Finite-time recurrent neural networks for solving nonlinear optimization problems and their application

Peng Miao^{1,2}, Yanjun Shen^{1,*}, Yujiao Li³, Lei Bao¹

Abstract

This paper focus on finite-time recurrent neural networks with continuous but non-smooth activation function solving nonlinearly constrained optimization problems. Firstly, definition of finite-time stability and finite-time convergence criteria are reviewed. Secondly, a finite-time recurrent neural network is proposed to solve the nonlinear optimization problem. It is shown that the proposed recurrent neural network is globally finite-time stable under the condition that the Hessian matrix of the associated Lagrangian function is positive definite. Its output converges to a minimum solution globally and finite-time, which means that the actual minimum solution can be derived in finite-time period. In addition, our recurrent neural network is applied to a hydrothermal scheduling problem. Compared with other methods, a lower consumption scheme can be derived in finite-time interval. At last, numerical simulations demonstrate the superiority and effectiveness of our proposed neural networks by solving nonlinear optimization problems with inequality constraints.

Keywords: finite-time stable, recurrent neural network, nonlinear optimization problems, continuous but non-smooth, hydrothermal scheduling problem

1. Introduction

Nonlinear constrained optimization problems are widely applied in the fields of science and engineering [1], for example, electrical networks planning, optimal control, mechanical design, hydrothermal scheduling, structure design and so on. In the past decades, numerical methods have been proposed to solve nonlinear convex optimization problems and approximate optimal solutions could be derived [2, 3, 4]. However, it is usual imperative to require real-time solutions in many applications. Because of parallel-distributed character and hardware-realization amenity, recurrent neural networks are capable of real-time computing and have been applied in solving nonlinear optimization problems with inequality constraints.

Different from traditional numerical optimization algorithms, recurrent neural networks governed by dynamic equations, can apply directly the techniques of the numerical ODE and the dynamic to solve constrained optimization problems effectively. In what follows, we give a detailed introduction for the development of recurrent neural networks. In 1986, Hopfield firstly presented a neural network to solve linear programming problems [5]. In order to solve nonlinear convex programming problems, Kennedy proposed a neural network with a finite penalty parameter [6]. Although the Kuhn-Tucker optimal conditions

hold for Kennedy's and Hopfield's, this neural network with a finite penalty parameter can not find the exact optimal solution. In addition, it is difficult to implement the neural network with a very large penalty parameter [7]. In order to overcome the imperfection, some scholars attempted to design neural networks without penalty parameter. For example, Rodriguez designed a switched-capacitor neural network to solve a class of optimization problems with nonlinear convex constraints [8]. For convex quadratic optimization problems with bounded constraints, Bouzerdoum presented a recurrent neural network to solve them [9]. Zhang proposed a Lagrange neural network to deal with nonlinear programming problems with equality constraints [10]. Especially, a projection neural network is developed to solve monotone variational inequality problems with limit constraints [11, 12]. By extending the projection neural network, two recurrent neural networks for solving strictly convex programming with nonlinear inequality constraints and linear constraints were presented in [13, 14], respectively. The projection neural network was also used to solve pseudomonotone variational inequality problems in [15]. The above recurrent neural networks are proposed to solve linear or nonlinear convex problems. But, many optimization problems may be nonconvex or nonsmooth in the engineering application fields, so it is interesting to study recurrent neural networks to solve nonconvex or nonsmooth optimization problems. Recently, some neural network models dealing with these optimization problems are proposed in [16, 17, 18, 19, 20, 21].

It should be noted that all the above mentioned neural networks are asymptotically stable or exponentially stable, which means that the exact solutions can be obtained when time goes to infinity. In order to accelerate the convergent speed, some

*Corresponding author

Email address: shenyj@ctgu.edu.cn (Yanjun Shen)

¹Hubei Provincial Collaborative Innovation Center for New Energy Micro-grid, China Three Gorges University, 443002, China

²Department of Basic Courses, Zhengzhou Science and Technology University, Zhengzhou 450064, Hennan, China. Email: miaopeng881026@163.com

³College of science, China Three Gorges University, 443002, China

neural networks with finite-time convergent properties are explored. For example, Cheng et. al. presented a recurrent neural network with a specific nonlinear unit to deal with convex optimization problems [22]. The proposed neural network converges to the optimal solution in a finite-time interval. The computation efficiency is increased dramatically as well. A finite-time convergent neural network has also been proposed to solve quadratic programming problems [23]. The actual optimal solutions can be derived in finite time interval. However, the neural networks proposed in [23] only deal with convex optimization problems. In this paper, in order to overcome these problems, we design a finite-time recurrent neural network based on the authors' previous work [24, 25, 26], to solve nonlinear nonconvex optimization problems. Under the condition that the Hessian matrix of the associated Lagrangian function is positive definite, we prove that the proposed recurrent neural network is globally finite-time stable. Then, the actual minimum solution can be derived in finite-time period. In addition, our recurrent neural network is applied to a hydrothermal scheduling problem. It is shown the superiority and effectiveness of our proposed neural networks by solving nonlinear optimization problems with inequality constraints.

This paper is organized as follows. In Section 2, definition of finite-time stability and finite-time convergence criteria are reviewed. In Section 3, we present a finite-time recurrent neural network to solve nonlinearly constrained optimization problems. In Section 4, our method is applied to solve a short-time hydrothermal scheduling problem. Numerical simulations and a test hydrothermal system are given to show the effectiveness of our methods in Section 5. Section 6 concludes the paper.

2. Preliminaries

For the system:

$$\dot{x}(t) = f(x(t)), f(0) = 0, x \in R^n, x(0) = x_0, \quad (1)$$

where $f : D \rightarrow R^n$ denotes a continuous function defined on an open neighborhood D such that $0 \in D$, we give the following definition of finite-time stability and finite-time stable criterion.

Definition 1 ([27, 28]). *If there exists an open neighborhood U such that $0 \in U$, and a function $T_x : U \setminus 0 \rightarrow (0, \infty)$, such that every solution $x(t, x_0)$ of (1) with the initial point $x_0 \in U \setminus 0$ is well-defined and unique in forward time for $t \in [0, T_x(x_0))$, $\lim_{t \rightarrow T_x(x_0)} x(t, x_0) = 0$, and $x(t, x_0) = 0$ for $t \geq T_x(x_0)$, then the system (1) finite-time converges to the equilibrium $x = 0$, $T_x(x_0)$ denotes the convergent time. The equilibrium of (1) is finite-time stable if it is Lyapunov stable and finite-time convergent. In addition, the origin is a globally finite-time stable equilibrium when $U = D = R^n$.*

Lemma 1 ([27]). *If there exists a positive definite function $V(x)$ defined on a neighborhood $U \subset R^n$ of the origin, two real numbers $k > 0$ and $0 < r < 1$ satisfying*

$$\dot{V}(x) \leq -kV(x)^r, \forall x \in U, \quad (2)$$

then, the origin of system (1) is finite-time stable. Moreover, the upper bound of the convergence time T_1 satisfies

$$T_1 \leq \frac{V(x_0)^{1-r}}{k(1-r)}, \forall x_0 \in U. \quad (3)$$

If $U = R^n$ and $V(x)$ is radially unbounded, the origin of system (1) is globally finite-time stable.

The following lemma will be used in the proof of our main results.

Lemma 2 ([29]). *Let $f : R^n \rightarrow R$ be continuously differentiable. Then $\max\{0, f(x)\}$ is a regular function, its Clarke's generalized gradient is given as follows,*

$$\partial \max\{0, f(x)\} = \begin{cases} \nabla f(x), & \text{if } f(x) > 0, \\ [0, 1]\nabla f(x), & \text{if } f(x) = 0, \\ 0, & \text{if } f(x) < 0. \end{cases} \quad (4)$$

3. Finite-time recurrent neural network

In this section, we consider the following nonlinear optimization problem (NOP):

$$\text{minimize } f(x), \quad (5a)$$

$$\text{subject to } c(x) \leq 0, x \geq 0, \quad (5b)$$

where $f : R^n \rightarrow R$, $c(x) = [c_1(x), \dots, c_m(x)]^T$, $x \in R^n$. In this paper, we assume that f and c_i are twice differentiable, and there exists at least a local optimal solution to the NOP.

It is clear that the NOP is a convex programming problem (CPP) if $f(x)$ and $c_i(x)$ are all convex, otherwise, the NOP is a nonconvex programming problem (NCP). From [9], we have that if x^* is a local optimal solution to the NOP, then there exists $y^* \in R^m$ such that (x^*, y^*) is a Karush-Kuhn-Tucker (KKT) point satisfying the following conditions

$$\begin{cases} y \geq 0, c(x) \leq 0, x \geq 0, \\ \nabla f(x) + \nabla c(x)y \geq 0, y^T c(x) = 0, \end{cases} \quad (6)$$

where $\nabla c(x)$ and $\nabla f(x)$ are the gradient of x . By the projection theorem [30], the following projection equations can be used to replace (6),

$$\begin{cases} (x - (\nabla f(x) + \nabla c(x)y))^+ - x = 0, \\ (y + c(x))^+ - y = 0, \end{cases} \quad (7)$$

where $x^+ = \max\{0, x\}$.

Let $z = (x, y)^T$, $z^+ = (x^+, y^+)^T$ and $\tilde{z} = (\tilde{x}, \tilde{y})^T = ((\nabla f(x^+) + \nabla c(x^+)y^+), -c(x^+))^T$. If and only if $z^* = (x^*, y^*)$ satisfies (7), then z^* is a KKT point of the NOP. Furthermore, z^* is a KKT point if z^* is also a solution to the variational inequality problem:

$$(z - z^*)F(z^*) \geq 0, \forall z \geq 0. \quad (8)$$

where $F(z^*) = [\nabla f(x^*) + \nabla c(x^*)y^*; -c(x^*)]$.

Next, we design the following finite-time recurrent neural network to solve (5) by extending the ODE method in [31],

$$\text{state equation } \dot{z}(t) = -\varepsilon \mathcal{F}(z - z^+ + \tilde{z}), \quad (9a)$$

$$\text{output equation } x(t) = x^+, \quad (9b)$$

where $\varepsilon > 0$ is a parameter and the activation function $\mathcal{F}(x)$ is defined as,

$$\mathcal{F}(x) = \text{sign}(x)|x|^r, \quad (10)$$

where $0 < r < 1$ is a real number, for $x = (x_1, x_2, \dots, x_n)^T$, $\mathcal{F}(x) = (\mathcal{F}(x_1), \mathcal{F}(x_2), \dots, \mathcal{F}(x_n))^T$ and $\text{sign}(x) = (\text{sign}(x_1), \text{sign}(x_2), \dots, \text{sign}(x_n))^T$.

We can use a recurrent neural network with a one-layer structure to realize the dynamical state equation expressed by (9). The circuit used to realize the neural network has $n + m$ integrators processors, $n + m$ continuous but nonsmooth activation functions processors for z , $n + m$ processors for z^+ and $n + m$ processors for $\tilde{z}$. Its complexity is dependent on the mapping z in the original problem. Fig. 1 shows the structure of the recurrent neural network (9).

Figure 1: The structure of the recurrent neural network (9).

The following lemmas are also useful for our main results.

Lemma 3. *There exists a solution to the recurrent neural network (9) on $[0, +\infty]$.*

Proof: Note that the functions (10) and $z - z^+ + \tilde{z}$ are continuous on $[0, +\infty]$. Then, $\mathcal{F}(z - z^+ + \tilde{z})$ is a continuous function on $[0, +\infty]$. By [32], there exists a solution to the recurrent neural network (9) on $[0, +\infty]$.

In what follows, let $y_0(t)$ denote the second state trajectory of (9) with zero initial point, and $S_0 = \{y \in \mathbb{R}^m | y = (y_0(t) + e^{t_0-t} y(t_0)) y(t_0)^+, t \in [t_0, \infty)\}$ denote an associated set, where $y(t_0) \in \mathbb{R}^m$ is a nonzero vector. A Lagrangian function concerned with the NOP is expressed by

$$L(x, y) = f(x) + y^T c(x).$$

Now, we give our main results.

Theorem 1. *If the Hessian matrix $\nabla_x^2 L(x, y)$ on $\mathbb{R}_+^n \times S_0$ is positive definite for any initial point $z_0 = (x_0, y_0)$, that is there exists a constant $\delta > 0$ such that $\nabla_x^2 L(x, y) > \delta I$, then, the recurrent neural network (9) is Lyapunov stable at a KKT point, it*

can converge to its equilibrium point finite time and the upper bound of the convergence time t_1 is given as follows

$$t_1 \leq \frac{V(0)^{\frac{1-r}{1+r}}}{K(1-r)(1+r)^{\frac{1-r}{1+r}}}. \quad (11)$$

where $V(0) = V_1(0) + V_2(0) = \frac{1}{r+1} \|x_0 - x_0^+ + \tilde{x}_0\|_{r+1}^{r+1} + \frac{1}{r+1} \|y_0 - y_0^+ - c(x_0^+)\|_{r+1}^{r+1}$ is the initial value of $V(t)$ defined by (12), $K = \min(\delta\varepsilon, \lambda\varepsilon)$, $0 \leq \lambda \leq 1$, $0 < r < 1$, and ε is a positive real number.

Proof: Defined the following Lyapunov function:

$$V(t) = V_1(t) + V_2(t), \quad (12)$$

where

$$V_1(t) = \frac{\|x - x^+ + \tilde{x}\|_{r+1}^{r+1}}{r+1},$$

$$V_2(t) = \frac{\|y - y^+ - c(x^+)\|_{r+1}^{r+1}}{r+1},$$

$\|x\|_{r+1}$ denotes the $r+1$ norm of a vector x , that is, $\|x\|_{r+1} = (\sum_{i=1}^n |x_i|^{r+1})^{\frac{1}{r+1}}$ for $x = [x_1, x_2, \dots, x_n]^T$. We calculate the time derivative of $V(t)$ along the trajectories of the system (9), then

$$\begin{aligned} \dot{V}(t) &= \dot{V}_1(t) + \dot{V}_2(t) \\ &= \left(\frac{dx}{dt}\right)^T \left(\frac{dV_1(t)}{dx}\right) + \left(\frac{dx}{dt}\right)^T \left(\frac{dV_2(t)}{dx}\right). \end{aligned} \quad (13)$$

Note that,

$$\begin{aligned} \dot{V}_1(t) &= \left(\frac{dx}{dt}\right)^T \left(\frac{dV_1(t)}{dx}\right) \\ &= \left(\frac{dx}{dt}\right)^T \frac{d(x - x^+ + \tilde{x})}{dx} \text{sign}(x - x^+ + \tilde{x}) |x - x^+ + \tilde{x}|^r. \end{aligned} \quad (14)$$

By Lemma 2, we have

$$\frac{dx_i^+}{dx_i} = \begin{cases} 1, & \text{if } x_i > 0, \\ [0, 1], & \text{if } x_i = 0, \\ 0, & \text{if } x_i < 0. \end{cases} \quad (15)$$

Therefore,

$$\begin{aligned} \frac{d(x - x^+ + \tilde{x})}{dx} &= I - \frac{dx^+}{dx} + \frac{d\tilde{x}}{dx} \\ &\geq \frac{d\tilde{x}}{dx} = \nabla_x^2 L(x, y). \end{aligned} \quad (16)$$

From (14), (15) and (16), we have

$$\begin{aligned} \dot{V}_1(t) &\leq \dot{x}^T \nabla_x^2 L(x, y) \text{sign}(x - x^+ + \tilde{x}) |x - x^+ + \tilde{x}|^r \\ &= -\varepsilon \mathcal{F}(x - x^+ + \tilde{x})^T \nabla_x^2 L(x, y) \text{sign}(x - x^+ + \tilde{x}) |x - x^+ + \tilde{x}|^r \\ &\leq -\varepsilon \delta (\text{sign}(x - x^+ + \tilde{x}) |x - x^+ + \tilde{x}|^r)^2 \\ &= -\delta \varepsilon \|x - x^+ + \tilde{x}\|_{2r}^{2r} \\ &\leq -\delta \varepsilon (r+1)^{\frac{2r}{r+1}} V_1(t)^{\frac{2r}{r+1}}. \end{aligned}$$

From (9), it also follows that

$$\begin{aligned}
\dot{V}_2(t) &= \frac{dy^T}{dt} \frac{d(y - y^+ - c(x^+))}{dy} \text{sign}(y - y^+ - c(x^+))|y - y^+ - c(x^+)|^r \\
&= \lambda \dot{y}^T \text{sign}(y - y^+ - c(x^+))|y - y^+ - c(x^+)|^r \\
&= -\lambda \mathcal{E} \mathcal{F}(y - y^+ - c(x^+))^T \text{sign}(y - y^+ - c(x^+))|y - y^+ - c(x^+)|^r \\
&= -\lambda \mathcal{E} \left(\text{sign}(y - y^+ - c(x^+))|y - y^+ - c(x^+)|^r \right)^2 \\
&= -\lambda \mathcal{E} \|y - y^+ - c(x^+)\|_{2r}^{2r} \\
&\leq -\lambda \mathcal{E} (r+1) \frac{2r}{r+1} V_2(t)^{\frac{2r}{r+1}}.
\end{aligned}$$

Let $K = \min\{\delta\mathcal{E}, \lambda\mathcal{E}\}$, then

$$\begin{aligned}
\dot{V}(t) &= \dot{V}_1(t) + \dot{V}_2(t) \\
&\leq -K(r+1) \frac{2r}{r+1} [V_1(t)^{\frac{2r}{r+1}} + V_2(t)^{\frac{2r}{r+1}}] \\
&\leq -K(r+1) \frac{2r}{r+1} [V_1(t) + V_2(t)]^{\frac{2r}{r+1}} \\
&\leq -K(r+1) \frac{2r}{r+1} V(t)^{\frac{2r}{r+1}}.
\end{aligned} \tag{17}$$

By Lemma 1, the recurrent neural network (9) is finite time stable and the upper bound of the convergence time is given by (11). The proof is completed.

Theorem 2. $x(t)$ is the optimal solution to the nonlinear optimization problem (5), if z^* is an equilibrium point of (9).

Proof: Since z^* is an equilibrium point of (9), z^* satisfies

$$-\mathcal{E} \mathcal{F}(z^* - (z^*)^+ + \tilde{z}^*) = 0.$$

Then,

$$z^* - (z^*)^+ + \tilde{z}^* = 0,$$

and

$$\begin{cases} y^* \geq 0, c(x^*) \leq 0, x^* \geq 0, \\ \nabla f(x^*) + \nabla c(x^*)y^* \geq 0, (y^*)^T c(x^*) = 0. \end{cases}$$

Therefore,

$$\begin{cases} (x^* - (\nabla f(x^*) + \nabla c(x^*)y^*))^+ - x^* = 0, \\ (y^* + c(x^*))^+ - y^* = 0. \end{cases}$$

Thus, z^* is a KKT point of the NOP (5). Therefore, x^* is a local optimal solution to the NOP [1].

Note that the set $S = \{z \in R^n | V(z) \leq V(z_0)\}$ is bounded and the solution trajectory $\{z(t)\} \subset S$ is also bounded. By LaSalle's invariance principle, $z(t)$ can convergence to Θ , the largest invariant subset of set:

$$\Theta = \left\{ \hat{z} \in S \mid \frac{dV}{dt} = 0 \right\}.$$

That is to say, $\lim_{t \rightarrow \infty} \text{dist}(z(t), \Theta) = 0$. Thus,

$$\lim_{t \rightarrow \infty} \text{dist}(x^+, X^*) = 0,$$

where $X^* = \{x^*\}$. Therefore, the output trajectory of the proposed neural network globally converges to the set of the minimizers of the NOP. Then, the proof is completed.

Remark 1. Compared with other methods, the proposed neural network with any initial point z_0 is finite-time stable at a KKT point. Its output can globally converge to the optimal solution to the NOP (5) in finite time interval, if $\nabla_x^2 L(x, y) > \delta I$ on $R_+^n \times S_0$ for a positive real number δ . Moreover, from the inequality (11), we have the conclusions that the convergent time is dependent on the parameters r and ε .

Remark 2. There exist several other neural network models which can be used to solve the problem (5) [6, 8, 11, 33, 4, 16]. In Table 1, we summarize the comparisons of these neural network models with our model (9).

Table 1: Comparisons of (9) with other neural network models by solving the problem (5)

Network	Error	Time	CPP	N CPP
[6]	non-zero	infinite	√	×
[8]	zero	infinite	√	×
[11]	zero	infinite	√	×
[33]	zero	infinite	√	×
[4]	zero	infinite	√	×
[16]	zero	infinite	√	√
(9)	zero	finite	√	√

Remark 3. In [19, 20, 21], neural networks have been presented to solve nonsmooth optimization problem. In this paper, the objective function is assumed to be twice differentiable. Therefore, it is intractable to extend the proposed neural networks to solve these problems. Future works should investigate the design of neural networks with finite time stability solving these nonsmooth problems.

4. Application to Hydrothermal Scheduling Problem

In this section, our recurrent neural networks will be applied to a classic hydrothermal scheduling problem. The objective of the short-term hydrothermal scheduling is to find optimal water volume of reservoir and water discharge of hydro plant in a given time interval such that the total cost of thermal generation is optimal by making full use of the hydro resource. At the same time, diverse hydraulic, thermal constraints are satisfied. In order to better describe the problem, a test system is used. We set up a multi-reservoir cascaded hydraulic power system. The hydropower generation is the function of water discharge rate and storage volume. Water transportation delay of reservoirs is also taken into account. In a hydrothermal power system the marginal cost of hydroelectric generation is negligible. The short-term scheduler then allocates water for hydrogeneration among various time intervals along the scheduling horizon so as to minimise the cost of fuel for thermal generation while trying to satisfy diverse constraints. The scheduling for time interval is one hour. A general system representation for formation of i th reservoir is shown in Fig. 2. Especially, the classic model of hydrothermal scheduling problem coming from [34], can be written as the following nonlinear optimal problem:

$$\begin{aligned} \text{minimize } f(x) = & \sum_{t=1}^{T_h} [\alpha_i + \beta_i (P_D^t - \sum_{i=1}^N P_{hi}^t) \\ & + \gamma_i (P_D^t - \sum_{i=1}^N P_{hi}^t)^2], \end{aligned} \quad (18a)$$

$$\text{subject to } P_s^{\min} \leq P_D^t - \sum_{i=1}^N P_{hi}^t \leq P_s^{\max}, \quad (18b)$$

$$P_{hi}^{\min} \leq P_{hi}^t \leq P_{hi}^{\max}, \quad (18c)$$

$$Q_i^{\min} \leq Q_i^t \leq Q_i^{\max}, \quad (18d)$$

$$V_i^{\min} \leq V_i^t \leq V_i^{\max}, \quad (18e)$$

$$V_i^t = V_i^{t-1} + M[I_i^t - Q_i^t - S_i^t + \sum_{m=i}^{N_u} (Q_m^{t-\tau_{m,i}} + S_m^{t-\tau_{m,i}})], \quad (18f)$$

$$V_i^0 = V_i^B, V_i^T = V_i^E, \quad (18g)$$

where $f(x)$ denotes the fuel total cost of thermal plant, α_i, β_i and γ_i are coefficients of the thermal generation of i th plant, P_{si}^t and P_{hi}^t are the power generation of thermal plant i and the power generation of hydro plant i at time interval t , respectively, P_D^t is the system load balance, V_i^t, Q_i^t are denoted as water volume of reservoir i at the end of time interval t and water discharge of hydro plant i at time interval t , respectively, the water spillage of hydro plant i at time interval t is S_i^t , the natural inflow into reservoir i at time interval t is I_i^t , N_u is number of upstream hydropower plants directly above i th hydro plant, τ_i^m is water transport delay time from reservoir m to i , M is conversion factor of water discharge into stored water. For this problem, let N_s, N_h denote number of thermal plants, number of hydro plants, respectively; T_h and t are denoted as total time horizon and time index, respectively.

Note that (18b) - (18g) can also be expressed as follows,

$$P_s^{\min} - (P_D^t - \sum_{i=1}^N P_{hi}^t) \leq 0, P_D^t - \sum_{i=1}^N P_{hi}^t - P_s^{\max} \leq 0,$$

$$P_{hi}^{\min} - P_{hi}^t \leq 0, P_{hi}^t - P_{hi}^{\max} \leq 0,$$

$$Q_i^{\min} - Q_i^t \leq 0, Q_i^t - Q_i^{\max} \leq 0,$$

$$V_i^{\min} - V_i^t \leq 0, V_i^t - V_i^{\max} \leq 0,$$

$$V_i^t - (V_i^{t-1} + M[I_i^t - Q_i^t - S_i^t + \sum_{m=i}^{N_u} (Q_m^{t-\tau_{m,i}} + S_m^{t-\tau_{m,i}})]) \leq 0,$$

$$V_i^{t-1} + M[I_i^t - Q_i^t - S_i^t + \sum_{m=i}^{N_u} (Q_m^{t-\tau_{m,i}} + S_m^{t-\tau_{m,i}})] - V_i^t \leq 0.$$

Then, $c(x)$ defined by (19) is given as

$$c(x) = [P_s^{\min} - (P_D^1 - \sum_{i=1}^{N_h} P_{hi}^1); \dots; P_s^{\min} - (P_D^{T_h} - \sum_{i=1}^{N_h} P_{hi}^{T_h}); P_D^1 -$$

$$\begin{aligned} & \sum_{i=1}^N P_{hi}^1 - P_s^{\max}; \dots; P_D^{T_h} - \sum_{i=1}^N P_{hi}^{T_h} - P_s^{\max}; P_{h1}^{\min} - P_{h1}^1; \dots; P_{hN_h}^{\min} \\ & - P_{hN_h}^{T_h}; P_{h1}^1 - P_{h1}^{\max}; \dots; P_{hN_h}^{T_h} - P_{hN_h}^{\max}; Q_1^{\min} - Q_1^1; \dots; Q_{N_h}^{\min} - Q_{N_h}^{T_h}; \\ & Q_1^1 - Q_1^{\max}; \dots; Q_{N_h}^{T_h} - Q_{N_h}^{\max}; V_1^{\min} - V_1^1; \dots; V_{N_h}^{\min} - V_{N_h}^{T_h}; V_1^1 - V_1^{\max}; \\ & \dots; V_{N_h}^{T_h} - V_{N_h}^{\max}; V_1^1 - V_1^0 + M[I_1^1 - Q_1^1 - S_1^1 + \sum_{m=1}^{N_u} (Q_m^{1-\tau_{m,1}} + \\ & S_m^{1-\tau_{m,1}})]; \dots; V_{N_h}^{T_h} - V_{N_h}^{T_h-1} + M[I_{N_h}^{T_h} - Q_{N_h}^{T_h} - S_{N_h}^{T_h} + \sum_{m=N_h}^{N_u} (Q_m^{T_h-\tau_{m,N_h}} \\ & + S_m^{T_h-\tau_{m,N_h}})]; -V_1^1 + V_1^0 + M[I_1^1 - Q_1^1 - S_1^1 + \sum_{m=1}^{N_u} (Q_m^{1-\tau_{m,1}} + S_m^{1-\tau_{m,1}})]; \\ & \dots; -V_{N_h}^{T_h} + V_{N_h}^{T_h-1} + M[I_{N_h}^{T_h} - Q_{N_h}^{T_h} - S_{N_h}^{T_h} + \sum_{m=N_h}^{N_u} (Q_m^{T_h-\tau_{m,N_h}} \\ & + S_m^{T_h-\tau_{m,N_h}})]]. \end{aligned} \quad (19)$$

In addition, we have

$$P_{hi}^t = C_{1i}(V_i^t)^2 + C_{2i}(Q_i^t)^2 + C_{3i}V_i^tQ_i^t + C_{4i}V_i^t + C_{5i}Q_i^t + C_{6i},$$

where $C_{1i}, C_{2i}, C_{3i}, C_{4i}, C_{5i}$ and C_{6i} are the hydro plant power generation coefficients [34]. Then, the optimization problem (18) is a nonlinear function of V_i^t and Q_i^t .

Figure 2: System Representation

We can use the recurrent neural network proposed in Section 3 to solve the optimization problem (18).

$$a_{ii} = \left\{ \begin{array}{l} 2\gamma(2C_{11}V_1^i + C_{31}Q_1^i + C_{41})^2 - 2C_{11}[\beta + \\ 2\gamma(P_D^i - \sum_{k=1}^4 P_{h_k}^i)], \\ i = 1, 2, \dots, 24 \\ 2\gamma(2C_{12}V_2^{i-24} + C_{32}Q_2^{i-24} + C_{42})^2 - 2C_{12}[\beta + \\ 2\gamma(P_D^{i-24} - \sum_{k=1}^4 P_{h_k}^{i-24})], \\ i = 25, 26, \dots, 48 \\ 2\gamma(2C_{13}V_3^{i-48} + C_{33}Q_3^{i-48} + C_{43})^2 - 2C_{13}[\beta + \\ 2\gamma(P_D^{i-48} - \sum_{k=1}^4 P_{h_k}^{i-48})], \\ i = 49, 50, \dots, 72 \\ 2\gamma(2C_{14}V_4^{i-72} + C_{34}Q_4^{i-72} + C_{44})^2 - 2C_{14}[\beta + \\ 2\gamma(P_D^{i-72} - \sum_{k=1}^4 P_{h_k}^{i-72})], \\ i = 73, 74, \dots, 96 \\ \dots \\ 2\gamma(2C_{1N_h}V_4^{i-23N_h} + C_{3N_h}Q_4^{i-23N_h} + C_{4N_h})^2 - \\ 2C_{1N_h}[\beta + 2\gamma(P_D^{i-23N_h} - \sum_{k=1}^4 P_{h_k}^{i-23N_h})], \\ i = 23N_h + 1, \dots, 24N_h \end{array} \right. \quad (20)$$

$$b_{ii} = \left\{ \begin{array}{l} 2\gamma(2C_{21}Q_1^i + C_{31}V_1^i + C_{51})^2 - 2C_{11}[\beta + \\ 2\gamma(P_D^i - \sum_{k=1}^4 P_{h_k}^i)], \\ i = 1, 2, \dots, 24 \\ 2\gamma(2C_{22}Q_2^{i-24} + C_{32}V_2^{i-24} + C_{52})^2 - 2C_{12}[\beta + \\ 2\gamma(P_D^{i-24} - \sum_{k=1}^4 P_{h_k}^{i-24})], \\ i = 25, 26, \dots, 48 \\ 2\gamma(2C_{23}Q_3^{i-48} + C_{33}V_3^{i-48} + C_{53})^2 - 2C_{13}[\beta + \\ 2\gamma(P_D^{i-48} - \sum_{k=1}^4 P_{h_k}^{i-48})], \\ i = 49, 50, \dots, 72 \\ 2\gamma(2C_{24}Q_4^{i-72} + C_{34}V_4^{i-72} + C_{54})^2 - 2C_{14}[\beta + \\ 2\gamma(P_D^{i-72} - \sum_{k=1}^4 P_{h_k}^{i-72})], \\ i = 73, 74, \dots, 96 \\ \dots \\ 2\gamma(2C_{2N_h}Q_4^{i-23N_h} + C_{3N_h}Q_4^{i-23N_h} + C_{5N_h})^2 - \\ 2C_{1N_h}[\beta + 2\gamma(P_D^{i-23N_h} - \sum_{k=1}^4 P_{h_k}^{i-23N_h})], \\ i = 23N_h + 1, \dots, 24N_h \end{array} \right. \quad (21)$$

$$d_{ii} = \left\{ \begin{array}{l} 2\gamma[(2C_{11}V_1^i + C_{31}Q_1^i + C_{41})(2C_{21}Q_1^i + C_{31}V_1^i + \\ C_{51}) - (P_D^i - \sum_{k=1}^4 P_{h_k}^i)] - C_{31}\beta, \\ i = 1, 2, \dots, 24 \\ 2\gamma[(2C_{12}V_2^{i-24} + C_{32}Q_2^{i-24} + C_{42})(2C_{22}Q_1^{i-24} + \\ C_{32}V_1^{i-24} + C_{52}) - (P_D^{i-24} - \sum_{k=1}^4 P_{h_k}^{i-24})] - C_{32}\beta, \\ i = 25, 26, \dots, 48 \\ 2\gamma[(2C_{13}V_3^{i-48} + C_{33}Q_3^{i-48} + C_{43})(2C_{23}Q_1^{i-48} + \\ C_{33}V_1^{i-48} + C_{53}) - (P_D^{i-48} - \sum_{k=1}^4 P_{h_k}^{i-48})] - C_{33}\beta, \\ i = 49, 50, \dots, 72 \\ 2\gamma[(2C_{14}V_4^{i-72} + C_{34}Q_4^{i-72} + C_{44})(2C_{24}Q_1^{i-72} + \\ C_{34}V_1^{i-72} + C_{54}) - (P_D^{i-72} - \sum_{k=1}^4 P_{h_k}^{i-72})] - C_{34}\beta, \\ i = 73, 74, \dots, 96 \\ \dots \\ 2\gamma[(2C_{1N_h}V_4^{i-23N_h} + C_{3N_h}Q_4^{i-23N_h} + C_{4N_h}) \times \\ (2C_{2N_h}Q_1^{i-23N_h} + C_{3N_h}V_1^{i-23N_h} + C_{5N_h}) \\ - (P_D^{i-23N_h} - \sum_{k=1}^4 P_{h_k}^{i-23N_h})] - C_{3N_h}\beta, \\ i = 23N_h + 1, 23N_h + 2, \dots, 24N_h \end{array} \right. \quad (22)$$

Theorem 3. The optimal solution to the hydrothermal scheduling problem (18) can be obtained in finite time by using the proposed neural network (9) with any initial point z_0 , if $\nabla_x^2 L(x, y)$ is positive definite, that is, there exists a constant $\delta > 0$ such that $\nabla_x^2 L(x, y) > \delta I$, where

$$\begin{aligned} \nabla_x^2 L(x, y) &= \nabla^2 f(x) + y^T \nabla^2 c(x) \\ &= \begin{pmatrix} A_1 & D_1 \\ D_1^T & B_1 \end{pmatrix} + \begin{pmatrix} A_2 & D_2 \\ D_2^T & B_2 \end{pmatrix} y, \end{aligned} \quad (23)$$

and $f(x)$ is given by (18a),

$$\begin{aligned} A_1 &= \text{diag}(a_{11}, a_{22}, \dots, a_{24N_h, 24N_h}), \\ B_1 &= \text{diag}(b_{11}, b_{22}, \dots, b_{24N_h, 24N_h}), \\ D_1 &= \text{diag}(d_{11}, d_{22}, \dots, d_{24N_h, 24N_h}), \\ A_2 &= 2\text{diag}(C_{11}I^{24 \times 24}, C_{12}I^{24 \times 24}, \dots, C_{1, N_h}I^{24 \times 24}), \\ B_2 &= 2\text{diag}(C_{21}I^{24 \times 24}, C_{22}I^{24 \times 24}, \dots, C_{2, N_h}I^{24 \times 24}), \\ D_2 &= \text{diag}(C_{31}I^{24 \times 24}, C_{32}I^{24 \times 24}, \dots, C_{3, N_h}I^{24 \times 24}), \end{aligned}$$

$c(x)$, a_{ii} , b_{ii} and d_{ii} are given by (19), (20), (21) and (22), respectively.

Proof: From (19), we have that $\nabla c(x) = \begin{pmatrix} A_2 & D_2 \\ D_2^T & B_2 \end{pmatrix} y$.

From (18a), it also follows that $\nabla f(x) = \begin{pmatrix} A_1 & D_1 \\ D_1^T & B_1 \end{pmatrix}$, where a_{ii} , b_{ii} and d_{ii} are given by (20), (21) and (22), respectively. Since $\nabla_x^2 L(x, y) > \delta I$, then the problem (18) can be solved by using the proposed neural network (9) and the optimal solution to (18) can be achieved in finite time by Theorem 1.

Remark 4. By theorem 3, our method can obtain a minimum solution to (18) in finite time compared with other methods which can obtain an approximate solutions. Consequently, the thermal plant cost obtained by the proposed neural network is smaller than by other methods.

5. Numerical Simulations

In this part, three illustrative examples and an engineering project for hydrothermal scheduling are used to show the effectiveness of our methods. The simulation is performed with the programming language Matlab 7.10.0 on a desktop computer with the Intel (R) Core(TM) G640 Duo CPU at 2.80 GHz, 1.59 GHz and 1.90 GB of RAM.

5.1. Illustrative examples

Example 1. A convex programming problem:

$$\text{minimize } f(x) = 2x_1^2 + x_2^2 + x_1x_2 - 12x_1 - 10x_2, \quad (24a)$$

$$\text{subject to } c_1(x) = x_1^2 + (x_2 - 5)^2 \leq 64, \quad (24b)$$

$$c_2(x) = (x_1 + 3)^2 + (x_2 - 1)^2 \leq 36, \quad (24c)$$

$$c_3(x) = (x_1 - 3)^2 + (x_2 - 1)^2 \leq 36, \quad (24d)$$

$$x_1, x_2 \geq 0.$$

It is clear that $f(x)$, $c_i(x)$ are all convex and the problem (24) has a unique solution $x^* = [2, 4]^T$. When the proposed network (9) is used to solve the problem (24), the output state trajectory of (9) always converges to x^* with any initial point. Let $\varepsilon = 50$, $r = 0.5$. Table 2 shows the simulation results by using different neural networks or numerical (Num) methods with different initial point. From the Table 2, it is observed that our proposed neural network not only converges to the exact solution in finite-time, but also has a smaller error norm. Furthermore, let $z_0 = 30 * \mathbf{1}^{5 \times 1}$, $K = 1$, we have $V(0) = V_1(0) + V_2(0) = 39.89 + 311.97 = 351.86$. By computing (11), we can obtain $t_1 \leq 1.233$. Fig. 3 gives the output state trajectory of (9) by solving Example 1 and the convergence time is about 0.108s.

Figure 3: The output state trajectory of (9) by solving Example 1 ($\varepsilon = 50$, $r = 0.5$, $z_0 = 30 * \mathbf{1}^{5 \times 1}$).

Example 2. A non-convex constraint programming problem:

$$\text{minimize } f(x) = x_1^2 + x_2^2, \quad (25a)$$

$$\text{subject to } c(x) = 1 - 0.3x_1x_2 \leq 0 \quad (25b)$$

where $2 \leq x_1 \leq 5$ and $1 \leq x_2 \leq 2$.

Figure 4: The trajectory of y in Example 2.

The objective function is convex but $c(x)$ is non-convex. $x^* = [2, 1.6666]^T$ is the optimal solution to this problem. Let $\varepsilon = 50$, $r = 0.5$, $z_0 = \mathbf{1}^{3 \times 1}$. From Fig. 4, we have that,

$$\nabla_x^2 L(x, y) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} + \begin{pmatrix} 0 & 0.3 \\ 0.3 & 0 \end{pmatrix} y > 0.$$

So, the proposed neural network (9) can be used to solve this problem by Theorem 1 and 2. Fig. 5 shows the simulation results and the convergence time 0.18s obtained by our method is smaller than that obtained by using the method in [16]. Since $c(x)$ is non-convex, the optimal solution x^* can not be obtained by the methods in [4, 12]. However, the neural network we presented can converge to the optimal solution in finite-time.

Figure 5: The error trajectory $\|x - x^*\|$ by solving Example 2 ($\varepsilon = 50$, $r = 0.5$, $z_0 = \mathbf{1}^{3 \times 1}$).

Example 3. A non-convex object function programming problem::

$$\text{minimize } f(x) = -x_1x_2, \quad (26a)$$

$$\text{subject to } x_1 + 4x_2 - 1 \leq 0 \quad (26b)$$

$$20(4x_1^2 + 5x_2^2 + 1.5x_1x_2) - 23 \leq 0 \quad (26c)$$

$$x \geq 0$$

For this problem, $f(x)$ is non-convex and the optimal solution is $x^* = [0.51741, 0.0702]^T$. We select $\varepsilon = 50$, $r = 0.5$, $z_0 = \mathbf{1}^{4 \times 1}$. From Fig. 6, we have that,

Table 2: Comparisons the performances of different methods with ours ($\varepsilon = 50, r = 0.5$) by solving problem (24)

Method	z_0	solution	time(s)	error
[4]	$30 * \mathbf{1}^{5 \times 1}$	(2.0191, 3.9538)	1.046	0.05
[12]	$30 * \mathbf{1}^{5 \times 1}$	(1.9294, 3.8999)	4.953	0.1225
[16]	$30 * \mathbf{1}^{5 \times 1}$	(2.0008, 4.0003)	0.172	8.54e-5
Ours	$30 * \mathbf{1}^{5 \times 1}$	(2.00000, 4.00000)	0.108	<1e-5
Num	$-30 * \mathbf{1}^{5 \times 1}$	(2.0187, 3.9537)	0.146	0.0499
[12]	$-30 * \mathbf{1}^{5 \times 1}$	(1.9982, 3.9993)	0.156	0.0019
[16]	$-30 * \mathbf{1}^{5 \times 1}$	(1.99997, 3.99999)	0.141	3.16e-5
Ours	$-30 * \mathbf{1}^{5 \times 1}$	(2.00000, 4.00000)	0.121	<1e-5
Num	$-300 * \mathbf{1}^{5 \times 1}$	(2.0187, 3.9537)	1.953	0.0499
[12]	$-300 * \mathbf{1}^{5 \times 1}$	(1.9994, 3.9994)	0.452	8.4e-4
[16]	$-300 * \mathbf{1}^{5 \times 1}$	(2.0000, 4.0003)	0.14	3e-4
Ours	$-300 * \mathbf{1}^{5 \times 1}$	(2.00000, 4.00000)	0.134	<1e-5

Figure 6: The trajectory of y in Example 3.

$$\nabla_x^2 L(x, y) = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} + 20 \begin{pmatrix} 8 & 1.5 \\ 1.5 & 10 \end{pmatrix} y > 0.$$

Therefore, our neural network (9) can be used to solve this problem (26) by Theorem 1 and 2. Fig. 7 shows the simulation results.

Figure 7: The error trajectory $\|x - x^*\|$ by solving Example 3 ($\varepsilon = 50, r = 0.5, z_0 = \mathbf{1}^{4 \times 1}$).

5.2. A classic hydrothermal scheduling in engineering project

In this section, our method will be applied to a classic hydrothermal scheduling problem. Its schematic diagram is shown in Fig. 8. There are four cascaded hydroplants along the river and an equivalent thermal plant. The scheduling period is one day with 1-h interval. We select the same parameters as [34],

Figure 8: The schematic diagram of a classic hydrothermal scheduling problem.

which are given in Table 3, 4 and 5. Another parameters are given as, $T_h = 24$, $N_h = 4$, $\alpha_i = 1000$, $\beta_i = 10$, $\gamma_i = 0.5$, $M = [10, 000, 8000, 1000, 0]^T m^3/h$, $\tau_{1,3} = 1h$, $\tau_{2,3} = 2h$ and $\tau_{3,4} = 2h$. In the following simulation, we set $\varepsilon = 50$, $r = 0.5$, $z_0 = \mathbf{1}^{384 \times 1}$. For this problem, we have $A_2 = -0.002 * \mathbf{I}^{96 \times 96}$, $B_2 = -0.2 * \mathbf{I}^{96 \times 96}$, $D_2 = 0.01 * \mathbf{I}^{96 \times 96}$ in (23), and the smallest eigenvalue of $-\begin{bmatrix} A_2 & D_2 \\ D_2^T & B_2 \end{bmatrix}$ is 0.001496. From Fig.9, we can obtain that,

$$\nabla_x^2 L(x, y) = \begin{pmatrix} A_1 & D_1 \\ D_1^T & B_1 \end{pmatrix} + \begin{pmatrix} A_2 & D_2 \\ D_2^T & B_2 \end{pmatrix} y > 0.$$

Then the proposed recurrent neural network (9) can be used to solve this problem (18) by Theorem 3.

Figure 9: The trajectory of y_{min} in the hydrothermal scheduling problem.

Table 3: Load demand(Unit: MW)

Time	1	2	3	4	5	6	7	8
Load	190	170	170	190	190	210	230	250
Time	9	10	11	12	13	14	15	16
Load	270	310	350	310	350	350	310	290
Time	17	18	19	20	21	22	23	24
Load	270	250	230	210	210	210	190	190

Figure 10: Hourly hydro plant storage.

From the simulation results, we can obtain the optimal V_i^l and Q_i^l , which are shown in Fig. 10 and Fig. 11, respectively. In addition, Table 6 shows that our method achieve a better performances concerned with the total thermal plant costs than others.

6. Conclusion

In the paper, finite-time recurrent neural networks were proposed and used to solve nonlinearly constrained optimization

Table 4: Hydro plant power generation coefficients

Plant i	C_{1i}	C_{2i}	C_{3i}	C_{4i}	C_{5i}	C_{6i}
1	-0.001	-0.1	0.01	0.40	4.0	-30
2	-0.001	-0.1	0.01	0.38	3.5	-30
3	-0.001	-0.1	0.01	0.30	3.0	-30
4	-0.001	-0.1	0.01	0.38	3.8	-30

Table 5: Characteristics of hydro plant

Plant i	$V_i^{\min}$	$V_i^{\max}$	V_i^1	V_i^{24}	$Q_i^{\min}$	$Q_i^{\max}$
1	80	150	100	120	5	15
2	60	120	80	70	6	15
3	100	240	170	170	10	30
4	70	160	120	120	13	25

Figure 11: Hourly hydro plant discharge.

problems. Firstly, definition of finite-time stability and finite-time convergence criteria were reviewed. Secondly, a finite-time recurrent neural network was proposed to solve the nonlinear optimization problem. It was shown that the proposed recurrent neural network was globally finite-time stable under the condition that the Hessian matrix of the associated Lagrangian function was positive definite. Its output could converge to a minimum solution globally and finite-time, which means that the actual minimum solution could be derived in finite-time period. In addition, our neural network was applied to a hydrothermal scheduling problem. Compared with other methods, a lower consumption scheme of hydrothermal scheduling was obtained in finite-time period. At last, numerical simulations demonstrated superiority and effectiveness of our proposed neural networks by solving nonlinear optimization problems with inequality constraints.

Recently, the researches on memristor-based recurrent neural networks and complex dynamical networks have attracted considerable attentions [37, 38, 39, 40]. It is worth studying these recurrent neural networks to solve nonlinear optimization problems.

Table 6: Comparison of thermal plant cost with other methods

Method	cost(\$)
Augmented Lagrange in [35]	154739.0
Two-phase neural network in [36]	154808.5
Chaotic hybrid differential evolution in [34]	154338.1
Our method (9)	152635.7

Acknowledgements

This work was supported by the National Science Foundation of China (No. 61374028, 51177088, 61273183), the Grant National Science Foundation of Hubei Provincial (2013CFA050), the Scientific Innovation Team Project of Hubei Provincial Department of Education (T201504).

References

- [1] M. S. Bazaraa, H. D. Sherali, C. M. Shetty (1993) Nonlinear Programming: Theory and Algorithms, 2nd ed. New York: Wiley.
- [2] F. Facchinei, A. Fischer, C. Kanzow (1999) A simply constrained optimization reformulation of KKT systems arising from variational inequalities, *Applied Mathematics and Optimization*, 40(1): 19-37.
- [3] A. Wachter and L. T. Biegler (2005), Line search filter methods for nonlinear programming: Motivation and global convergence, *SIAM Journal on Optimization*, 16: 1-31.
- [4] M. V. Solodov, P. Tseng (1996) Modified projection-type methods for monotone variational inequalities, *SIAM Journal on Control and Optimization*, 2: 1814-1830.
- [5] D. W. Tank, J. J. Hopfield (1986) Simple neural optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit, *IEEE Transactions Circuits System*, CAS-33 (5): 533-541.
- [6] M. P. Kennedy, L. O. Chua (1988) Neural networks for nonlinear programming, *IEEE Transactions Circuits System*, CAS-35 (5): 554-562.
- [7] W. E. Lillo, M. H. Loh, S. Hui, S. H. Zak (1993) On solving constrained optimization problems with neural networks: A penalty method approach, *IEEE Transactions on Neural Networks*, 4 (6): 931-939.
- [8] A. Rodriguez-Vazquez, R. Dominguez-Castro, A. Rueda, J. L. Huertas, E. Sanchez-Sinencio (1990) Nonlinear switched-capacitor neural networks for optimization problems, *IEEE Transactions Circuits System*, 37 (3): 384-397.
- [9] A. Bouzerdoum, T. R. Pattison (1993) Neural network for quadratic optimization with bound constraints, *IEEE Transactions Neural Networks*, 4 (2): 293-304.
- [10] S. Zhang, A. G. Constantinides (1992) Lagrange programming neural networks, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39 (7): 441-452.
- [11] Y. S. Xia, H. Leung, J. Wang (2002) A projection neural network and its application to constrained optimization problems, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 49 (4): 447-458.
- [12] Y. S. Xia (2004) An extended projection neural network for constrained optimization, *Neural Computing*, 16 (4): 863-883.
- [13] Y. S. Xia, J. Wang (2004) A recurrent neural network for solving nonlinear optimization subject to nonlinear inequality constraints, *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51 (7): 1385-1394.
- [14] Y. S. Xia, J. Wang (2005) A recurrent neural network for solving nonlinear convex programs subject to linear constraints, *IEEE Transactions on Neural Networks*, 16 (2): 379-386.
- [15] X. Hu, J. Wang (2006) Solving pseudomonotone variational inequalities and pseudoconvex optimization problems using the projection neural network, *IEEE Transactions on Neural Networks*, 6: 1487-1499.
- [16] Y. S. Xia, G. Feng, J. Wang (2008) A novel recurrent neural network for solving nonlinear optimization problems with inequality constraints, *IEEE Transactions on neural networks*, 19 (8): 1340-1353.
- [17] D. Beyer, R. Ogier (2000) Tabu learning: A neural network search method for solving nonconvex optimization problems, *IEEE International Joint Conference on Neural Networks*, 2: 953-961.
- [18] C. Y. Sun, C. B. Feng (2005) Neural networks for nonconvex nonlinear programming problems: A switching control approach, in *Lecture Notes in Computer Science*, 3496: 694-699.
- [19] L. Cheng, Z. G. Hou, Y. Lin, M. Tan, W. C. Zhang, F. X. Wu (2011) Recurrent neural network for non-smooth convex optimization problems with application to the identification of genetic regulatory networks, *IEEE Transactions on neural networks*, 22 (5): 714-726.
- [20] M. Forti, P. Nistri, M. Quincampoix (2006) Convergence of neural networks for programming problems via a nonsmooth Łojasiewicz inequality, *IEEE Transactions on neural networks*, 17 (6): 1471-1486.
- [21] W. Bian, X. Xue, Subgradient-based neural networks for nonsmooth non-convex optimization problems, *IEEE Transactions on neural networks*, 20(6): 1024-1038.
- [22] L. Cheng, Z. G. Hou, N. Homma, M. Tan and M. M. Gupta, Solving convex optimization problems using recurrent neural networks in finite time, *International Joint Conference on Neural Networks, IJCNN 2009*, Atlanta, Georgia, USA 2009 539-543.
- [23] S. Li, Y. M. Li and Z. Wang (2013) A class of finite-time dual neural networks for solving quadratic programming problems and its k-winners-take-all application. *Neural Networks*, 39: 27-39.
- [24] P. Miao, Y. J. Shen, X. H. Xia (2014) Finite time dual neural networks with a tunable activation function for solving quadratic programming problems and its application, *Neurocomputing* 143: 80-89.
- [25] Y. J. Shen, P. Miao, Y. H. Huang, Y. Shen (2015) Finite-time stability and its application for solving time-varying Sylvester equation by recurrent neural network, *Neural Process Lett* 42: 763-784.
- [26] P. Miao, Y. J. Shen, Y. H. Huang, Y. W. Wang (2015) Solving time-varying quadratic programs based on finite-time Zhang neural networks and their application to robot tracking, *Neural Computing & Application*, 26(3): 693-703.
- [27] S. Bhat, D. Bernstein (2000) Finite-time stability of continuous autonomous systems, *SIAM Journal on Control and Optimization*, 38: 751-766.
- [28] Y. J. Shen, X. H. Xia (2008) Semi-global finite-time observers for nonlinear systems, *Automatica*, 44: 3152-3156.
- [29] Clarke (1969) *Optimization and non-smooth analysis*. New York: Wiley.
- [30] D. Kinzderlehrer and G. Stampcchia (1980) *An Introduction to variational inequalities and their applications*. New York: Academic.
- [31] Y. S. Xia (1996) ODE methods for solving convex programming problems with bounded variables, *Chinese Journal Numerical Mathematics Applied*, 4: 402-408.
- [32] J. K. Hale (1980) *Ordinary differential equations*, Pure and Applied Mathematics XXI, second ed., Krieger, Malabar, FL.
- [33] X. B. Gao (2003) A neural network for solving nonlinear convex programming problems, *IEEE Transactions on Neural Networks*, 15(3): 613-621.
- [34] X. H. Yuan, B. Cao, B. Yang, Y. B. Yuan (2008) Hydrothermal scheduling using chaotic hybrid differential evolution, *Energy Conversion and Management*, 49: 3627-3633.
- [35] X. Guan, B. Peter (1995) Nonlinear approximation method in Lagrangian relaxation based algorithms for hydrothermal scheduling. *IEEE Transactions on Power Systems*, 10(2): 772-778.
- [36] R. Naresh, J. Sharma (1999) Two-phase neural network based solution technique for short term hydrothermal scheduling, *IEE Proceedings Generation, Transmission and Distribution*, 146(6): 657-663.
- [37] G. Zhang, Y. Shen (2015) Exponential stabilization of memristor-based chaotic neural networks with time-varying delays via intermittent Control, *IEEE Transactions on Neural Networks and Learning Systems*, 26(7): 1431-1441.
- [38] Y. W. Wang, T. Bian, J. Xiao, C. Wen (2015) Global synchronization of complex dynamical networks through digital communication with limited data rate, *IEEE Transactions on Neural Networks and Learning Systems*, DOI: 10.1109/TNNLS.2014.2387443.
- [39] Y. Zhang, Y. Shen, X. Wang, L. Cao (2015) A Novel Design for Memristor-Based Logic Switch and Crossbar Circuits, *IEEE Trans. Circuits Syst. I, Reg. Papers*, 62(5): 1402-1411.
- [40] G. Zhang G, Y. Shen (2014) Exponential synchronization of delayed memristor-based chaotic neural networks via periodically intermittent control, *Neural Networks*, 2014, 55: 1-10.