{aligned}
\dot{V}(t) \leqslant & -N^{\frac{1-\alpha}{2}} (4m^2 \lambda_2(L_\alpha))^{\frac{\alpha+1}{2}} V(t)^{\frac{\alpha+1}{2}} - \\
& (4m^2 \lambda_2(L_\beta))^{\frac{\beta+1}{2}} V(t)^{\frac{\beta+1}{2}}.
\end{aligned}$$

By Lemma 1, the distributed fixed-time optimization in Eq. (2) is achieved within time $T_1$. ■

**Lemma 6** Assumptions 1, 2, and 4 hold. The optimization problem in Eq. (2) is solved by a distributed algorithm in Eq. (5) within time

$$\begin{aligned}
T_2 = & \frac{\pi}{\left(\frac{\alpha-\beta}{2}\right)\left(4m^2\lambda_2\left(L_\beta\right)\right)^{\frac{\beta+1}{2}}} \csc\left(\frac{1-\beta}{\alpha-\beta}\pi\right) \times \\
& \left( \left(4m^2\right)^{\frac{\beta+1}{\alpha+1}} \frac{\left(\lambda_2\left(L_\beta\right)\right)^{\frac{\beta+1}{2}}}{N^{\frac{1-\alpha}{2}} \left(\lambda_2\left(L_\alpha\right)\right)^{\frac{\alpha+1}{2}}} \right)^{\frac{1-\beta}{\alpha-\beta}}.
\end{aligned}$$

**Proof** Define the Lyapunov function $V(t)=f(x(t))-f(x^*)$, then

$$\begin{aligned}
\dot{V}(t) = & \sum_{i=1}^{N} \nabla f_i(x_i(t)) (- \sum_{i=1}^{N} l_{ij} \nabla f_j(x(t)) \|\nabla f_j(x_j(t))\|^{-\alpha} - \\
& \sum_{i=1}^{N} l_{ij} \nabla f_j(x_j(t)) \|\nabla f_j(x_j(t))\|^{-\beta}) - \\
& \sum_{i=1}^{N} \sum_{j=1}^{N} \nabla f_i(x_i(t)) l_{ij} \nabla f_j(t) \|\nabla f_j(x_j(t))\|^{-\alpha} - \\
& \sum_{i=1}^{N} \sum_{j=1}^{N} \nabla f_i(x_i(t)) l_{ij} \nabla f_j(x_j(t)) \|\nabla f_j(x_j(t))\|^{-\beta}
\end{aligned} \tag{16}$$

The proof process is the same as shown in Theorem 1. ■

## V. Numerical Simulation

In this section, Example 1 is used to verify the effectiveness of the algorithm in a large-scale convex optimization with an equality constraint. Example 2 corresponds to the fixed-time optimization investigated in Ref. [20]. Example 3 compares the different search optimal solution processes of three distributed optimization algorithms using the example in Ref. [41]. Example 4 utilizes the proposed distributed convex optimization algorithm as a search tool for nonconvex optimization in Ref. [29], which implements the algorithm acceleration.

**Example 1** The undirected graph with 100 nodes for Eq. (3) is considered.

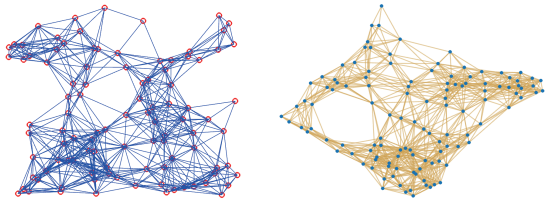
$$\begin{aligned}
& \min_x \quad \sum_{i=1}^{100} f_i(x_i(t)), \\
& \text{s.t.,} \quad \sum_{i=1}^{100} x_i(t) = 150
\end{aligned} \tag{17}$$

where $f_i(x_i(t)) = (x_i(t) - 0.1)^2/2, i = 1, 2, ..., 20$, $f_i(x_i(t)) = (x_i(t) - 0.3)^2/2$, $i = 21, 22, ..., 60$, and $f_i(x_i(t)) = (x_i(t) - 0.5)^2/2$, $i = 61, 62, ..., 100$. This example is used to simulate the simplified situation of large-scale nodes, usually applicable to problems such as resource allocation, ED, engineering optimization, etc.

Random values between 0 and 2.5 are chosen for $x_1(0), x_2(0), ..., x_{80}(0)$. The vector $A_{\text{label}}$ is labeled as $[x_1(0), x_2(0), ..., x_{80}(0)]$. The value $150 - \text{sum}(A_{\text{label}})$ is then distributed equally among the remaining 20 numbers $x_{81}(0), x_{82}(0), ..., x_{100}(0)$.

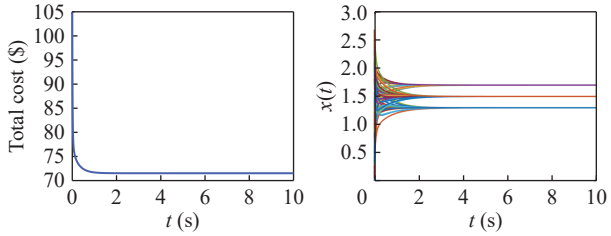
Figure 2 shows the topology relationship diagram of 100 nodes. While maintaining connectivity, the connection relationships between nodes are randomly generated. Figure 3 displays the variation curve of the cost function. Because of

the inclusion of exponential terms in distributed algorithms, the cost function quickly approaches the minimum value that satisfies the equality constraint along the gradient descent direction. Figure 3 also shows the trajectory of the decision variable $x(t)$ in the algorithm. It can be seen that the trajectory converges quickly. The final state of the trajectory is in three situations. This is because there are three types of objective functions.



(a) Example 1 (b) Example 3

**Figure 2** Topology diagram of examples.




(a) Cost function (b) Optimal solution

**Figure 3** Trajectory of cost function and optimal solution.

**Example 2** Consider a microgrid system with six generators [20]. This example is used to compare convex optimization algorithms with equality constraints.

$$\begin{aligned} \min_{x} \quad & \sum_{i=1}^{6}(a_i x_i^2(t)+b_i x_i(t)+c_i), \\ \text{s.t.,} \quad & \sum_{i=1}^{6} x_i(t)=600 \end{aligned} \tag{18}$$

The optimization problem in Eq. (18) is solved by Eq. (3), with parameters $\alpha=0.8$ and $\beta=2.0$ set to be consistent with Ref. [20]. The term coefficients in distributed algorithms are uniformly set to 1, because they can affect the convergence speed. The coefficients of the objective function and the initial values of $x$ are shown in Table 1.

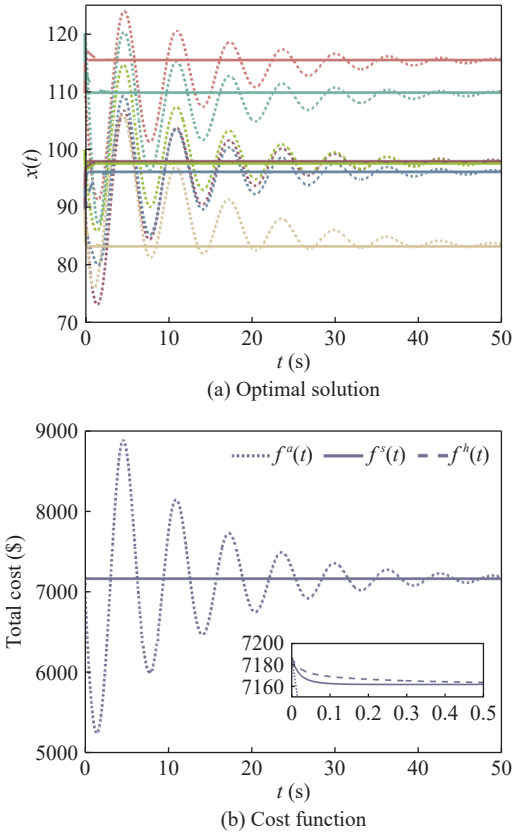
Figures 4–7 show a comparison of the convergence trajectories of two fixed-time NdMs and a classical asymptotic time NdM based on the Lagrange function method to the optimal solution. Figures 4–8 show the comparison of cost function trajectories of three methods. Among them, the superscript $a$ represents the NdS that asymptotically converges to the optimal solution, the superscript $s$ represents the system in Eq. (3), and the superscript $h$ represents the NdS proposed by Ref. [20]. The cost function value after obtaining the dispatch is 7162.04. The optimal solutions are $x_1^*=79.9218$, $x_2^*=115.3540$, $x_3^*=66.9500$, $x_4^*=97.5670$, $x_5^*=109.7500$, and $x_6^*=96.1166$. For convex optimization problems with equality constraints, algorithms $h$ and $s$ reach their optimal values in a short time $t$. Algorithm $a$ reaches its optimum after a long period of fluctuation $t$. The changes in three algorithm parameters and their corresponding displays are shown in Table 2.

**Table 1** Parameter of ED in Eq. (18).

| Generator number | $a_i$ | $b_i$ | $\bar{x}_i$ | $x_i(0)$ |
|---|---|---|---|---|
| 1 | 0.077 | 20 | 78.78 | 90 |
| 2 | 0.010 | 40 | 38.00 | 110 |
| 3 | 0.250 | 20 | 33.00 | 90 |
| 6 | 0.010 | 40 | 45.00 | 100 |
| 8 | 0.022 | 20 | 556.50 | 120 |
| 9 | 0.010 | 40 | 30.50 | 90 |
| 12 | 0.032 | 20 | 45.00 | 90 |

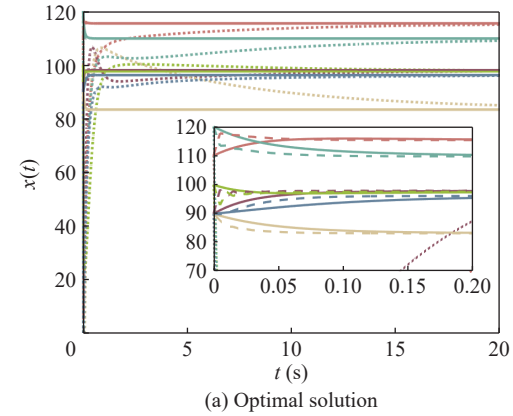
**Table 2** Corresponding diagram for adjusting three algorithm parameters.

| Generator | $k^s$ | $k^h$ | $k^a$ | Step size |
|---|---|---|---|---|
| Figure 4 | 1 | $[1,1,1]^{\mathrm{T}}$ | 1 | 0.005,00 |
| Figure 5 | 1 | $[40,20,1]^{\mathrm{T}}$ | $1\times10^3$ | 0.005,00 |
| Figure 6 | 1 | $[100,20,20]^{\mathrm{T}}$ | $1\times10^4$ | 0.000,83 |
| Figures 7 and 8 | 1 | $[40,20,1]^{\mathrm{T}}$ | — | 0.005,00 |
| | 50 | $[40,30,1]^{\mathrm{T}}$ | — | 0.005,00 |
| | 500 | $[50,20,1]^{\mathrm{T}}$ | — | 0.005,00 |

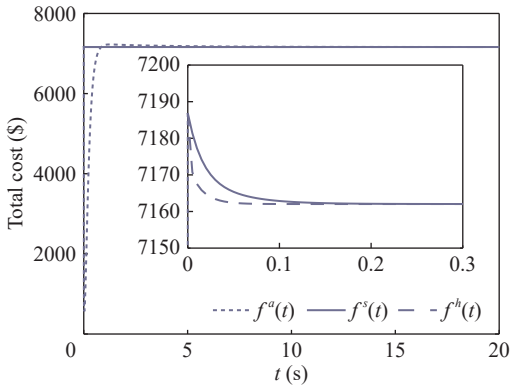



(a) Optimal solution (b) Cost function

**Figure 4** Comparison of three algorithms with coefficients equal to 1.

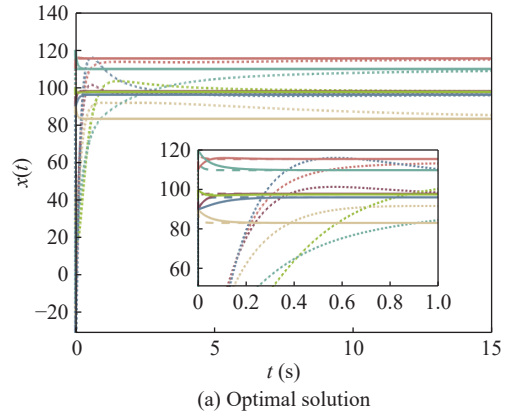



(a) Optimal solution

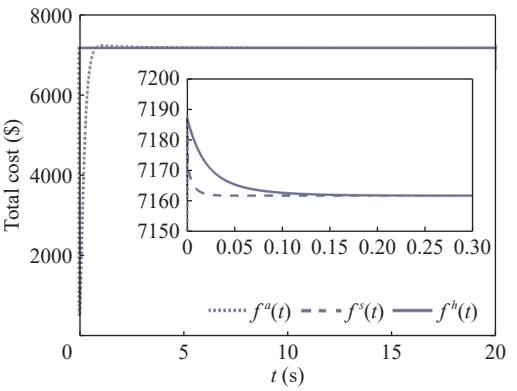



(b) Cost function

**Figure 5** Comparison of our algorithm ($k^s = 1$) with two comparison algorithms (maximum adjustable situation at step size 0.005).




(a) Optimal solution




(b) Cost function

**Figure 6** Comparison of our algorithm ($k^s = 1$) with two comparison algorithms (data from Ref. [20] or maximum adjustable situation at step size 0.000,83).

Compared with algorithms with the same coefficients, it can be seen that our algorithm converges faster. The reason is that we directly use the gradient of the objective function to reach the optimal solution faster. Meanwhile, the state variables are not affected by the convergence of other auxiliary variables.

Figure 5 shows the comparison between algorithm $s$ with parameter 1 and other algorithms. At a step size of 0.05, the coefficient of algorithm $h$ increases to $k^h = [40, 20, 1]^{\mathrm{T}}$, and algorithm $a$ increases to $1 \times 10^4$. For larger parameters, the system will lose control. The results demonstrate that two fixed-time methods converge to the optimal solution early.

Subsequently, the step size is reduced. The coefficient of each algorithm can be further increased. The coefficient of algorithm $h$ is changed to $k^h = [100, 20, 20]^{\mathrm{T}}$, which is consistent with the setting in Ref. [20]. The coefficient of algorithm $a$ increases to $1 \times 10^4$. The coefficient of algorithm $s$ remains at 1. The results show that the convergence speed of algorithm $a$ is improved, but the effect is still inferior to algorithm $h$ and algorithm $s$.

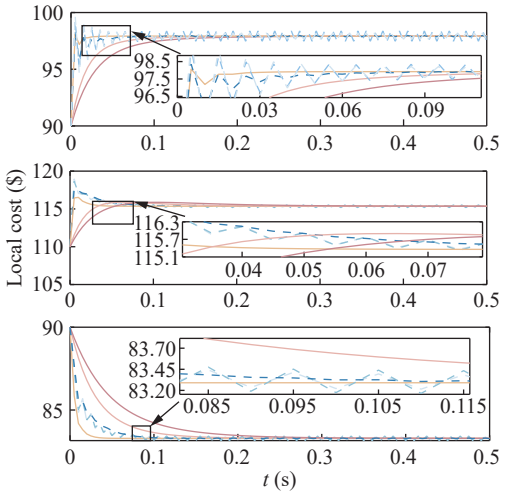
Finally, algorithm $h$ and algorithm $s$ are compared in detail. Three sets of parameters are set for each algorithm. It is worth noting that Table 2 does not display the maximum coefficient of algorithm $s$. Under the current step size limitation, the coefficient of algorithm $s$ can still be further increased. But the latter two parameters of algorithm $h$ experience significant fluctuations. The specific value is that when position 1 exceeds 50 or position 2 exceeds 30, the proposed system will lose control. This demonstrates that algorithm $s$ has greater flexibility and stronger robustness. Figure 8 shows the cost function trajectories for these six scenarios, which ultimately converge to the same optimal solution.

**Example 3** This example refers to the IEEE 57-bus system in Ref. [41]. This example is used to compare convex optimization algorithms with equality and inequality constraints. The fixed-time algorithm coefficients are set to $\alpha = 1.5$ and $\beta = 0.5$. The system consists of 7 generators. The mathematical description of the optimization problem is

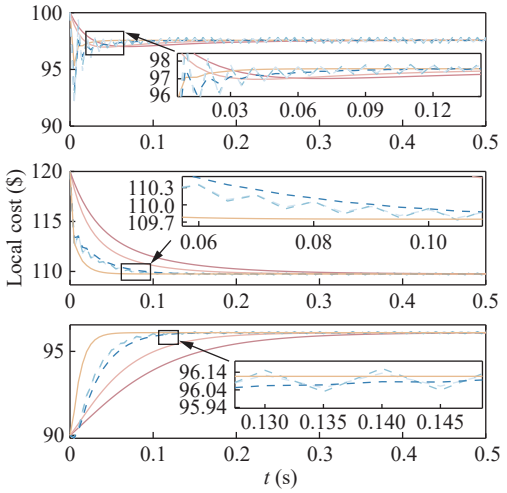
$$\begin{aligned} \min_x \quad & \sum_{i=1}^{7} (a_i x_i^2(t) + b_i x_i(t)), \\ \text{s.t.,} \quad & \sum_{i=1}^{7} x_i(t) = 235.26, \\ & 0 \leqslant x_i(t) \leqslant \overline{x}_i \end{aligned} \tag{19}$$

The cost function and initial value data are shown in Table 3. After adding inequality constraints, the power-law term for $\beta$ becomes more sensitive. Equation (11) about $x_i(t)$ is rewritten as $\dot{x}_i(t) = -\sum_{i=1}^{N} l_{ij} \mathrm{sig}^{\alpha}(\nabla h_j(x_j(t))) - k^s \sum_{i=1}^{N} l_{ij} \mathrm{sig}^{\beta}(\nabla h_j(x_j(t)))$. Among them, the increase of coefficient $k^s$ is to reflect the sensitivity of this term, with values of $k_1^s = 0.5$, $k_2^s = 0.8$, and $k_3^s = 1.0$, respectively.

Figure 9 shows the cost function value. There is still shaking at the minimum value after convergence. The result




(a) $x_1$, $x_2$, and $x_3$




(b) $x_4$, $x_5$, and $x_6$

**Figure 7** Comparison of our algorithm ($k^s = 1, 50$, and 500) with comparison algorithm (adjustable situation at step size 0.005).

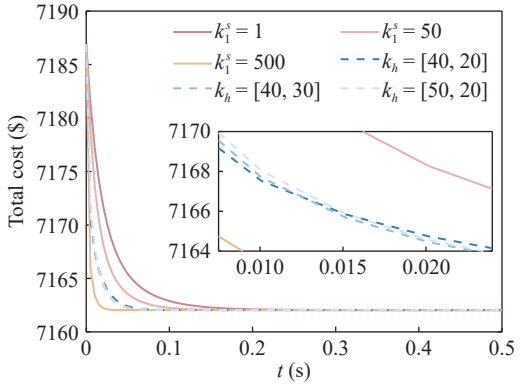


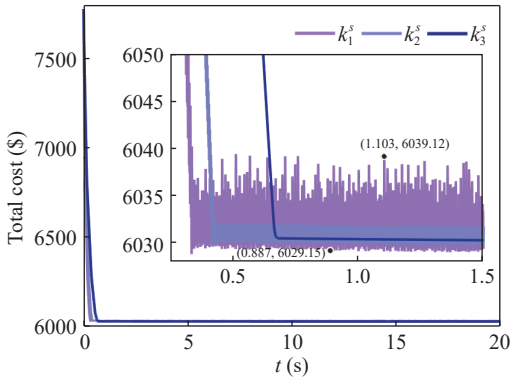

**Figure 8** Trajectory of cost function.

has a certain error with the optimal value 6036.40 in Ref. [41]. Due to the introduction of the sign function, the optimal value of the graph fluctuates in [6029.15, 6039.12]. As $k^s$ gradually decreases, the fluctuation range decreases accordingly. When $k_1^s = 0.5$, the value stabilizes around the optimal solution 6030. Figure 9 also shows the optimal solution. The

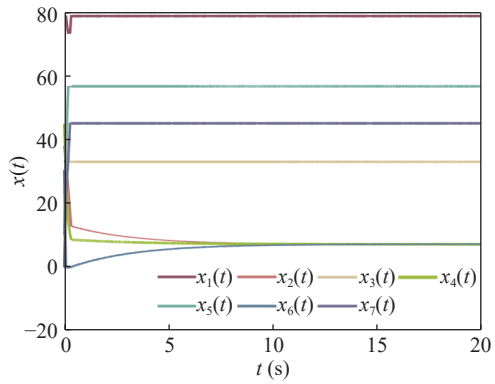
**Table 3** Parameter of ED in Eq. (19).

| Generator | $a_i$ | $b_i$ | $c_i$ | $x_i(0)$ |
|---|---|---|---|---|
| 1 | 0.096 | 1.22 | 51 | 90 |
| 2 | 0.072 | 3.41 | 31 | 110 |
| 3 | 0.105 | 2.53 | 78 | 90 |
| 4 | 0.082 | 4.02 | 42 | 100 |
| 5 | 0.078 | 2.90 | 57 | 120 |
| 6 | 0.090 | 2.72 | 49 | 90 |

fluctuation is due to the chattering phenomenon caused by the sign function. To control accuracy, the proportion of the sign term can be reduced, but it will also increase the convergence time.




(a) Cost function




(b) Optimal solution

**Figure 9** Trajectory of cost function and optimal solution.

**Example 4** In this example, a 120-unit case is generated, which is three times the size of a 40-unit case [40] with valve point loading. This example is provided to simulate large-scale nonconvex optimization problems in real-world scenarios. 120 units of the communication network are randomly generated in Fig. 2.

$$\begin{aligned} &\min_x \sum_{i=1}^{120} f_i(x_i(t)), \\ &\text{s.t.,} \sum_{i=1}^{120} x_i = 16{,}000, \\ &\underline{x}_i \leqslant x_{i,t} \leqslant \overline{x}_i, \\ &|x_{i,t} - x_{i,t-1}| \leqslant 50 \end{aligned} \tag{20}$$

The form of the local objective function is

$$f_i(p_{i,t}) = \sum_{j=1}^{120}(\bar{a}_{ij}p_{i,t}^2 + \bar{b}_{ij}p_{i,t} + \bar{c}_{ij} + |\bar{e}_{ij}\sin(\bar{f}_{ij}(\underline{p}_{ij} - p_{i,t}))|), \quad \text{and}$$

the parameters are given in Ref. [40].

Figure 10 shows the asymptotic convergence of the optimization problem in Eq. (20) to the minimum cost, using the estimation of the optimal cost in Ref. [29]. However, our distributed optimization algorithm in Fig. 10 completes a cumulative reward during the training process within less than 0.2 s. It helps to quickly accumulate rewards in RL and reduces the time burden of accumulating rewards. This is of great significance for the rapid response of allocation. The solving algorithm for this example consists of the RL framework in Ref. [29], where the distributed algorithm used is accelerated. This can more effectively accumulate rewards and quickly find nonconvex optimization solutions.

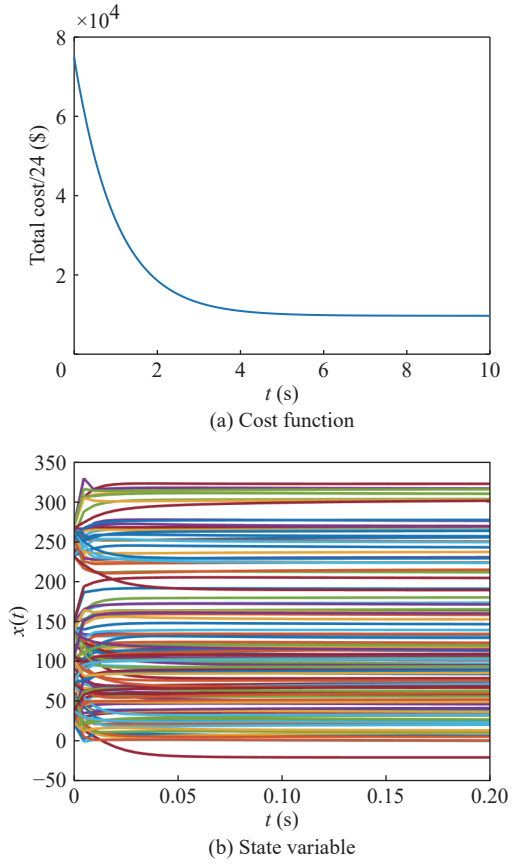



**Figure 10** Trajectory of cost function and state variable $x(t)$.

## VI. Conclusion

An NdS is designed for convex optimization with an equality constraint. This will significantly improve the processing efficiency of engineering tasks. It can be integrated with RL to solve a nonconvex ED problem. The primary advantage lies in its ability to efficiently and repeatedly search for feasible solutions. First, the state variables in the differential equation need to satisfy the zero-sum initial constraint to ensure that the equality constraint always holds. Second, the ramp rate constraints and capacity constraints in ED are considered in the partitioning of feasible states within the RL framework, reducing the inequality constraint related variables in distributed optimization algorithms. In the future, applications in dynamic ED problems warrant further exploration.

## Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 61973070 and 62373089), the Nature Science Foundation of Liaoning Province, China (No. 2022JH25/10100008), and the SAPI Fundamental Research Funds, China (No. 2018ZCX22).

## References

[1] G. Binetti, A. Davoudi, D. Naso, B. Turchiano, and F. L. Lewis, A distributed auction-based algorithm for the nonconvex economic dispatch problem, *IEEE Trans. Ind. Inform.*, 2014, 10(2), 1124–1132.

[2] P. Elbert, T. Nüesch, A. Ritter, N. Murgovski, and L. Guzzella, Engine on/off control for the energy management of a serial hybrid electric bus via convex optimization, *IEEE Trans. Veh. Technol.*, 2014, 63(8), 3549–3559.

[3] X. Sui, Z. Jiang, Y. Lyu, R. Fan, H. Hu, and Z. Liu, Integrating convex optimization and deep learning for downlink resource allocation in LEO satellites networks, *IEEE Trans. Cognit. Commun. Netw.*, 2024, 10(3), 1104–1118.

[4] P.-Y. Chen and I. W. Selesnick, Group-sparse signal denoising: Non-convex regularization, convex optimization, *IEEE Trans. Signal Process.*, 2014, 62(13), 3464–3478.

[5] X. Ren, D. Li, Y. Xi, and H. Shao, Distributed global optimization for a class of nonconvex optimization with coupled constraints, *IEEE Trans. Autom. Control*, 2022, 67(8), 4322–4329.

[6] Z. Li and Z. Ding, Distributed multiobjective optimization for network resource allocation of multiagent systems, *IEEE Trans. Cybern.*, 2021, 51(12), 5800–5810.

[7] C. Su, X. Wang, S. Wang, H. Dai, and J. Chang, Discrete predefined-time distributed optimization algorithm for resource allocation problem in smart grids, in *Proc. 2024 36th Chinese Control and Decision Conference*, Xi'an, China, 2024, 479–484.

[8] Z. Chen, J. Wang, and Q.-L. Han, A collaborative neurodynamic optimization approach to distributed chiller loading, *IEEE Trans. Neural Netw. Learn. Syst.*, 2024, 35(8), 10950–10960.

[9] H. Li and S. Qin, A neurodynamic approach for solving time-dependent nonlinear equation system: A distributed optimization perspective, *IEEE Trans. Ind. Inform.*, 2024, 20(8), 10031–10039.

[10] Y. Huang, Z. Meng, J. Sun, and W. Ren, A unified distributed method for constrained networked optimization via saddle-point dynamics, *IEEE Trans. Autom. Control*, 2024, 69(3), 1818–1825.

[11] S. Yang, Q. Liu, and J. Wang, A collaborative neurodynamic approach to multiple-objective distributed optimization, *IEEE Trans. Neural Netw. Learn. Syst.*, 2018, 29(4), 981–992.

[12] X. Zeng, J. Chen, and Y. Hong, Distributed optimization design for computation of algebraic Riccati inequalities, *IEEE Trans. Cybern.*, 2022, 52(3), 1924–1935.

[13] I. Notarnicola, R. Carli, and G. Notarstefano, Distributed partitioned big-data optimization via asynchronous dual decomposition, *IEEE Trans. Control Netw. Syst.*, 2018, 5(4), 1910–1919.

[14] X. Wang, W. Zheng, and G. Wang, Distributed finite-time optimization of second-order multiagent systems with unknown velocities and disturbances, *IEEE Trans. Neural Netw. Learn. Syst.*, 2023, 34(9), 6042–6054.

[15] M. Chen, D. Wang, X. Wang, Z.-G. Wu, and W. Wang, Distributed aggregative optimization via finite-time dynamic average consensus, *IEEE Trans. Netw. Sci. Eng.*, 2023, 10(6), 3223–3231.

[16] W. Zhu and Q. Wang, Distributed finite-time optimization of multi-agent systems with time-varying cost functions under digraphs, *IEEE Trans. Netw. Sci. Eng.*, 2024, 11(1), 556–565.

[17] H. Liu, W. Zheng, and W. Yu, Continuous-time algorithm based on finite-time consensus for distributed constrained convex optimization, *IEEE Trans. Autom. Control*, 2022, 67(5), 2552–2559.

[18] G. Chen and Z. Guo, Initialization-free distributed fixed-time convergent algorithms for optimal resource allocation, *IEEE Trans. Syst. Man Cybern. Syst*., 2022, 52(2), 845–854.

[19] X. Shi, C. Mu, and C. Sun, A fixed-time distributed constrained optimization algorithm over weight-unbalanced directed network, in *Proc. 2023 International Annual Conference on Complex Systems and Intelligent Science*, Shenzhen, China, 2023, 369–374.

[20] H. Dai, J. Jia, L. Yan, X. Fang, and W. Chen, Distributed fixed-time optimization in economic dispatch over directed networks, *IEEE Trans. Ind. Inform*., 2021, 17(5), 3011–3019.

[21] L. Ma, C. Hu, J. Yu, L. Wang, and H. Jiang, Distributed fixed/preassigned-time optimization based on piecewise power-law design, *IEEE Trans. Cybern*., 2023, 53(7), 4320–4333.

[22] X. Wang, G. Wang, and S. Li, A distributed fixed-time optimization algorithm for multi-agent systems, *Automatica*, 2020, 122, 109289.

[23] C. Li, X. Yu, T. Huang, and X. He, Distributed optimal consensus over resource allocation network and its application to dynamical economic dispatch, *IEEE Trans. Neural Netw. Learn. Syst*., 2018, 29(6), 2407–2418.

[24] Z. Yu, S. Yu, H. Jiang, and X. Mei, Distributed fixed-time optimization for multi-agent systems over a directed network, *Nonlinear Dyn*., 2021, 103(1), 775–789.

[25] L. T. Nguyen, X. Yu, A. Eberhard, and C. Li, Fixed-time gradient dynamics with time-varying coefficients for continuous-time optimization, *IEEE Trans. Autom. Control*, 2023, 68(7), 4383–4390.

[26] Y. Song, J. Cao, and L. Rutkowski, A fixed-time distributed optimization algorithm based on event-triggered strategy, *IEEE Trans. Netw. Sci. Eng*., 2022, 9(3), 1154–1162.

[27] K. Garg and D. Panagou, Fixed-time stable gradient flows: Applications to continuous-time optimization, *IEEE Trans. Autom. Control*, 2021, 66(5), 2002–2015.

[28] K. Li, Q. Hu, Q. Liu, Z. Zeng, and F. Cheng, A predefined-time consensus algorithm of multi-agent system for distributed constrained optimization, *IEEE Trans. Netw. Sci. Eng*., 2024, 11(1), 957–968.

[29] D. Li, L. Yu, N. Li, and F. Lewis, Virtual-action-based coordinated reinforcement learning for distributed economic dispatch, *IEEE Trans. Power Syst*., 2021, 36(6), 5143–5152.

[30] Y. Liu, Z. Xia, and W. Gui, Multiobjective distributed optimization via a predefined-time multiagent approach, *IEEE Trans. Autom. Control*, 2023, 68(11), 6998–7005.

[31] B. Ning, Q. Han, and Z. Zuo, Distributed optimization for multiagent systems: An edge-based fixed-time consensus approach, *IEEE Trans. Cybern*., 2019, 49(1), 122–132.

[32] S. P. Bhat and D. S. Bernstein, Finite-time stability of continuous autonomous systems, *SIAM J. Control Optim*., 2000, 38(3), 751–766.

[33] A. Polyakov, Nonlinear feedback design for fixed-time stabilization of linear control systems, *IEEE Trans. Autom. Control*, 2012, 57(8), 2106–2110.

[34] C. Hu, H. He, and H. Jiang, Fixed/preassigned-time synchronization of complex networks via improving fixed-time stability, *IEEE Trans. Cybern*., 2021, 51(6), 2882–2892.

[35] G. H. Hardy, J. E. Littlewood, and G. Polya, *Inequalities*. Cambridge, UK: Cambridge University Press, 1952.

[36] C. Godsil and G. Royle, *Algebraic Graph Theory*. New York, NY, USA: Springer, 2001.

[37] F. Facchinei and J. S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problems*. New York, NY, USA: Springer, 2003.

[38] J. J. Hopfield and D. W. Tank, "Neural" computation of decisions in optimization problems, *Biol. Cybern*., 1985, 52(3), 141–152.

[39] H. Che and J. Wang, A collaborative neurodynamic approach to global and combinatorial optimization, *Neural Netw*., 2019, 114, 15–27.

[40] N. Sinha, R. Chakrabarti, and P. K. Chattopadhyay, Evolutionary programming techniques for economic load dispatch, *IEEE Trans. Evol. Comput*., 2003, 7(1), 83–94.

[41] C. Li, X. Yu, W. Yu, T. Huang, and Z. Liu, Distributed event-triggered scheme for economic dispatch in smart grids, *IEEE Trans. Ind. Inform*., 2016, 12(5), 1775–1785.

**Yiyang Ge** received the BS degree from Department of Mathematics, Lvliang College, China, in 2018, and the MS degree in applied mathematics from College of Mathematics and System Science, Xinjiang University, China, in 2022. She is currently pursuing the PhD degree at College of Information Science and Engineering, Northeastern University, Shenyang, China. Her research interests include distributed optimization and reinforcement learning.

**Zhanshan Wang** received the MS degree in control theory and control engineering from Liaoning Shihua University, Fushun, China, in 2001, and the PhD degree in control theory and control engineering from Northeastern University, Shenyang, China, in 2006. He is currently a professor at Northeastern University, China. His research interests include the stability analysis of recurrent neural networks, fault diagnosis, fault tolerant control, intelligent automation, and their applications in power systems and smart grid.

**Bibo Zheng** received the MS degree in applied mathematics from College of Mathematic and System Science, Xinjiang University, China, in 2020. She is currently pursuing the PhD degree at College of Information Science and Engineering, Northeastern University, China. Her research interests include fractional-order systems and network control.