Neural Networks 80 (2016) 110–117

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

A recurrent neural network for adaptive beamforming and array correction

Hangjun Che [a], Chuandong Li [a,*], Xing He [a], Tingwen Huang [b]

[a] School of Electronics and Information Engineering, Southwest University, Chongqing 400715, PR China

[b] Department of Mathematics, Texas A&M University at Qatar, Doha, P.O.Box 23874, Qatar

ARTICLE INFO

Article history:
Received 12 June 2015
Received in revised form 8 March 2016
Accepted 22 April 2016
Available online 29 April 2016

Keywords:
Beamforming optimization
Exact solutions
Recurrent neural network
Array state

ABSTRACT

In this paper, a recurrent neural network (RNN) is proposed for solving adaptive beamforming problem. In order to minimize sidelobe interference, the problem is described as a convex optimization problem based on linear array model. RNN is designed to optimize system's weight values in the feasible region which is derived from arrays' state and plane wave's information. The new algorithm is proven to be stable and converge to optimal solution in the sense of Lyapunov. So as to verify new algorithm's performance, we apply it to beamforming under array mismatch situation. Comparing with other optimization algorithms, simulations suggest that RNN has strong ability to search for exact solutions under the condition of large scale constraints.

1. Introduction

Optimization theory is a commonly-used method in engineering area (Corsini & Leoreanu, 2003; Hudson, 1981; Lebret & Boyd, 1997; Li, Yu, Yu, Chen, & Wang, in press; Li, Yu, Yu, Huang, & Liu, 2015; Monzingo & Miller, 1980; Razavizadeh, Ahn, & Lee, 2014; Yu, Ser, Er, Gu, & Li, 2009). In wireless communications, antenna arrays provide an efficient means to detect and process singles arriving from different directions. Recently, the theoretical and applied aspects of beamforming have received great research interests during the last decades (Corsini & Leoreanu, 2003; Lebret & Boyd, 1997; Yu et al., 2009). Compared with a single antenna that is limited in directivity and bandwidth, an array of sensors can arbitrarily change its beampattern through altering array's weight. The fifth generation of mobile communications (5G) also considers it as the key technology (Razavizadeh et al., 2014).

In particular, beamforming is designed to optimize according to some specific criteria, such as minimum variance, maximum entropy, and maximum signal-to-interference-plus-noise ratio (Hudson, 1981; Monzingo & Miller, 1980). Different criteria produce different models such as quadratic programming, second order cone programming and some nonconvex programming. Every model has its own advantages and disadvantages.

Because of large scale of sensors in reality, the need for real-time processing of signals is urgent. Based on the previous work of Tank and Hopfield (1986), there has been increasing achievements in the theory, methodology and application of recurrent neural network for optimization. In 1988, the dynamical canonical nonlinear programming circuit (NPC) was introduced by Kennedy and Chua for nonlinear programming by utilizing a finite penalty parameter which only obtain the approximate optimal (Kennedy & Chua, 1988). Latter on, the Lagrangian network (based on Lagrangian method) was proposed by Zhang and Constantinides for solving convex nonlinear programming problem (Zhang & Constantinides, 1992; Zhu, Zhang, & Constantinides, 1992). And then the primal–dual neural network with global stability was proposed for providing the exact solutions for linear and quadratic programming problems (Bouzerdoum & Pattison, 1993; He, Huang, Li, Che, & Dong, 2015; Lillo, Loh, Hui, & Zak, 1993; Maa & Shanblatt, 1992; Tao, Cao, Xue, & Qiao, 2001; Wang, 1993, 1994; Xia, 1996a; Xia & Wang, 2000). The projection neural network developed by Xia and Wang was proposed to efficiently solve many optimization problems and variational inequalities (Gao, 2004; He, Huang, & Yu, in press; He, Li, & Huang, 2014; He, Li, Huang, & Li, 2014; He, Yu, Huang, Li, & Li, 2014; Hu & Wang, 2007; Hu & Zhang, 2009; Liu & Cao, 2010; Liu, Cao, & Xia, 2005; Liu & Wang, 2011; Xia, 1996b; Xia & Wang, 1998). Moreover, in recent years, recurrent neural network has been widely applied in wide range of programming problems (Li, Yu, Huang, Chen, & He, 2015; Liu & Wang, 2015a, 2015b).

* Corresponding author.
E-mail addresses: chj11711@163.com (H. Che), licdswu@163.com (C. Li), hexingdoc@swu.edu.cn (X. He), tingwen.huang@qatar.tamu.edu (T. Huang).

http://dx.doi.org/10.1016/j.neunet.2016.04.010
0893-6080/

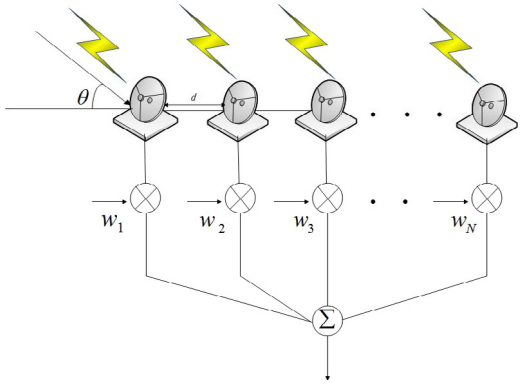

Fig. 1. N sensors' linear array.

In this paper, we study a adaptive beamforming problem based on linear array model. Firstly, the problem is converted to a convex optimization problem according to arrays' state and plane wave's information. The number of constraints is mainly determined by sampling interval's width, traditional optimization methods are time-consuming and getting poor results. Inspired by the effectiveness and efficiency of neural network optimization method, we adopt this algorithm dealing with beamforming and array mismatch correction. This paper's bright spot is to design a recurrent neural network (RNN) based on beamforming problem's auxiliary cost function and gradient method. Using Lyapunov function theory, it is proven that the proposed neural network is Lyapunov stable and globally convergent to the optimal solution. Comparing with other four classical optimization algorithms, RNN shows superior performance.

The rest of this paper is organized as follows. In the next section, a convex beamforming model is built. In Section 3, a novel recurrent neural network based on gradient method is described. The theoretical analysis of the proposed neural network is presented in Section 4. In Section 5, four comparison algorithms are introduced briefly. Simulation results on the numerical example are given in Section 6. Finally, Section 7 concludes this paper.

2. Beamforming model description

In this part, we consider a uniform linear array (ULA) consisting of N isotropic sensors with inter-element spacing d (see Fig. 1). A plane wave of wavelength λ is incident on the array from an angle θ, let the first sensor on the left be benchmark sensor, so the array steering vector is $s(\theta) = [1, e^{-j\phi}, \ldots, e^{-j(N-1)\phi}]^H$, where $\phi = \frac{2d\pi\cos(\theta)}{\lambda}$. The array response function $G(\theta)$ is given by $G(\theta) = s(\theta)^H\mathbf{w}$, $\mathbf{w} = (w_1, \ldots, w_N)^H$ is array weight vector. Actually, we require that the maximum gain of the main lobe is obtained at θ_m and the maximum sidelobe as small as possible. So the beamforming can be formulated as following Min–Max optimization problem:

Assumption 1. Let Θ_{tar} and Θ_{rej} be target angle set and reject angle, system output is mainly interfered by non-target angle plane wave.

Assumption 2. ALL receiving antennas are omnidirectional antennas and isotropic, transmission signal is narrow-band signal.

Remark 1. In fact, we know that antennas have various kind of loss such as medium loss, cable loss and signal reflection loss. In this paper, we regard side lobe as the main interference object. So we consider each antenna as an ideal point source, generally speaking they are isotropic.

Remark 2. Generally speaking, Θ_{tar} always has only one angle, however, Θ_{rej} always contains many angles. Meanwhile, distance between two sensors is also an influential factor, it is usually set as half of wave's length.

$$\min \max_{\theta_j \in \Omega} |s^H(\theta_j)\mathbf{w}|^2 \quad j = 1, \ldots, M \tag{1}$$
$$\text{s.t. } s^H(\theta_m)\mathbf{w} = 1$$
$$\Omega = [-90^\circ, 90^\circ] \setminus \theta_m.$$

In order to convert Min–Max problem to a standard optimization problem form, we add a variable z then model (1) can be rewritten as:

$$\min_{w} z \tag{2}$$
$$\text{s.t. } |s^H(\theta_j)\mathbf{w}|^2 \leqq z$$
$$s^H(\theta_m)\mathbf{w} = 1.$$

For the sake of expressing the problem from the view of real-value functions and variables, let $s(\theta_j) = rs_j + iIs_j, s(\theta_m) = rs_m + iIs_m, \mathbf{w} = rw + iIw$, where $rs_j = (rs_{j1}, \ldots, rs_{jN})^T, Is_j = (Is_{j1}, \ldots, Is_{jN})^T\ j = 1, \ldots, M, rs_m = (rs_{m1}, \ldots, rs_{mN})^T, Is_m = (Is_{m1}, \ldots, Is_{mN})^T, rw = (rw_1, \ldots, rw_N)^T, Iw = (Iw_1, \ldots, Iw_N)^T$, where $i = \sqrt{-1}$.

So model (2) could be reconstructed as a new form:

$$\min\ e^T x \tag{3}$$
$$\text{s.t. } x^T a_j a_j^T x + x^T b_j b_j^T x - e^T x \leq 0 \quad j = 1, \ldots, M$$
$$a_m^T x = 1$$
$$b_m^T x = 0$$
$$e, x, a_j, b_j, a_m, b_m \in \Re^{2N+1}$$
$$e = (0, \ldots, 1)^T, \qquad x = (rw_1, \ldots, rw_N, Iw_1, \ldots, Iw_N, z)^T,$$
$$a_j = (rs_{j1}, \ldots, rs_{jN}, -Is_{j1}, \ldots, -Is_{jN}, 0)^T,$$
$$b_m = (Is_{m1}, \ldots, Is_{mN}, rs_{m1}, \ldots, rs_{mN}, 0)^T,$$
$$b_j = (Is_{j1}, \ldots, Is_{jN}, rs_{j1}, \ldots, rs_{jN}, 0)^T,$$
$$a_m = (rs_{m1}, \ldots, rs_{mN}, -Is_{m1}, \ldots, -Is_{mN}, 0)^T.$$

The above may be expressed more compactly by defining:

$$A_j = a_j a_j^T + b_j b_j^T \quad B_m = \begin{pmatrix} a_m^T \\ b_m^T \end{pmatrix} \qquad b = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$
$$A_j \in \Re^{(2N+1)\times(2N+1)}, \qquad B_m \in \Re^{2\times(2N+1)}.$$

So model (3) can be simplified as following expression:

$$\min\ e^T x \tag{4}$$
$$\text{s.t. } x^T A_j x - e^T x \leq 0 \quad j = 1, \ldots, M$$
$$B_m x = b.$$

Obviously, model (4) is a convex optimization problem, it consists of quadratic constraints and linear constraints. Based on model (4), some new models which have practical significance can be evolved:

2.1. Power constraint model

We know that, Power consumption is very important in practical applications. In the beamforming signal synthesizer, every signal comes from a receiving sensor amplifying the signal

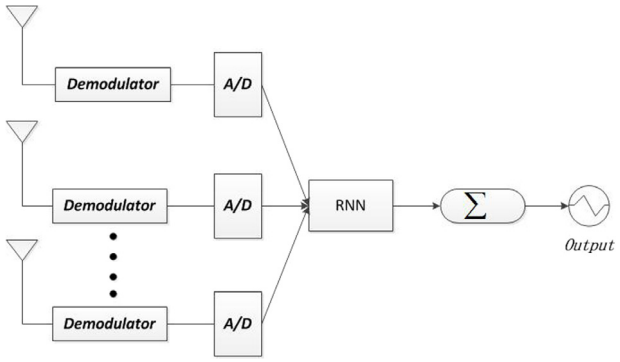

Fig. 2. Block diagram of recurrent neural network beamforming process.

by power amplifier (PA). This means that in consideration of energy efficiency or power control. x_i is not arbitrary value.

$$\begin{aligned} &\min\ e^T x \\ &\text{s.t. } x^T A_j x - e^T x \le 0 \quad j = 1, \ldots, M \\ &B_m x = b \\ &\sum_{i=1}^{N} \sqrt{x_i^2 + x_{i+N}^2} \le \delta. \end{aligned} \tag{5}$$

δ is the maximum power which system can provide and the third constraint is a nonconvex term. It is more difficult to solve problem.

2.2. Specific angle constraint model

$$\begin{aligned} &\min\ e^T x \\ &\text{s.t. } x^T A_j x - e^T x \le 0 \quad j = 1, \ldots, M \\ &B_n x = b_n, \end{aligned} \tag{6}$$

$$B_n = \begin{pmatrix} a_m^T \\ b_m^T \\ a_z^T \\ b_z^T \end{pmatrix}, \qquad b_n = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

$$a_z = (rs_{z1}, \ldots, rs_{zN}, -Is_{z1}, \ldots, -Is_{zN}, 0)^T,$$
$$b_z = (Is_{z1}, \ldots, Is_{zN}, rs_{z1}, \ldots, rs_{zN}, 0)^T,$$

a_z and b_z are two vectors defining similarly as a_m, b_m. This model is applied to eliminate the interference of certain angle θ_z.

Although (Luo, 2003) has mentioned a maximization of the Signal-to-Interference-plus-Noise Ratio (SINR) model, but its interference-plus-noise covariance matrix may not be available and is usually approximated by the data covariance matrix. In contrast, model (4) is simpler and easier to implement.

Remark 3. In reality, the incident angle of received plane wave is a continuous variable from $-90°$ to $90°$. Therefore, model (4) originally has unlimited number of constraints. To simplify the problem, we sample incident angle every small interval. So model (4) is converted to limited constraints optimization problem and the number of constraints depends on the size of the sampling interval. Fig. 2 shows the block diagram of recurrent neural network beamforming process.

3. Recurrent neural network algorithms

Definition 1. Consider the problem to minimize $f(x)$ subject to $g_i(x) \le 0$, $h_j(x) = 0$, where $i = 1, \ldots, m, j = 1, \ldots, q$. Both objective function and constraint functions are differentiable. If a point $(x^*, \lambda^*, \mu^*) \in \Re^n \times \Re^m \times \Re^q$ satisfies the conditions:

$$\begin{aligned} &\nabla f(x^*) + \sum_{i=1}^{n} \nabla \lambda_i^* g_i(x^*) + \sum_{j=1}^{q} \nabla \mu_j^* h_j(x^*) = 0, \\ &\lambda_i^* \ge 0, \qquad g_i(x^*) \le 0, \qquad \lambda_i^* g_i(x^*) = 0, \\ &h_j(x^*) = 0, \qquad \mu_j^* \in \Re^q \end{aligned}$$

then x^* is said to be a KKT point.

Definition 2. $F : \Re^n \to \Re^n$ is continuously differentiable function, Γ is a closed and convex subset of $\Re^n$. VIP(Γ, F) is called variational inequality problems if $x^* \in \Gamma$ and $\forall x \in \Gamma$ there is:

$$F(x^*)^T (x - x^*) \ge 0. \tag{7}$$

Definition 3. $f: \Re^n \to \Re$ is a convex function, if it satisfies the following condition:

$$\begin{aligned} &f(\lambda x + (1-\lambda)y) \le \lambda f(x) + (1-\lambda) f(y), \\ &\quad \forall x, y \in \Re^n,\ \lambda \in [0, 1]. \end{aligned}$$

According to definition, if $f(x)$ is continuously differentiable, above condition has two equivalent expressions:

1. $f(y) - f(x) \ge \nabla f(x)^T (y - x),\ \forall x, y \in \Re^n$;
2. $(\nabla f(y) - \nabla f(x))^T (y - x) \ge 0,\ \forall x, y \in \Re^n$.

Definition 4. $L(c) = \{x \in \Omega | f(x) \le c,\ c \in \Re\}$ is called the level set of $f(x)$, $\Omega \subset \Re^n$ is feasible region.

Lemma 1. *Let $\Phi : \Omega \subseteq \Re^n \to \Re$, where Ω is unbounded. Then all level sets of Φ are bounded if and only if $\lim_{k\to\infty} \Phi(x^k) = +\infty$ whenever $x^k \subset D$ and $\lim_{k\to\infty} \|x^k\| = +\infty$.*

For the sake of having concise derivation, we let number of inequality and equality constraints in model (4) be m and q. Similarly, $x \in \Re^n$. Based on Definitions 1–4 and beamforming model (4), a recurrent neural network is described as follows:

$$\frac{d}{dt}\begin{pmatrix} x \\ \lambda \\ \mu \end{pmatrix} = -\Lambda \begin{pmatrix} \nabla_x(\lambda + X^T A X - E^T x)^+ \\ \quad \times \{(\lambda + X^T A X - E^T x)^+ - \lambda\} \\ \quad + \{\nabla_x^2 (E^T x) + \nabla_x^2 (X^T A X - E^T x)\lambda_I \\ \quad + \nabla_x^2 (B_m x - b)\mu_I\}\{\nabla_x (E^T x) \\ \quad + \nabla_x (X^T A X - E^T x)\lambda + \nabla_x (B_m x - b)\mu\} \\ \quad + \nabla_x (B_m x - b)(B_m x - b); \\ \{\nabla_\lambda (\lambda + X^T A X - E^T X)^+ - 1\} \\ \quad \times \{(\lambda + X^T A X - E^T x)^+ - \lambda\} \\ \quad + \nabla_x (X^T A X - E^T x)^T \{\nabla_x (E^T x) \\ \quad + \nabla_x (X^T A X - E^T x)\lambda + \nabla_x (B_m x - b)\mu\}; \\ \nabla_x (B_m x - b)^T \{\nabla_x (E^T x) \\ \quad + \nabla_x (X^T A X - E^T x)\lambda + \nabla_x (B_m x - b)\mu\} \end{pmatrix}. \tag{8}$$

In model (8), $\Lambda = \mathrm{diag}(a_1, a_2, \ldots, a_{n+m+q})$, $\Lambda \in \Re^{(n+m+q)\times(n+m+q)}$, all members are positive. $(\cdot)^+$ denotes $\max[0, \cdot]$, ∇_x, ∇_λ represent partial derivatives of variable x and λ, ∇_x^2 represents second-order partial derivative of x. I_n denotes $n \times n$ identity matrix $A = [A_1^T, \ldots, A_m^T]^T$, $I = [\underbrace{I_n, \ldots, I_n}_{m}]^T$, $E = [\underbrace{e, \ldots, e}_{m}]$, $X = I * x$, $\lambda = [\lambda_1, \ldots, \lambda_m]^T$, $\mu = [\mu_1, \ldots, \mu_q]^T$. $\lambda_I = [\lambda_1 * I_n, \ldots, \lambda_m * I_n]^T$, $\mu_I = [\mu_1 * I_n, \ldots, \mu_q * I_n]^T$. In order to get a concise expression, we replace $\{(\lambda + X^T A X - E^T x)^+ - \lambda\}$ with $A(x, \lambda)$, similarly, using $\nabla_x B(x, \lambda, \mu)$ substituting $\{\nabla_x (E^T x) + \nabla_x (X^T A X -$

$E^T x)\lambda + \nabla_x (B_m x - b)\mu\}$. $(B_m x - b)$ is replaced by $C(x)$. Simplified neural network model is stated as follows:

$$\frac{d}{dt}\begin{pmatrix} x \\ \lambda \\ \mu \end{pmatrix} = -\Lambda \begin{pmatrix} \nabla_x A(x,\lambda)A(x,\lambda) + \nabla_x^2 B(x,\lambda,\mu)\,\nabla_x \\ \quad \times B(x,\lambda,\mu) + \nabla_x C(x)C(x); \\ \nabla_\lambda A(x,\lambda)A(x,\lambda) + \nabla_x (X^T AX - E^T x)^T\,\nabla_x \\ \quad \times B(x,\lambda,\mu); \\ \nabla_x C(x)^T\,\nabla_x B(x,\lambda,\mu) \end{pmatrix}. \quad (9)$$

Remark 4. Model (9) can solve a wide class of optimization whose objection function and constraints are continuously differentiable. In our network, every variable corresponds to a neuro unit, having the ability to process data in parallel.

4. Theoretical analysis

In this section, some necessary theoretical analysis about network (9) for solving problem (4) will be given. Firstly, model (9) is proven to be stable in the sense of Lyapunov. And then, we will show it globally convergent to an optimal solution.

Definition 5. M is called equilibria set of the neural network (9), $\bar{z} = (\bar{x}^T, \bar{\lambda}^T, \bar{\mu}^T)^T$ if and only if:

$$M = \left\{ \bar{z} \in \Re^{n+m+q} : \frac{d\bar{z}}{dt} = 0 \right\}.$$

Theorem 1. *If model (4) is a convex optimization problem, object function and constraints are continuously differentiable, then the global optimal solution of problem (4) is an equilibrium point of model (9).*

Proof. According to model (9), equilibrium point is found if it satisfies:

$$\frac{dz}{dt} = (0) \quad (10)$$

we can easily find following conditions also satisfy formula (10):

$A(\bar{x}, \bar{\lambda}, \bar{\mu}) = 0$, where it can be equivalently written as:

$$\bar{\lambda} \geq 0, \qquad -(\bar{X}^T A\bar{X} - E^T\bar{x}) \geq 0, \qquad \bar{\lambda}^T(\bar{X}^T A\bar{X} - E^T\bar{x}) = 0,$$
$$B(\bar{x}, \bar{\lambda}) = 0, \qquad C(\bar{x}) = 0.$$

Obviously, $(\bar{x}, \bar{\lambda}, \bar{\mu})$ is a KKT point what defined in Definition 1. It is also the sufficient condition of global optimal solution to convex optimization. We can say KKT point is included in the equilibria set.

Theorem 2. *For any initial point $z(t_0) = (x(t_0)^T, \lambda(t_0)^T, \mu(t_0)^T)^T \in \Re^{n+m+q}$ in the feasible region Ω, where $\Omega = \{x | x^T A_j x - e^T x \leq 0, B_m x = 0, j = 1, \ldots, M\}$, model (9) is Lyapunov stable and converges to exact solution.*

Proof. Firstly, according to method mentioned in Xia and Wang (1998), constructing an auxiliary function based on KKT condition:

$$\Psi(x,\lambda,\mu) = \frac{1}{2}\{\|A(x,\lambda)\|_2^2 + \|C(x)\|_2^2 + \|B(x,\lambda,\mu)\|_2^2\}$$

and model (9) is deduced from following:

$$\frac{dz}{dt} = -\Lambda \bigtriangledown \Psi(x,\lambda,\mu) \quad (11)$$

we define a Lyapunov function as:

$$V(z(t)) = \frac{1}{2}(z(t) - z^*)^T P(z(t) - z^*) \quad (12)$$

where P is a positive definite matrix, obviously $V(z(t))$ is positive number. We let $P = \Lambda^{-1}$.

$$\begin{aligned} \frac{dV(z(t))}{dt} &= \frac{dV(z(t))}{dz(t)}\frac{dz(t)}{dt} \\ &= -\begin{pmatrix} (x(t) - x^*)^T \\ (\lambda(t) - \lambda^*)^T \\ (\mu(t) - \mu^*)^T \end{pmatrix}^T P\Lambda \\ &\quad \times \begin{pmatrix} \nabla_x A(x,\lambda)A(x,\lambda) + \nabla_x^2 B(x,\lambda,\mu) \\ \quad \times \nabla_x B(x,\lambda,\mu) + \nabla_x C(x)C(x); \\ \nabla_\lambda A(x,\lambda)A(x,\lambda) \\ \quad + \nabla_x (X^T AX - E^T x)^T\,\nabla_x B(x,\lambda,\mu); \\ \nabla_x C(x)^T\,\nabla_x B(x,\lambda,\mu) \end{pmatrix} \\ &= -\{(x(t) - x^*)^T(\nabla_x A(x,\lambda)A(x,\lambda) \\ &\quad + \nabla_x^2 B(x,\lambda,\mu)\,\nabla_x B(x,\lambda,\mu) + \nabla_x C(x)C(x)) \\ &\quad + (\lambda(t) - \lambda^*)^T(\nabla_\lambda A(x,\lambda)A(x,\lambda) \\ &\quad + \nabla_x (X^T AX - E^T x)^T\,\nabla_x B(x,\lambda,\mu)) \\ &\quad + (\mu(t) - \mu^*)^T(\nabla_x C(x)^T\,\nabla_x B(x,\lambda,\mu))\} \\ &\leq \frac{1}{2}\{\|A(x^*,\lambda^*)\|_2^2 + \|C(x^*)\|_2^2 + \|B(x^*,\lambda^*,\mu^*)\|_2^2 \\ &\quad - \|A(x,\lambda)\|_2^2 - \|C(x)\|_2^2 - \|B(x,\lambda,\mu)\|_2^2\} \leq 0 \end{aligned}$$

because of $\Psi(x,\lambda,\mu) = \frac{1}{2}\{\|A(x,\lambda)\|_2^2 + \|C(x)\|_2^2 + \|B(x,\lambda,\mu)\|_2^2\}$ is convex, according to the first condition of Definition 3, the first sign of inequality is hold. From Definition 1, we can easily find z^* is minimum point of $\Psi(x,\lambda,\mu)$, so the second inequality sign is hold too. If model (9) gets equilibrium point, there having:

$$\frac{dv(z(t))}{dt} = 0, \qquad \Psi(x,\lambda,\mu) = \Psi(x^*,\lambda^*,\mu^*).$$

So M is the global optimal solution set of model (4), based on Theorem 1, we know $\frac{dz}{dt} = 0$ if and only if KKT condition is hold, that is to say, ever equilibrium point is optimal solution. According to Lemma 1 and LaSalle's invariance principle, $V(z(t))$ has a bounded level set, the $z(t)$ converges to $\aleph = \{z(t) \in \Theta | \frac{dV(z(t))}{dt} = 0\}$, $\Theta = \{z(t) \in \Re^{n+m+q} | V(z(t)) \leq V(z(t_0))\}$.

Corollary 1. *If part of arrays in model (4) are damaged, the proposed neural network can also adaptively obtain a correction waveform based on the remaining arrays' state.*

Proof. This process is shown in Fig. 3, we can find that the system will judge array's state every check interval, if system runs well, it will remind the same weights. When some arrays in the model (4) are damaged, that means corresponding elements in a_j, b_j, a_m, b_m are set to zero. For example, if the third array is out of work, rs_{j3}, Is_{j3}, rs_{m3}, Is_{m3} are zero. Obviously, When model (4) is under mismatch situation, it also remain convex. Similar to Theorems 1 and 2, we can still obtain an optimal solution through using algorithm (9).

Remark 5. Different from the neural network based on penalty function, model (9) can converge to the exact solution in feasible region, comparing with other intelligent method such as particle swarm algorithm (PSO), it also display strong search ability.

5. Comparison algorithm

For the sake of describing algorithm's performance, this section will briefly introduce four other optimization algorithms to compare with the proposed algorithm. These algorithms are also used in beamforming and similar optimization problems (Che, Li, He, & Huang, 2015; Kennedy & Chua, 1988; Kurup, Himdi, &

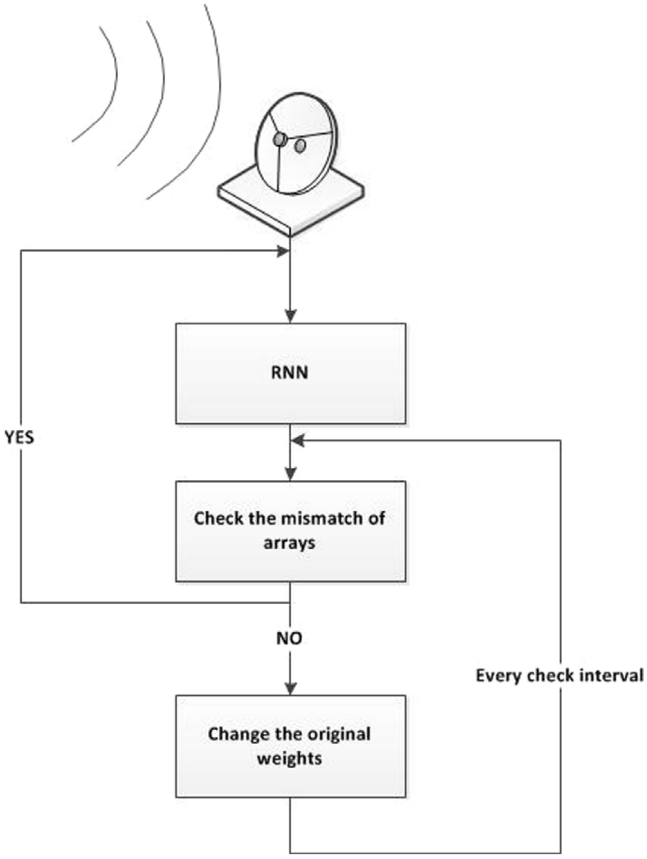

Fig. 3. Array correction system.

Rydberg, 2003; Robinson & Rahmat-Samii, 2004). One algorithm is recurrent neural network based on penalty method (RNNBP), Others are Particle Swarm Optimization (PSO), Artificial Bee Colony Algorithm (ABC), Shuffled Frog Leaping Algorithm (SFLA).

The first step of RNNBP is to convert an optimization problem into an unconstrained problem described as follows (Kennedy & Chua, 1988):

$$\min\ E_s(x) = f(x) + \frac{s}{2}\left\{\sum_{i=1}^{m}[g_i(x)]^2 + \sum_{j=1}^{q}[h_j(x)]^2\right\} \tag{13}$$

where s is the penalty factor. The second step is constructing an associated system of ordinary differential equation:

$$\frac{dx}{dt} = -\Lambda\frac{dE_s(x)}{dt}. \tag{14}$$

RNNBP has been proven to be stable in terms of Lyapunov and converge to approximate optimal solution when s approaching infinity.

In PSO algorithm, each particle in a swarm represents a potential solution. Define $p_i = (p_{i1}, \ldots, p_{in})^T \in \Re^n$ as the best previous position obtaining the maximum fitness value of each particle, $g_i = (g_1, \ldots, g_n)^T$ as the best position of all particle. Denote $x_i = (x_{i1}, \ldots, x_{in})^T \in \Re^n$ as the position of the ith particle and $v_i = (v_{i1}, \ldots, v_{in}) \in \Re^n$ as the velocity of ith particle, where n is the dimension of each particle (Demarcke, Rogier, Goossens, & De Jaeger, 2009).

$$x_{i+1} = x_i + x_i, \tag{15}$$

$$v_{i+1} = v_i + cr_1(p_i - x_i) + cr_2(g - x_i) \tag{16}$$

where c is always set 2, r_1 and r_2 are two random numbers in [0, 1]. When fitness function gets the maximum, we also regard that particle's position as an optimal solution. In comparison process, fitness function is defined as:

$$E_p(x) = -\left\{f(x) + \sum_{i=1}^{m}[g_i^+(x)]^2 + \sum_{j=1}^{q}[h_j(x)]^2\right\} \tag{17}$$

the performance of PSO is decided by the number of particles and Iterations. When $E_p(x)$ has achieved the max, the objective function $f(x)$ get the min in feasible region.

Differential Evolution (DE) is also a efficient method to solve beamforming problem, it includes three main processes: Mutation, Crossover and Selection. For each target vector $x_{i,G}$, $i = 1, 2, 3, \ldots, NP$, a mutant vector is produced by following equation:

$$v_{i,G+1} = x_{r_1,G} + F \times (x_{r_2,G} - x_{r_3,G}) \tag{18}$$

$r_1, r_2, r_3 \in 1, 2, \ldots, NP$ are random integer indexes and $F > 0$. The randomly chosen integers r_1, r_2 and r_3 are also chosen to be different from the running index i, so that $NP \geq 4$. $F \in [0, 2]$ is a real and constant factor which controls the amplification of the differential variation $(x_{r_2,G} - x_{r_3,G})$.

In crossover process, the aim is to increase the diversity of the perturbed parameter vectors. To this end, the trial vector:

$$u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \ldots, u_{Di,G+1}) \tag{19}$$

is formed, where

$$u_{ji,G} = \begin{cases} v_{ji,G+1} & \text{if (rand } b(j) \leq CR) \text{ or } j = rnbr(i); \\ x_{ji,G} & \text{if (rand } b(j) > CR) \text{ and } j \neq rnbr(i); \\ j = 1, 2, \ldots, D. \end{cases} \tag{20}$$

In (20), rand $b(j)$ is the jth evaluation of a uniform random number generator with outcome $\in$ [0, 1]. CR is the crossover constant $\in$ [0, 1] which has to be determined by the user. $rnbr(i)$ is a randomly chosen index $\in 1, 2, \ldots, D$ which ensures that $u_{i,G+1}$ gets at least one parameter from $v_{i,G+1}$.

After the crossover, we need to decide whether or not it should become a member of generation $G + 1$, the trial vector $u_{i,G+1}$ is compared to the target vector $x_{i,G}$ using the greedy criterion. If vector $u_{i,G+1}$ yields a smaller cost function value than $x_{i,G}$, then $x_{i,G+1}$ is set to $u_{i,G+1}$; otherwise, the old value $x_{i,G}$; otherwise, the old value $x_{i,G}$ is retained. More information about (DE) could be found in Storn and Price (1997).

In SFLA, the search process is similar to the ABC. We also regard E_p as SFLA's fitness function, the update expression could be found in formula 23, 24 of Che et al. (2015).

6. Numerical examples

In the following, we adopt the presented neural network mentioned above to deal with model (4). ALL simulations are based on the Matlab 2012b experimental platform.

Example 1. The first example is to test five optimization algorithms' performance under different number of sensors. Since the code and simulation platform are different, there is no significance to consider algorithms' time consumption. In the experiments, we set sampling interval 2°, the central angle of main lobe is 0°, so there have 90 inequality constraints and two equality constraints. Because of the Lagrange multipliers, RNN has more unknown variables, this difference is shown in Table 1. On the other hand, Figs. 4 and 5 show the convergence of RNN and RNNBP which are proven by previous theory. We regard the maximum sidelobe gain and feasible solutions as two standards of evaluation. In Table 1, we know RNN can get low sidelobe gain and feasible solutions, other four algorithms could not always guarantee their solutions in feasible region. Figs. 6 and 7 show the beamforming results of RNN.

Table 1
Comparison of five algorithms.

Precision: $e = 10^{-3}$	RNN	RNNBP	PSO	DE	SFLA
8 sensors' number of variables	109	17	17	17	17
8 sensors' maximum sidelobe gain	−12.89 dB	−1.138 dB	−6.13 dB	−8.56 dB	−5.54 dB
8 sensors' number of inequalities satisfied	All	All	Not stable	Not stable	Not stable
8 sensors' number of equalities satisfied	All	All	Not stable	Not stable	Not stable
16 sensors' number of variables	125	33	33	33	33
16 sensors' maximum sidelobe gain	−12.7 dB	−2.97 dB	−5.27 dB	−7.85 dB	−6.47 dB
16 sensors' number of inequalities satisfied	All	86	Not stable	Not stable	Not stable
16 sensors' number of equalities satisfied	All	All	Not stable	Not stable	Not stable
32 sensors' number of variables	177	65	65	65	65
32 sensors' maximum sidelobe gain	−5.83 dB	−7.93 dB	−9.003 dB	−8.357 dB	−5.334 dB
32 sensors' number of inequalities satisfied	All	86	Not stable	Not stable	Not stable
32 sensors' number of equalities satisfied	All	All	Not stable	Not stable	Not stable

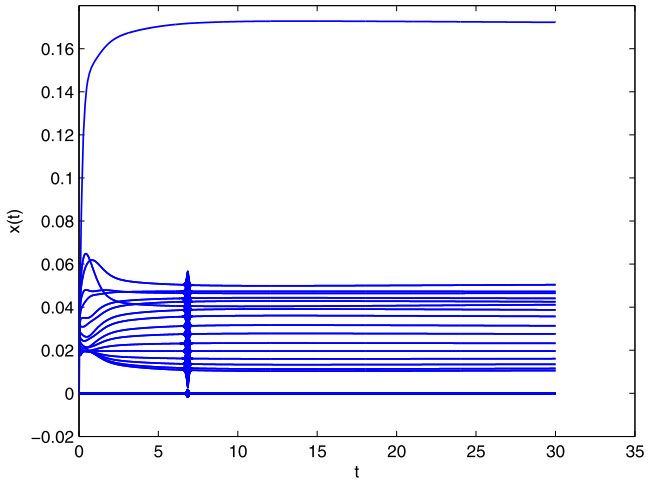

Fig. 4. Transient behavior of $x(t)$ of RNN with random initial points in 32 sensors' situation.

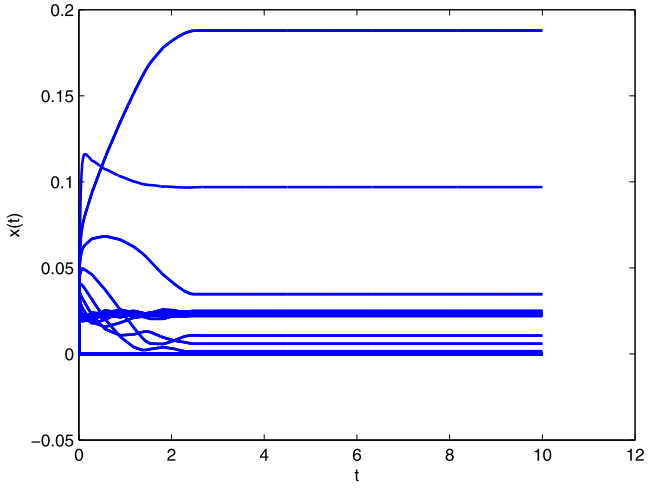

Fig. 5. Transient behavior of $x(t)$ of RNNBP with random initial points in 32 sensors' situation.

Example 2. In this test, we want to show the distinction between adopting different sampling interval (SI). From Fig. 8, we can find that bigger SI leads to wider mainlobe width and higher sidelobe. This interesting phenomenon displays smaller SI produces more constraints, in order to meet all these constraints, the algorithm should balance the system performance and constraints.

Example 3. Another trial which can test algorithm's performance is array mismatch correction. If some of sensors are stopping working sometime, result of beamforming will be dissatisfied.

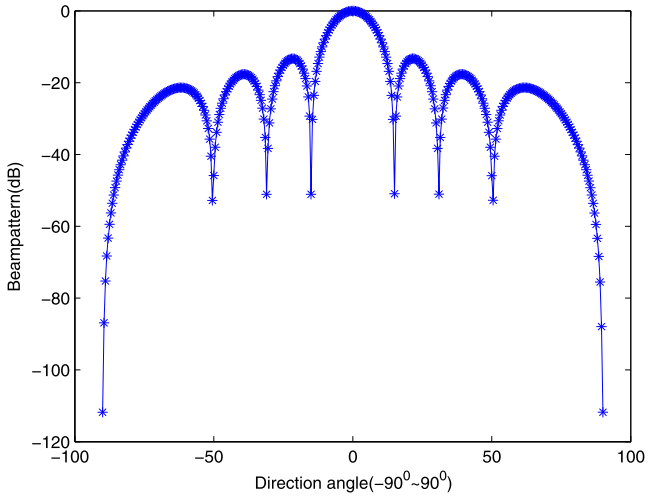

Fig. 6. 8 sensors' beamforming based on RRN.

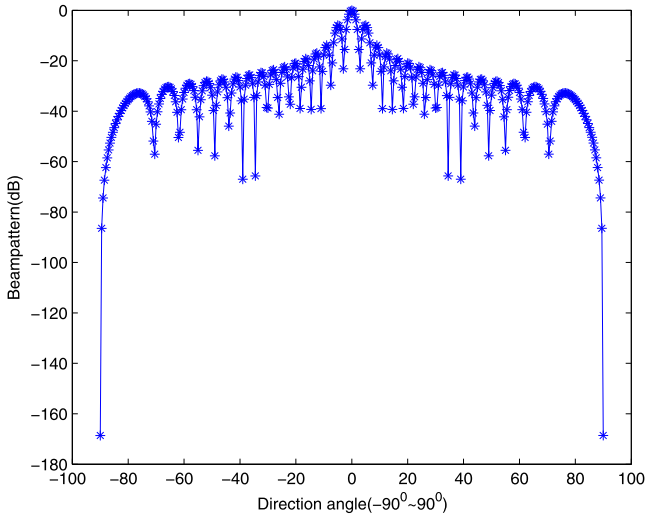

Fig. 7. 32 sensors' beamforming based on RNN.

Sidelobe interference increases without control under mismatch situation. We use three groups of experiments to test and verify proposed algorithm has great ability to correct array mismatch. Figs. 9–11 show mismatch correction in case of different sensor number. The red line is mismatch beamforming, the green line displays correction result, angle of main lobe is −10°. Table 2 shows mismatch sensors' serial number. The experiments show that RNN could efficiently correct beamforming when array lost in mismatch.

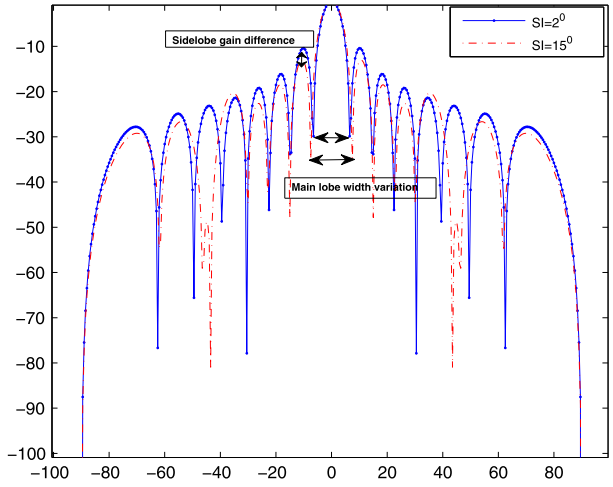

Fig. 8. The influence of sampling interval.

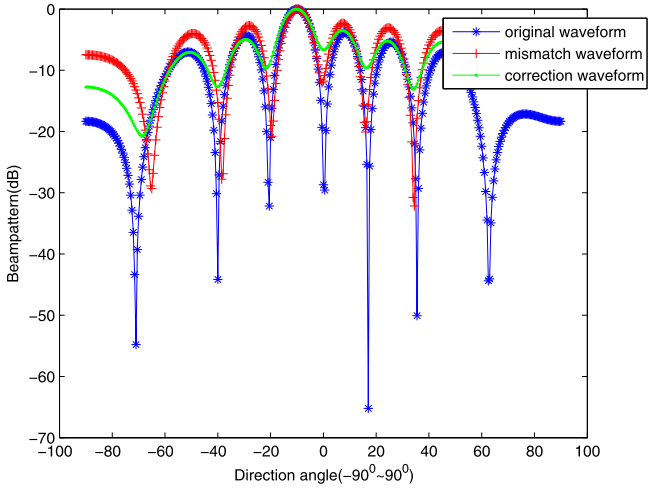

Fig. 9. 8 sensors' beamforming correction. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

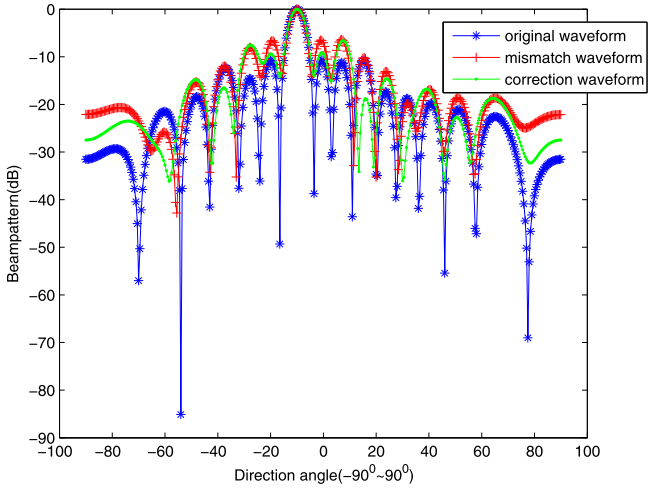

Fig. 10. 16 sensors' beamforming correction. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Mismatch sensors' number.

8 sensors	16 sensors	32 sensors
2, 4, 6	9, 10, 11, 12	10, 11, 12, 13, 22, 23, 24, 25

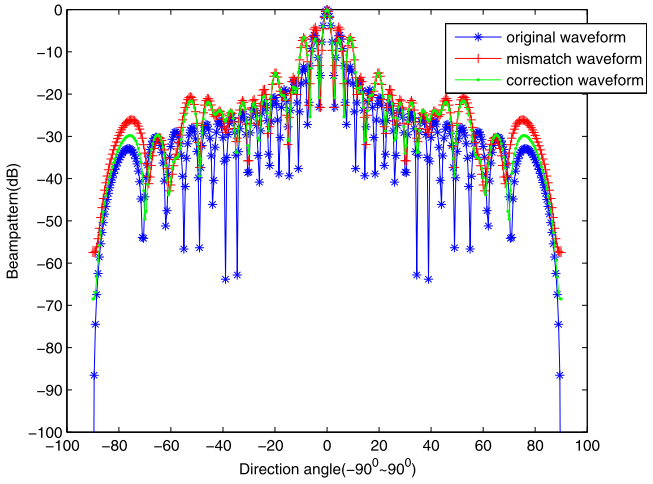

Fig. 11. 32 sensors' beamforming correction. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7. Conclusion

In this paper, we build a convex beamforming model and the model can be extended by adding other constraints. By using KKT condition and Lyapunov function, convergence of presented recurrent neural network can be guaranteed. Through the experiments, the new algorithm has ability to converge to the optimal solution and the width of the mainlobe will be controlled in a small interval. All results are shown that the new algorithm is stable and has strong searching capability.

Acknowledgments

This work is supported by Natural Science Foundation of China (Grant nos: 61403313, 61374078). It is also supported by Chongqing Research Program of Basic Research and Frontier Technology (No. cstc2015jcyjBX0052) and the Natural Science Foundation Project of Chongqing CSTC (Grant no. cstc2014jcyjA40014). Graduate Student Research Innovation Project of Chongqing (Project No. CYS14053) and Fundamental Research Funds for the Central Universities (Project No. XDJK2016B017) also supports this work. This publication was made possible by NPRP Grant no. NPRP 4-1162-1-181 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

Bouzerdoum, A., & Pattison, T. R. (1993). Neural network for quadratic optimization with bound constraints. *IEEE Transactions on Neural Networks*, *4*(2), 293–304.

Che, H., Li, C., He, X., & Huang, T. (2015). An intelligent method of swarm neural networks for equalities-constrained nonconvex optimization. *Neurocomputing*, *167*, 569–577.

Corsini, P., & Leoreanu, V. (2003). *Advances in mathematics (Dordrecht), Applications of hyperstructure theory*. Dordrecht: Kluwer.

Demarcke, P., Rogier, H., Goossens, R., & De Jaeger, P. (2009). Beamforming in the presence of mutual coupling based on constrained particle swarm optimization. *IEEE Transactions on Antennas and Propagation*, *57*(6), 1655–1666.

Gao, X. B. (2004). A novel neural network for nonlinear convex programming. *IEEE Transactions on Neural Networks*, *15*(3), 613–621.

He, X., Huang, T., Li, C., Che, H., & Dong, Z. (2015). A recurrent neural network for optimal real-time price in smart grid. *Neurocomputing*, *149*, 608–612.

He, X., Huang, T., Yu, J., et al. (2016). An inertial projection neural network for solving variational inequalities. *IEEE Transactions on Cybernetics*, in press.

He, X., Li, C., & Huang, T. (2014). A recurrent neural network for solving bilevel linear programming problem. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(4), 824–830.

He, X., Li, C., Huang, T., & Li, C. (2014). Neural network for solving convex quadratic bilevel programming problems. *Neural Networks*, *51*, 17–25.

He, X., Yu, J., Huang, T., Li, C., & Li, C. (2014). Neural network for solving Nash equilibrium problem in application of multiuser power control. *Neural Networks*, *57*, 73–78.
Hu, X., & Wang, J. (2007). A recurrent neural network for solving a class of general variational inequalities. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, *37*(3), 528–539.
Hu, X., & Zhang, B. (2009). An alternative recurrent neural network for solving variational inequalities and related optimization problems. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, *39*(6), 1640–1645.
Hudson, J. E. (1981). *Adaptive array principles, Vol. 11*. IET.
Kennedy, M. P., & Chua, L. O. (1988). Neural networks for nonlinear programming. *IEEE Transactions on Circuits and Systems*, *35*(5), 554–562.
Kurup, D. G., Himdi, M., & Rydberg, A. (2003). Synthesis of uniform amplitude unequally spaced antenna arrays using the differential evolution algorithm. *IEEE Transactions on Antennas and Propagation*, *51*(9), 2210–2217.
Lebret, H., & Boyd, S. (1997). Antenna array pattern synthesis via convex optimization. *IEEE Transactions on Signal Processing*, *45*(3), 526–532.
Li, C., Yu, X., Huang, T., Chen, G., & He, X. (2015). A generalized Hopfield network for nonsmooth constrained convex optimization: Lie derivative approach. *IEEE Transactions on Neural Networks and Learning Systems*, *27*(2), 308–321.
Li, Chaojie, Yu, Xinghuo, Yu, Wenwu, Chen, Guo, & Wang, Jianhui (2016). Efficient computation for sparse load shifting in demand side management. *IEEE Transactions on Smart Grid*, in press.
Li, Chaojie, Yu, Xinghuo, Yu, Wenwu, Huang, Tingwen, & Liu, Zhi-wei (2015). Distributed event-triggered scheme for economic dispatch in smart grids, IEEE Transactions on Industrial Informatics. http://dx.doi.org/10.1109/TII.2015.2479558.
Lillo, W. E., Loh, M. H., Hui, S., & Zak, S. H. (1993). On solving constrained optimization problems with neural networks: a penalty method approach. *IEEE Transactions on Neural Networks*, *4*(6), 931–940.
Liu, Q., & Cao, J. (2010). A recurrent neural network based on projection operator for extended general variational inequalities. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, *40*(3), 928–938.
Liu, Q., Cao, J., & Xia, Y. (2005). A delayed neural network for solving linear projection equations and its analysis. *IEEE Transactions on Neural Networks*, *16*(4), 834–843.
Liu, Q., & Wang, J. (2011). A one-layer recurrent neural network for constrained nonsmooth optimization. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, *41*(5), 1323–1333.
Liu, Q., & Wang, J. (2015a). A projection neural network for constrained quadratic minimax optimization. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(11), 2891–2900.
Liu, Q., & Wang, J. (2015b). A second-order multi-agent network for bound-constrained distributed optimization. *IEEE Transactions on Automatic Control*, *60*(12), 3310–3315.
Luo, Z. Q. (2003). Applications of convex optimization in signal processing and digital communication. *Mathematical Programming*, *97*(1–2), 177–207.
Maa, C. Y., & Shanblatt, M. A. (1992). Linear and quadratic programming neural network analysis. *IEEE Transactions on Neural Networks*, *3*(4), 580–594.
Monzingo, R. A., & Miller, T. W. (1980). *Introduction to adaptive arrays*. SciTech Publishing.
Razavizadeh, S., Ahn, M., & Lee, I. (2014). Three-Dimensional Beamforming: A new enabling technology for 5G wireless networks. *Signal Processing Magazine, IEEE*, *31*(6), 94–101.
Robinson, J., & Rahmat-Samii, Y. (2004). Particle swarm optimization in electromagnetics. *IEEE Transactions on Antennas and Propagation*, *52*(2), 397–407.
Storn, R., & Price, K. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341–359.
Tank, D., & Hopfield, J. J. (1986). Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Transactions on Circuits and Systems*, *33*(5), 533–541.
Tao, Q., Cao, J., Xue, M., & Qiao, H. (2001). A high performance neural network for solving nonlinear programming problems with hybrid constraints. *Physics Letters A*, *288*(2), 88–94.
Wang, J. (1993). Analysis and design of a recurrent neural network for linear programming. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, *40*(9), 613–618.
Wang, J. (1994). A deterministic annealing neural network for convex programming. *Neural Networks*, *7*(4), 629–641.
Xia, Y. (1996a). A new neural network for solving linear and quadratic programming problems. *IEEE Transactions on Neural Networks*, *7*(6), 1544–1548.
Xia, Y. (1996b). A new neural network for solving linear programming problems and its application. *IEEE Transactions on Neural Networks*, *7*(2), 525–529.
Xia, Y., & Wang, J. (1998). A general methodology for designing globally convergent optimization neural networks. *IEEE Transactions on Neural Networks*, *9*(6), 1331–1343.
Xia, Y. S., & Wang, J. (2000). On the stability of globally projected dynamical systems. *Journal of Optimization Theory and Applications*, *106*(1), 129–150.
Yu, Z. L., Ser, W., Er, M. H., Gu, Z., & Li, Y. (2009). Robust adaptive beamformers based on worst-case optimization and constraints on magnitude response. *IEEE Transactions on Signal Processing*, *57*(7), 2615–2628.
Zhang, S., & Constantinides, A. G. (1992). Lagrange programming neural networks. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, *39*(7), 441–452.
Zhu, X., Zhang, S., & Constantinides, A. G. (1992). Lagrange neural networks for linear programming. *Journal of Parallel and Distributed Computing*, *14*(3), 354–360.