# Chapter 2

# RECURRENT NEURAL NETWORKS FOR OPTIMIZATION: THE STATE OF THE ART

**Youshen Xia and Jun Wang**

**Department of Mechanical & Automation Engineering**
**The Chinese University of Hong Kong**
**Shatin, New Territories, Hong Kong**

## I. INTRODUCTION

Optimization problems arise in a wide variety of scientific and engineering applications including signal processing, system identification, filter design, function approximation, regression analysis, and so on. In many practical optimization problems such as the planning of power systems and routing of telecommunication systems, the numbers of decision variables and constraints are usually very large. It is even more challenging when a large-scale optimization procedure has to be performed in real time to optimize the performance of a dynamical system. For such applications, classical optimization techniques may not be competent due to the problem dimensionality and stringent requirement on computational time. One possible and very promising approach to real-time optimization is to apply artificial neural networks. Neural networks are composed of many massively connected neurons. Resembling more or less their biological counterparts in structures, artificial neural networks are representational and computational models composed of interconnected simple processing elements called artificial neurons. In processing information, the processing elements in an artificial neural network operate concurrently and collectively in a parallel and distributed fashion. Because of the inherent nature of parallel and distributed information processing in neural networks, the convergence rate of the solution process is not decreasing as the size of the problem increases. Furthermore, unlike other parallel algorithms, neural networks can be implemented physically in designated hardware such as application-specific integrated circuits where optimization is carried out in a truly parallel and distributed manner. This feature is particularly desirable for real-time optimization in decentralized decision-making situations. Neural networks are promising computational models for solving large-scale optimization problems in real time. Therefore, the neural network approach can solve optimization problems in running times at the orders of magnitude much faster than the most popular optimization algorithms executed on general-purpose digital computers.

Neural network research stemmed back from McCulloch and Pitts' pioneering work a half century ago. Since then, numerous neural network models have been developed. One of the well-known classic neural network models is the Perceptron developed by Rosenblatt. The Perceptron is a single-layer adaptive feedforward network of threshold logic units, which possess some learning capability.

© 2001 by CRC Press LLC

Another important early neural network model is the Adaline which is a one-layer linear network using the delta learning rule for learning. The Perceptron and Adaline were designed primarily for the purpose of pattern classification. Given a set of input-output training patterns, the Perceptron and Adaline could learn from the exemplar patterns and adapt their parametric representations accordingly to match the patterns. The limitation of the Perceptron and Adaline is that they could only classify linearly separable patterns because, among others, they lacked an internal representation of stimuli.

The first attempt to develop analog circuits for solving linear programming problems was perhaps Pyne in 1956 [Pyne, 1956]. Soon after, some other circuits were proposed for solving various optimization problems. In 1986, Tank and Hopfield [Hopfield and Tank, 1985; Tank and Hopfield, 1986] introduced a linear programming neural network implemented by using an analog circuit which is well suited for applications that require on-line optimization. Their seminal work has inspired many researchers to investigate alternative neural networks for solving linear and nonlinear programming problems. Many optimization neural networks have been developed. For example, Kennedy and Chua [Kennedy and Chua, 1988] proposed a neural network for solving nonlinear programming problems. This network inculde the Tank and Hopfield network as a special case. The disadvantages of this network is that it contains penalty parameters and thus its equilibrium points correspond to approximate optimal solutions only. To overcome the shortcoming, Rodríguez-Vázquez et al. [1990] proposed a switched-capacitor neural network for solving a class of optimization problems. This network is suitable when the optimal solution lies in the feasible region only. Otherwise, the network may have no equilibrium point. Wang [Wang, 1994] proposed a deterministic annealing neural network for solving convex programming. This network guarantees an optimal solution can be obtained. Yet, the given sufficient condition is not easy to be verfied sometimes. From the optimization point of view, most of the methods employed by these existing neural networks belong to either the penalty function method or Lagrangian method. For more discussion on the advantages and disadvantages of these models and their modification, see Cichocki and Unbehauen [1993]. More recently, using the gradient and projection methods Bouzerdoum and Pattison [Bouzerdoum and Pattison, 1993] presented a neural network for solving quadratic optimization problems with bounded variables only. The network has the good performence in computation and implementation but can not solve general linear and quadratic programming problems. By the dual and projection methods Xia and Wang developed some neural networks for solving general linear and quadratic programming problems. These new neural networks have shown to be of good performence in computation and implementation.

Organized in two parts, this chapter is going to discuss the primal-dual neural networks for solving linear and quadratic programming problems (LP and QP) and develop the neural network for solving linear complementary problems (LCP). Following a unified method for designing neural network models, the first part of this chapter describes in detail primal-dual recurrent neural networks, with continuous time, for solving LP and QP. The second part of this chapter focuses on

© 2001 by CRC Press LLC

primal -dual discrete time neural networks for QP and LCP. The discrete assignment neural networks are described in detail.

# II. CONTINUOUS-TIME NEURAL NETWORKS FOR QP AND LCP

## A. PROBLEMS AND DESIGN OF NEURAL NETWORKS

### 1. Problem Statement

We consider convex quadratic programming with bound contraints:

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{2}x^T Ax + c^T x \\ \text{subject to} \quad & Dx = b, \\ & 0 \le x \le d \end{aligned} \tag{1}$$

where $x \in \Re^n$ is the vector of decision variables, $A \in \Re^{n\times n}$ is a positive semidefinite matrix, $b \in \Re^m, c \in \Re^n, d \in \Re^n$ are constant column vectors, $D \in \Re^{m\times n}$ is a coefficient matrix, $m \le n$. When $d = \infty$, (1) become a standard quadratic programming:

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{2}x^T Ax + c^T x \\ \text{subject to} \quad & Dx = b, x \ge 0 \end{aligned} \tag{2}$$

When $A = 0$, (1) becomes linear programming with bound contraints:

$$\begin{aligned} \text{Minimize} \quad & c^T x \\ \text{subject to} \quad & Dx = b, \\ & 0 \le x \le d \end{aligned} \tag{3}$$

We consider also linear complementary problems below: Find a vector $z \in R^l$ such that

$$z^T(Mz + q) = 0, (Mz + q) \ge 0, z \ge 0 \tag{4}$$

where $q \in R^l$, and $M \in R^{l\times l}$ is a positive semidefinite matrix but not necessarily symmetric. LCP has been recognized as a unifying description of a wide class of problems including LP and QP, fixed point problems and bimatrix equilibrium points [Bazaraa, 1990]. In electrical engineering applications, it is used for the analysis and modeling of piecewise linear resistive circuits [Vandenberghe, 1989].

### 2. Design of Neural Networks

A neural network can operate in either continuous-time or discrete-time form. A continuous- time neural network described by a set of ordinary differential equations enables us to solve optimization problems in real time due to the massively parallel operations of the computing units and due to its real-time convergence rate. In comparison, discrete-time models can be considered as special cases of discretization of continuous-time models. Thus, in this part, we first discuss continuous-time neural networks.

© 2001 by CRC Press LLC

The procedure of a continuous-time neural network design to optimization usually begins with the formulation of an energy function based on the objective function and constraints of the optimization problem under study. Ideally, the minimum of a formulated energy function corresponds to the optimal solution (minimum or maximum, whatever applicable) of the original optimization problem. Clearly, a convex energy function should be used to eliminate local minima. In nontrivial constrained optimization problems, the minimum of the energy function has to satisfy a set of prespecified constraints. The majority, if not all, of the existing neural network approaches to optimization formulates an energy function by incorporating objective function and constraints through functional transformation and numerical weighting. Functional transformation is usually used to convert constraints to a penalty function to penalize the violation of constraints. Numerical weighting is often used to balance constraint satisfaction and objective minimization (or maximization). The way the energy function is formulated plays an important role in the optimization problem-solving procedure based on neural networks.

The second step in designing a neural network for optimization usually involves the derivation of a dynamical equation (also known as state equation or motion equation) of the neural network based on a formulated energy function. The dynamical equation of a neural network prescribes the motion of the activation states of the neural network. The derivation of a dynamical equation is crucial for success of the neural network approach to optimization. A properly derived dynamical equation can ensure that the state of neural network reaches an equilibrium and the equilibrium state of the neural network satisfies the constraints and optimizes the objective function of the optimization problems under study. Presently, the dynamical equations of most neural networks for optimization are derived by letting the time derivative of a state vector to be directly proportional to the negative gradient of an energy function.

The next step is to determine the architecture of the neural network in terms of the neurons and connections based on the derived dynamical equation. An activation function models important characteristics of a neuron. The range of an activation function usually prescribes the domain of state variables (the state space of the neural network). In the use of neural networks for optimization, the activation function depends on the feasible region of decision variables delimited by the constraints of the optimization problem under study. Specifically, it is necessary for the state space to include the feasible region. Any explicit bound on decision variables can be realized by properly selecting the range of activation functions. The activation function is also related to the energy function. If the gradient-based method is adopted in deriving the dynamical equation, then the convex energy function requires an increasing activation function. Precisely, if the steepest descent method is used, the activation function should be equal to the derivative of the energy function. Figure 1 illustrates four examples of energy functions and corresponding activation functions, where the linear activation function can be used for unbounded variables.

The last step in developing neural networks for optimization is usually devoted

© 2001 by CRC Press LLC

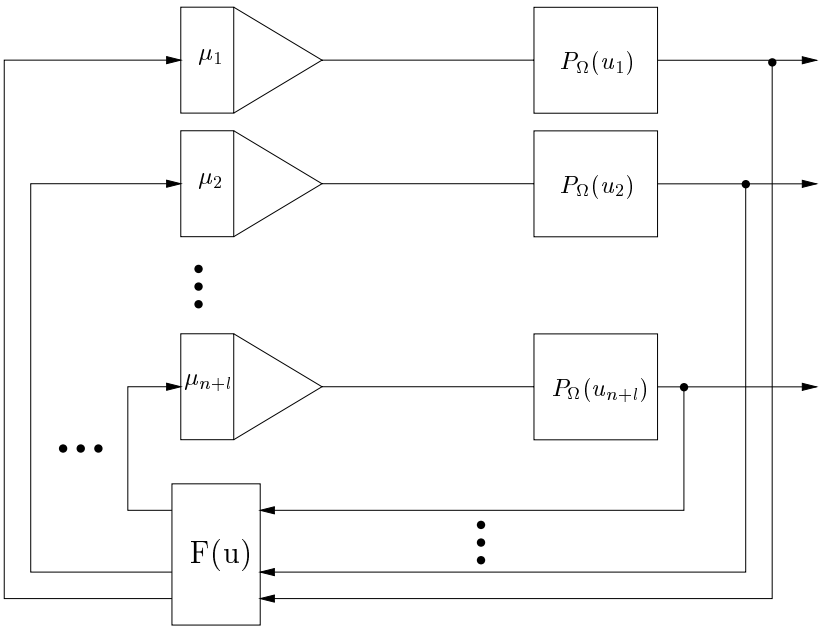


Figure 1. A block diagram of the neural network model in (5) (Xia, Y. and Wang, J., A general method for designing optimization neural networks with global converge nce, *IEEE Transactions on Neural Networks,* ©1998 IEEE)

© 2001 by CRC Press LLC

to simulation to test the performance of the neural network. Simulations can be performed numerically using commercial software packages or self-programmed simulators. Simulation can also be implemented physically in hardware (e.g., using off-the-shelf electronic components).

In summary, to formulate a optimization problem in terms of a neural network, there are two types of methods. One approach commonly used in developing an optimization neural network is to first convert the constrained optimization problem into an associated unconstrained optimization problem, and then design a neural network that solves the unconstrained problem with gradient methods. Another approach is to construct a set of differential equations such that their equilibrium points correspond to the desired solutions and then find an appropriate Lyapunov function such that all trajectory of the systems converges to some equilibrium points. Combining the above two types of methods, we give a deterministic procedure to be used directly to construct neural network models [Xia and Wang, 1998].

Step 1. Find a continuous function $\Phi : \Omega \subset R^{n+l} \rightarrow R$ such that its minima correspond to the exact or approximate solutions to P, where $\Omega = \{u = (u_1, ..., u_{n+l})^T | \ u_i$ satisfies some interval constraints and P denotes one of (1)-(4).$\}$

Step 2. Construct a continuous vector valued function $F : \Omega \subset R^{n+l} \rightarrow R^{n+l}$ such that the functions $F(u)$ and $\Phi(u)$ satisfy

(I) if $u^*$ is a minimizer of $\Phi$, then $F(u^*) = 0$,

(II) $(u - u^*)^T F(u) \leq -\alpha(\Phi(u) - \Phi(u^*)), \forall u \in \Omega$

where $F(u)$ satisfies local Lipschitz conditions, $\alpha > 0$ and fixed.

Step 3. Let the neural network model for solving P be represented by the following dynamic systems

$$\frac{du}{dt} = \Lambda F(u), \quad u \in \Omega \tag{5}$$

where $\Lambda = diag(\mu_1, ..., \mu_{n+l})$, and $\mu_i > 0$ which is to scale the convergence rate of (5).

Step 4. Based on the systems in (5), design the neural network architecture for solving P.

A block diagram of the neural network is shown in Fig. 1, where the projection operator $P_\Omega(u)$ enforces state vector $u$ in $\Omega$, which is defined by $P_\Omega(u) = [P_\Omega(u_1), ..., P_\Omega(u_{n+l})]^T$ and for $i \in L - I$, $P_\Omega(u_i) = u_i$; for $i \in I$,

$$P_\Omega(u_i) = \begin{cases} d_i & u_i < d_i \\ u_i & d_i \geq u_i \geq h_i \\ h_i & u_i > h_i \end{cases}$$

where $L = \{1, ..., n + l\}$ and $I \subset L$.

**3. Theoretical Result of the Method**

About the proposed method, we have the theoretical results below [Xia and Wang, 1998].

© 2001 by CRC Press LLC

**Theorem 1.** Any neural network derived from the proposed method is stable in the sense of Lyapunov and globally convergent to an exact or approximate solution to P.

**Proof.** Without loss of generality we assume that the set of minimizers of P is unbounded, and thus the set of global minimizers of $\Phi$ is unbounded.

First, we know from the first step of the method that an exact or approximate solution to P corresponds to a minmizer of $\Phi$, and from the second step that $F(u) = 0$ if and only if $u$ is a minimizer of $\Phi$. Thus it follows that $F(u) = 0$ if and only if $u$ is an exact or an approximate solution to P. That is, the equilibrium points of the system in (5) correspond to exact or approximate solutions to P.

Next, by the existence theory of ordinary differential equations [Miller and Michel, 1982], we see that for any an initial point taken in $\Omega$ there exists a unique and continuous solution $u(t) \subset \Omega$ for the systems in (5) over $[t_0, T)$ since the function $F(u)$ satisfies local Lipschitz conditions.

Now, we consider the positive definite function

$$V(u) = \frac{1}{2}\|\Lambda_1(u - u^*)\|_2^2, \quad u \in \Omega$$

where $\Lambda_1 = diag(\mu_1^{-\frac{1}{2}}, ..., \mu_{n+l}^{-\frac{1}{2}})$ and $u^*$ is a fixed minimizer of $\Phi$. From condition (II) we have

$$\begin{aligned} \frac{d}{dt}V(u) &= \frac{dV}{du}\frac{du}{dt} = (u - u^*)^T\Lambda_1^2\Lambda F(u) \\ &= (u - u^*)^T F(u) \leq -\alpha(\Phi(u) - \Phi(u^*)) \leq 0. \end{aligned} \tag{6}$$

Thus

$$\|u(t) - u^*\|_2 \leq \beta\|u(t_0) - u^*\|_2 \quad \forall t \in [t_0, T)$$

where $\beta$ is a positive constant. Then the solution $u(t)$ is bounded on $[t_0, T)$, and thus $T = \infty$. Moreover, the system in (5) is Lyapunov stable at each equilibrium point.

On the other hand, since $\lim_{k\to\infty} V(u^k) = +\infty$ whenever the sequence $u^k \subset \hat{\Omega}$ and $\lim_{k\to\infty} \|u^k\| = +\infty$, by Lemma 2 we see that all level sets of $V$ are bounded though all level sets of $\Phi$ are unbounded, thus $\hat{\Omega} = \{u \in \Omega | V(u) \leq V(u^0)\}$ is bounded. Because $V(u)$ is continuously differentiable on the compact set $\hat{\Omega}$ and $\{u(t)|t \geq t_0\} \subset \hat{\Omega}$ , it follows from the LaSalle's invariance principle that trajectories $u(t)$ converge to $\Sigma$, the largest invariant subset of the following set

$$E = \{u \in \hat{\Omega} \quad | \quad \frac{dV}{dt} = 0\}.$$

Note that if $dV/dt = 0$, then $(u - u^*)^T F(u) = 0$. So $-\alpha(\Phi(u) - \Phi(u^*)) \geq 0$ by condition (II). Thus $\Phi(u) = \Phi(u^*)$ and $u$ is an equilibrium point of the system in (7); i.e.,

$$\frac{du}{dt} = \Lambda F(u) = 0.$$

© 2001 by CRC Press LLC

Conversely, if $du/dt = 0$, then $F(u) = 0$, and $dV/dt = (u - u^*)^T F(u) = 0$. So $du/dt = 0$ if and only if $dV/dt = 0$. Hence

$$E = \{u \in \hat{\Omega} | \frac{du}{dt} = 0\}$$

Finally, let $\lim_{k\to\infty} u(t_k) = \hat{u}$, then $\hat{u} \in \Omega^*$. Therefore, for $\forall \epsilon > 0$ there exists $q > 0$ such that

$$\|\Lambda_1(u(t_k) - \hat{u})\| < \epsilon \quad k \geq q$$

Note that (6) holds for each $u^* \in \Omega^*$, then $\|\Lambda_1(u(t) - \hat{u})\|$ is decreasing as $t \to \infty$. It follows that

$$\|\Lambda_1(u(t) - \hat{u})\|_2 \leq \|\Lambda_1(u(t_{k_q}) - \hat{u})\|_2 < \epsilon \quad t \geq k_q,$$

then

$$\lim_{t\to\infty} \|\Lambda_1(u(t) - \hat{u})\|_2 = 0.$$

So

$$\lim_{t\to\infty} u(t) = \hat{u}\,.$$

Remark: From Theorem 1 we see that any neural network designed by using the proposed method is globally stable and convergent. Following the proposed method we will derive two neural network models for QP and LCP, in which equilibrium points give exact solutions and there is no need for penalty or variable parameter in the models.

## B. PRIMAL-DUAL NEURAL NETWORKS FOR LP AND QP

### 1. Neural Network Models

From the dual theory we see that the dual problem of (1) is as follows

$$\begin{array}{ll} \text{Maximize} & b^T y - \frac{1}{2} x^T A x - d^T v \\ \text{subject to} & Ax - D^T y + c + v \geq 0, \\ & v \geq 0 \end{array} \tag{7}$$

where $y \in \Re^m, v \in \Re^n$ are the vectors of dual decision variables. By the complementary slackness theorem [Bertsekas, 1982], $x^*$ and $(y^*, v^*)$ are optimal solutions respectively to the primal problem (1) and the dual problem (7) if and only if $x^*$ and $(y^*, v^*)$ satisfy $Dx^* = b, 0 \leq x^* \leq d, v^* \geq 0$, and the following complementary conditions

$$\begin{cases} (v^*)^T(d - x^*) = 0 \\ (x^*)^T(Ax^* - D^T y^* + c + v^*) = 0 \end{cases} \tag{8}$$

© 2001 by CRC Press LLC

It is easy to see that (8) is equivalent to the equation of projection

$$x^* = P_\Omega(x^* - Ax^* + D^T y^* - c)$$

where $\Omega = \{x \in \Re^n | 0 \le x \le d\}, P_\Omega(x) = [P_\Omega(x_1), P_\Omega(x_2), \ldots, P_\Omega(x_n)]^T$, and for $i = 1, 2, \ldots, n$,

$$P_\Omega(x_i) = \begin{cases} 0, & \text{if } x_i < 0 \\ x_i, & \text{if } 0 \le x_i \le d_i \\ d_i, & \text{if } x_i > d_i \end{cases}.$$

In Xia [1995], through improving the structure of the modifying extragradient algorithm [Marcotte, 1991], we proposed the following primal-dual neural network model for solving (2)

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = -\begin{Bmatrix} D^T(Dx - b) + \beta_0[2Ax - D^T y + c \\ -A(x - Ax + D^T y - c)^+] \\ \beta_0[D(x - Ax + D^T y - c)^+ - b] \end{Bmatrix} \tag{9}$$

and the following model for solving (3)

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = -\begin{Bmatrix} D^T(Dx - b) - \beta_1(D^T y + c) \\ \beta_1[DP_\Omega(x + D^T y - c) - b] \end{Bmatrix}, \tag{10}$$

respectively, where $x \in \Omega, y \in R^m$, $\beta_1 = \|x - P_\Omega(x + D^T y - c)\|_2^2$, $\beta_0 = \|x - (x - Ax + D^T y - c)^+\|_2^2$, $(x)^+ = \{[x_1]^+, \ldots, [x_l]^+\}^T$, and $[x_i]^+ = \max\{0, x_i\}$. Here, directly extending the structure of the above two models we can obtain the following neural network model for solving (1)

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = -\begin{Bmatrix} D^T(Dx - b) + \beta[2Ax - D^T y + c \\ -AP_\Omega(x - Ax + D^T y - c)] \\ \beta[DP_\Omega(x - Ax + D^T y - c) - b] \end{Bmatrix} \tag{11}$$

where $x \in \Omega, y \in R^m$, and $\beta = \|x - P_\Omega(x - Ax + D^T y - c)\|_2^2$. For simplicity, eqn. (11) can be written as follows:

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = -\begin{Bmatrix} Bx - q + \beta v(x, y) \\ \beta[Dg(x, y) - b] \end{Bmatrix} \tag{12}$$

where $B = D^T D, q = D^T b, v(x, y) = 2Ax - D^T y + c - Ag(x, y)$, and $g(x, y) = P_\Omega(x - Ax + D^T y - c)$. The primal-dual neural network consists of a multivariable system with an excitation function $v(x, y)$ and a multivariable decaying system $g(x, y)$ with a time-varying parameter $\beta$. Figure 2 illustrates the architecture of the primal-dual network.

**2. Global Convergence and Stability**

First, we introduce two lemmas.

**Lemma 1.** Let $u^* \in \Omega, u \in \Re^n$, then

$$[P_\Omega(u) - u^*]^T[u - u^*] \ge \|u^* - P_\Omega(u)\|_2^2.$$

© 2001 by CRC Press LLC

**Proof.** See [Gafni and Bertsekas, 1984].

**Lemma 2.** Let $\Phi_0(x,,y) = \|Dx-b\|_2^2+\|P_\Omega(x-Ax+D^Ty-c)-x\|_2^4$. Then $\Phi_0(x,,y) \geq 0$ and $\Phi_0(x,,y) = 0$ if and only if $(x,y)$ is an optimal solution to the original and dual problems, and if $\Phi_0(x,,y) = 0$, then $(x,y)$ is an equilibrium point of the system (11).

**Proof.** From (4) and the structure of the system (11) it is easy to know the conclusion of Lemma 2.

**Theorem 2.** Assume that the original problem has an optimal solution. Then the primal-dual network (11) is stable in the sense of Lyapunov and globally convergent to a point corresponding to the optimal solution of both (1) and (7).

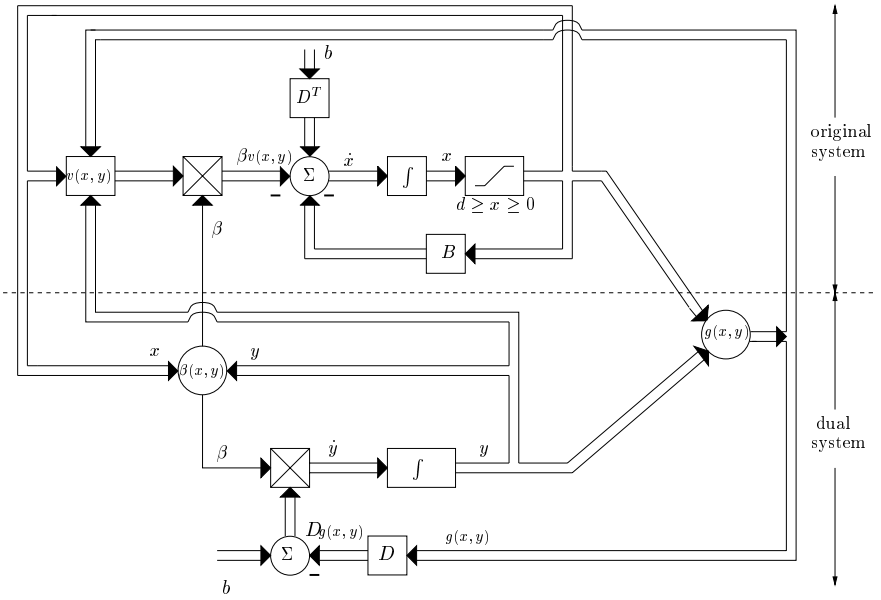


Figure 2. A block diagram of the neural network model in (12 )

**Proof.** Let $(x_0, y_0) \in \Omega \times \Re^m$ be an any given initial point. Since the projection function $P_\Omega(x - Ax + D^Ty - c)$ is Lipschitz continuous in $\Re^{n+m}$,

$$F(x,y) = \left\{ \begin{array}{c} D^T(Dx-b) + \beta[A(x - P_\Omega(x - Ax + D^Ty - c)) \\ +Ax - D^Ty - c] \\ \beta[DP_\Omega(x - Ax + D^Ty - c) - b] \end{array} \right\}$$

is also Lipschitz continuous. From the existence theory of ordinary differential equation we see that there exists a unique and continuous solution $w(t) = (x(t), y(t))$ with $w(0) = (x_0, y_0)$ for (11) on some interval $[0, \infty)$. Let $w^* =$

© 2001 by CRC Press LLC

$(x^*, y^*)$, where $x^*$ and $y^*$ is an optimal solution to (1) and (7), respectively. Then

$$\begin{aligned}
& - \quad (w(t) - w^*)^T F(x, y) \\
& = \quad \begin{pmatrix} x - x^* \\ y - y^* \end{pmatrix}^T \begin{pmatrix} D^T(Dx - b) + \beta[2Ax - D^T y + c \\ -AP_\Omega(x - Ax + D^T y - c)] \\ \beta[DP_\Omega(x - Ax + D^T y - c) - b] \end{pmatrix} \\
& = \quad \|Dx - b\|_2^2 + \beta(x - x^*)^T\{A[x - P_\Omega(x - Ax + D^T y - c)] \\
& + \quad (Ax - D^T y + c)\} + \beta(y - y^*)^T D[P_\Omega(x - Ax + D^T y - c) - x] \\
& + \quad \beta(y - y^*)^T(Dx - b) \\
& = \quad \|Dx - b\|_2^2 + \beta[x - P_\Omega(x - Ax + D^T y - c)]^T[Ax - D^T y + c] \\
& + \quad \beta(x - x^*)^T D^T(y - y^*) \\
& + \quad \beta[x - P_\Omega(x - Ax + D^T y - c)]^T[D^T y^* - Ax^* - c] \\
& + \quad \beta(x - x^*)^T(Ax - D^T y + c)
\end{aligned}$$

On one hand, using the same optimal value of both (1) and (7) we obtain that

$$\begin{aligned}
& - \quad (w(t) - w^*)^T F(x, y) \\
& \geq \quad \|Dx - b\|_2^2 + \beta[x - P_\Omega(x - Ax + D^T y - c)]^T[Ax - D^T y + c] \\
& + \quad \beta(x - x^*)^T A^T(x - x^*)
\end{aligned}$$

On the other hand, from Lemma 1 we let $u = x - Ax + D^T y - c, u^* = x$, then

$$[P_\Omega(x - Ax + D^T y - c) - x]^T[x - Ax + D^T y - c - x] \geq \|x - P_\Omega(x - Ax + D^T y - c)\|_2^2.$$

Therefore,

$$\begin{aligned}
& - \quad (w(t) - w^*)^T F(x, y) \\
& \geq \quad \|Dx - b\|_2^2 + \beta\|x - P_\Omega(x - Ax + D^T y - c)\|_2^2 + (x - x^*)^T A(x - x^*) \\
& \geq \quad \Phi_0(x, y)
\end{aligned}$$

since $\beta = \|x - P_\Omega(x - Ax + D^T y - c)\|_2^2$ and $A$ is a positive semidefinite matrix. So

$$(w(t) - w^*)^T F(x, y) \leq -(\Phi_0(x, y) - \Phi_0(x^*, y^*)). \tag{13}$$

So $\Phi_0(x, y)$ and $F(x, y)$ satisfy the conditions (I) and (II). By Theorem 1 we can obtain the proof of Theorem 2.

## C. NEURAL NETWORKS FOR LCP

### 1. Neural Network Model

According to Kinderlehrer [1980], $z^*$ is a solution to (4) if and only if $z^*$ satisfies the following equation

$$P_\Omega(z - Mz - q) = z \tag{14}$$

© 2001 by CRC Press LLC

where $\Omega = \{z \in R^l | z \geq 0\}$ and $P_\Omega(\cdot)$ denotes the projection onto the set $\Omega$. In Wang [1998] we proposed the following neural network model

$$\frac{dz}{dt} = F(z) = (I + M^T)((z - Mz - q)^+ - z) \tag{15}$$

The system described by (15) can be easily realized by a recurrent neural network with a two-layer structure shown in Fig. 3 where the vector $z$ is the network output, $\beta q = (q_i)$ is the network input vector, and $(I + M^T) = (m_{ij})$ and $M = (w_{ij})$ are weighted connections. We see from Fig. 3 that the proposed neural network can be implemented only by using simple hardware without analog multipliers for variables or penalty parameter. The circuit realizing the recurrent neural network consists of $2l^2 + 3l$ simple summers, $l$ integrators, and $2l^2$ weighted connections. The projection operator $(\cdot)^+$ may be implemented by using a piecewise activation function [Bouzerdoum and Pattison, 1993].

**2. Global Convergence and Stability**

Using Theorem 1 we can obtain the following result.

**Theorem 3.** Assume that $\Omega^* = \{z \in R^l | z \text{ satisfies (4)}\}$ is a nonempty set. Then the neural network of (15) is stable in the sense of Lyapunov and globally convergent to a solution to (4).

**Proof.** We see first that $z^* \in \Omega^*$ if and only if $\Phi(z^*) = 0$ where $\Phi(z) = z - (z - Mz - q)^+$. Therefore, $\Phi(u)$ and $F(u)$ satisfy the first step and condition (I) in the second step. By the fact that $(u - Mu - q)^+$ is the projection of $(x - Mx - q)$ onto $\Omega$ and $x^*$ is in $\Omega$, using Lemma 2 we have

$$[z^* - (u - Mu - q)^+]^T [Mz + q - z + (u - Mu - q)^+] \geq 0, \forall z \in R^l.$$

Since $z^*$ is a solution to (4), then

$$\{(u - Mu - q)^+ - z^*\}^T \{Mz^* + q\} \geq 0, \quad \forall z \in R^l.$$

Adding the two resulting inequalities yields

$$\{z^* - (u - Mu - q)^+\}^T \{Mz - Mx^* + (u - Mu - q)^+ - z\} \geq 0,$$

then

$$\begin{aligned} (z^* - z)^T M (z^* - z) &\leq (z - z^*)^T (I + M^T)(z - (u - Mu - q)^+) \\ &- \|z - (u - Mu - q)^+\|_2^2. \end{aligned}$$

Noting that $(z - z^*)^T M (z - z^*) \geq 0$, it follows that

© 2001 by CRC Press LLC

$$(z - z^*)^T (I + M^T)(z - (u - Mu - q)^+\} \leq -\|z - (u - Mu - q)^+\|_2^2,$$

thus

$$(z - z^*)^T F(z) \leq -(\Phi(z) - \Phi(z^*)).$$

So $\Phi(z)$ and $F(z)$ satisfy conditions (I) and (II). By Theorem 1 we can obtain the proof of Theorem 3.

## III. DISCRETE-TIME NEURAL NETWORKS FOR QP AND LCP

In many operations, discrete-time neural networks are preferable to their continuous-time counterparts because of the availability of design tools and the compatibility with computers and other digital devices. In this section, we discuss discrete-time neural networks. Generally speaking, a discrete-time neural network model can be obtained from a continuous-time one by converting differential equations into appropriate difference equations though discretization. However, the resulting discrete-time model is usually not guaranteed to be globally convergent to optimal solutions. In addition, difficulties may arise in selecting design parameters since the parameters may not be bounded in a small range. Moreover, it is not straightforward to realize variable parameters in hardware implementation of neural networks.

In this section, we present discrete-time recurrent neural networks with fixed design parameters. These networks are readily realized in digital circuits, and the proposed recurrent neural networks are guaranteed to globally converge to an optimal solution.

### A. NEURAL NETWORKS FOR QP AND LCP

We first consider the relation between LCP and the following QP

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}x^T Ax + c^T x \\ &\text{subject to} \quad Dx \geq b, \quad x \geq 0. \end{aligned} \tag{16}$$

It is easy to see that its dual problem is

$$\begin{aligned} &\text{maximize} \quad b^T y - \frac{1}{2}x^T Ax \\ &\text{subject to} \quad D^T y - Ax \leq c, \quad y \geq 0 \end{aligned} \tag{17}$$

where $y \in R^m$. From Lagrangian duality [Bertsekas, 1982], one can see that $x^*, y^*$ is an optimal solution to (16),(17), respectively, if and only if $(x^*, y^*)$ satisfies

$$\begin{aligned} c + A^T x - D^T y \geq 0, \quad x \geq 0, \quad x^T(c + A^T x - D^T y) = 0, \\ Dx - b \geq 0, \quad y \geq 0, \quad y^T(Dx - b) = 0. \end{aligned} \tag{18}$$

© 2001 by CRC Press LLC

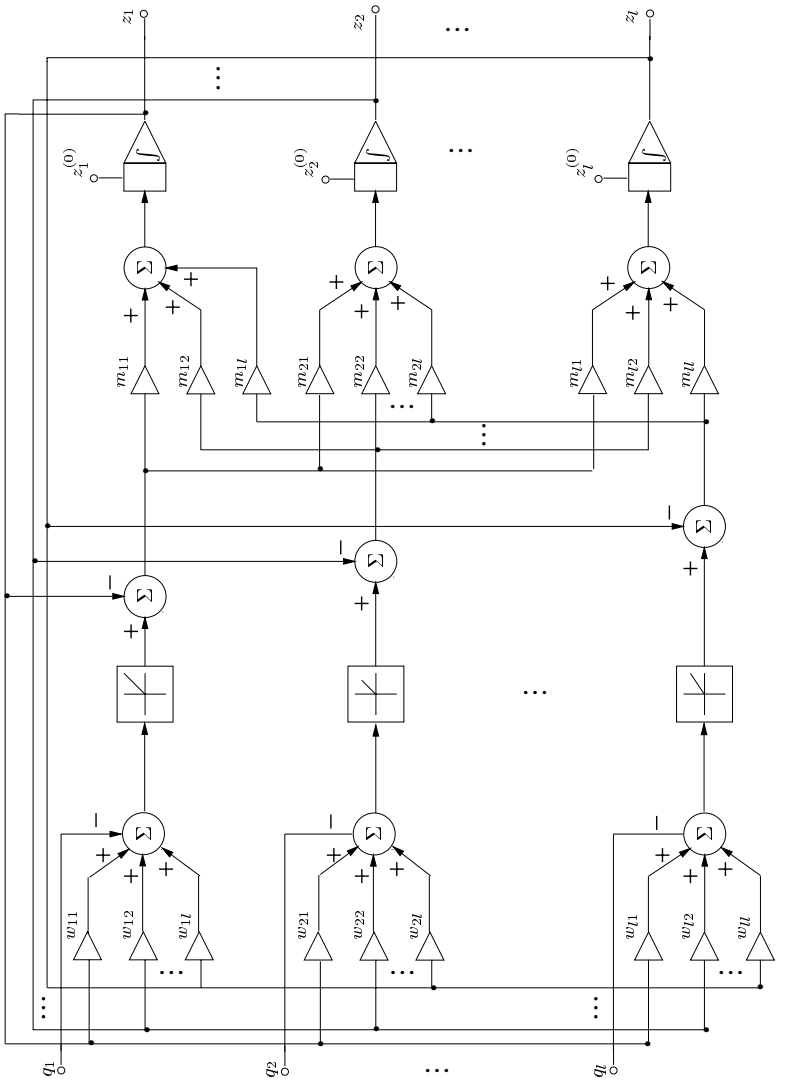


Figure 3. A block diagram of the neural network model in (15)

© 2001 by CRC Press LLC

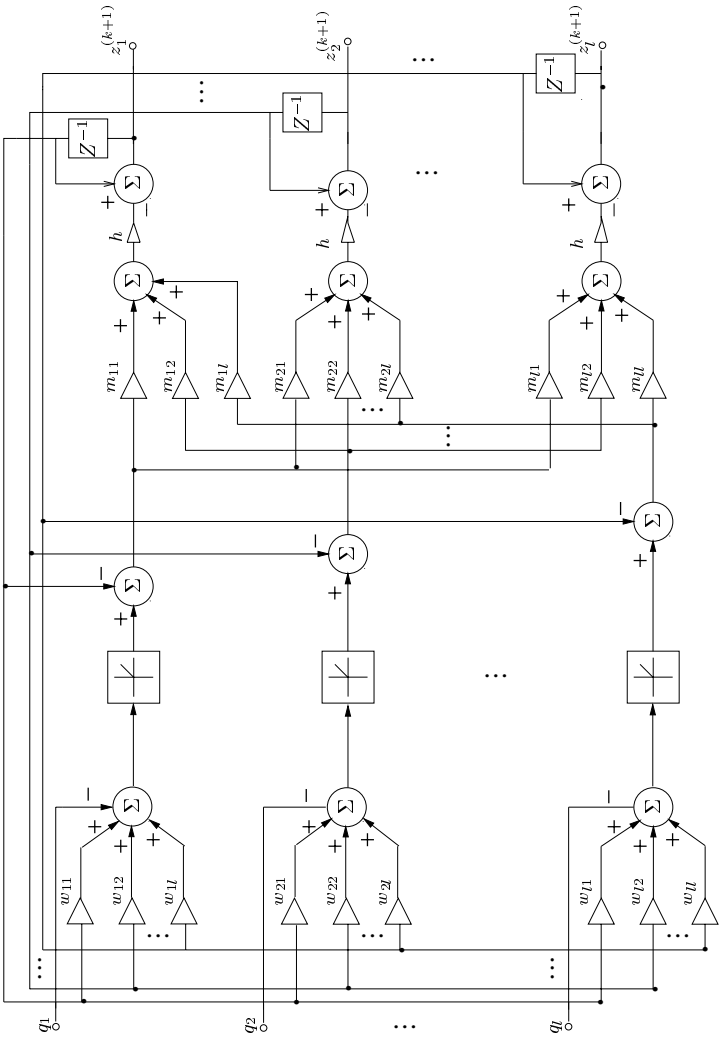


Figure 4. A block diagram of the neural network model in (19)

© 2001 by CRC Press LLC

Let

$$M = \begin{bmatrix} A & -D^T \\ D & 0 \end{bmatrix}, \quad q = \begin{bmatrix} c \\ -b \end{bmatrix}.$$

Then the problem (16) and (17) may become an LCP problem. Therefore, we discuss only discrete-time neural network for LCP.

Applying the Euler formula to the proposed neural network model

$$\frac{dz}{dt} = (I + M^T)((z - Mz - q)^+ - z),$$

we can get the following discrete-time neural network

$$z(k+1) = z(k) - hF(z(k)) \quad k = 1, 2, ... \tag{19}$$

where $F(z) = (I + M^T)((z - Mz - q)^+ - z)$ and $h > 0$ is a fixed design parameter. Figure 4 shows its digital implementation.

For (19) we have the following theoretical result.

**Theorem 4.** If $h < \frac{2}{\|I+M^T\|^2}$ , then sequence $\{z(k)\}$ generated by the discrete-time network of (19) is globally convergent to an optimal solution of the problem (16) and (17).

**Proof.** From the proof of the paper [Solodov and Tseng, 1996] we see that when $h = 2\|F(z(k))\|^{-2}\|r(z(k))\|^2$, where $r(z) = ((z - Mz - q)^+ - z)$, the sequence $\{z(k)\}$ generated by the discrete-time network of (19) is globally convergent to an optimal solution of the problem (16) and (17). Since

$$\|(I + M^T)r\|^2 \leq \|I + M^T\|^2\|r\|^2,$$

$$\frac{2}{\|I + M^T\|^2} \leq 2\|F(z(k))\|^{-2}\|r(z(k))\|^2.$$

Thus the conclusion of Theorem 4 holds.

## B. PRIMAL-DUAL NEURAL NETWORK FOR LINEAR ASSIGNMENT

### 1. Problem Formulation

The assignment problem can be formulated as the following zero-one integer linear program:

$$\text{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}, \tag{20}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, 2, \ldots, n; \tag{21}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, \ldots, n; \tag{22}$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, 2, \ldots, n. \tag{23}$$

© 2001 by CRC Press LLC

where $c_{ij}$ and $x_{ij}$ are respectively the cost coefficient and decision variable associated with assigning entity $i$ to position $j$. In general, a cost coefficient can be positive representing a loss or negative representing a gain. The decision variable is defined such that $x_{ij} = 1$ if and only if entity $i$ is assigned to position $j$. The objective function (20) to be minimized is the total cost for the assignment. Constraint (21) ensures that exactly one entity is assigned to each position; i.e., each column of $x_{ij}$ has only one decision variable being 1. Constraint (22) ensures that each entity is assigned to exactly one position; i.e., each row of $x_{ij}$ has only one decision variable being 1. Constraint (23) is the zero-one integrality constraint on decision variables. The assignment problem has a unique solution for almost all the cost coefficient matrix $[c_{ij}]$, and thus the uniqueness of the solution of (20)-(23) is assumed throughout this paper. It is well known, from the optimal solution point of view, that if the optimal solution is unique, then the assignment problem is equivalent to a linear programming problem by replacing the zero-one integrality constraints (23) with nonnegativity constraints, due to the total unimodularity property [Bazaraa et al., 1990]:

$$x_{ij} \geq 0, \quad i, j = 1, 2, \ldots, n. \tag{24}$$

We can therefore obtain the solution of (20)-(23) by solving its equivalent linear program which has also a unique solution. Based on the primal assignment problem, the dual assignment problem can be formulated as follows:

$$\text{maximize} \qquad \sum_{i=1}^{n} (u_i + v_i) \tag{25}$$

$$\text{subject to} \qquad u_i + v_j \leq c_{ij},\ i, j = 1, 2, \ldots, n; \tag{26}$$

where $u_i$ and $v_i$ denote the dual decision variables. The number of decision variables and inequality constraints in the dual assignment problem is $2n$ and $n^2$, respectively. According to the duality theorem in optimization theory [Bazaraa et al., 1990], the value of the objective function at its maximum is equal to the total cost of the primal assignment problem at its minimum; i.e., no duality gap.

**2. Neural Network Models**

In this section, we discuss the existing primal-dual assignment networks [Wang and Xia, 1998] for solving the primal and dual assignment problems.

Consider the following energy function:

$$\begin{aligned} E(x,u,v) &= \frac{1}{2}\left\{\sum_{i=1}^{n}\left[\sum_{j=1}^{n} c_{ij}x_{ij} - (u_i + v_i)\right]\right\}^2 + \frac{1}{2}\sum_{i=1}^{n}\left[\sum_{j=1}^{n} x_{ij} - 1\right]^2 \\ &+ \frac{1}{2}\sum_{j=1}^{n}\left[\sum_{i=1}^{n} x_{ij} - 1\right]^2 + \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_{ij}^2 - |x_{ij}|x_{ij}) \\ &+ \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n}\left[c_{ij} - (u_i + v_j)\right]\left[c_{ij} - (u_i + v_j) - |c_{ij} - (u_i + v_j)|\right] \end{aligned} \tag{27}$$

© 2001 by CRC Press LLC

where $x = (x_{11}, \cdots, x_{ij}, \cdots, x_{nn})^T \in R^{n^2}$, $v = (v_1, \cdots, v_i, \cdots, v_n)^T \in R^n$, and $u = (u_1, \cdots, u_i, \cdots, u_n)^T \in R^n$. The first term in eqn. (27) is the squared duality gap, the second and third terms are respectively for the equality constraints (21) and (22) and the fourth term is for the nonnegativity constraint (24) in the primal assignment problem, the last term is for the inequality constraint (26) in the dual assignment problem. Clearly, the function $E(x, u, v)$ is continuously differentiable, convex, and nonnegative on $R^{n^2+2n}$. By the duality theorem, $x^*$ and $(u^*, v^*)$ are optimal solutions respectively to the primal problem and the dual problem if and only if $E(x^*, u^*, v^*) = 0$.

The continuous-time version of the primal-dual assignment network is tailored from the ones for general linear programming [Xia, 1996]. If we let the time derivative of a state variable equal the partial derivative of the energy function defined in eqn. (27) with respect to the state variable, the dynamic equation of the continuous-time primal-dual assignment network is defined in the following differential equations: for $i, j = 1, 2, \ldots, n$;

$$\begin{aligned}
\frac{dx_{ij}}{dt} &= -\mu\{c_{ij}\sum_{p=1}^{n}(\sum_{q=1}^{n} c_{pq}x_{pq} - u_p - v_p) + \frac{1}{2}(x_{ij} - |x_{ij}|) \\
&+ \sum_{l=1}^{n}(x_{il} + x_{lj}) - 2\} \qquad (28) \\
\frac{du_i}{dt} &= -\mu\{-\sum_{p=1}^{n}(\sum_{q=1}^{n} c_{pq}x_{pq} - u_p - v_p) - \frac{1}{2}\sum_{l=1}^{n}(c_{il} - u_i - v_l \\
&- |c_{il} - u_i - v_l|)\} \qquad (29) \\
\frac{dv_i}{dt} &= -\mu\{-\sum_{p=1}^{n}(\sum_{q=1}^{n} c_{pq}x_{pq} - u_p - v_p) - \frac{1}{2}\sum_{l=1}^{n}(c_{li} - u_l - v_i \\
&- |c_{li} - u_l - v_i|)\} \qquad (30)
\end{aligned}$$

where $\mu > 0$ is a design parameter which scales the convergence rate of the continuous-time assignment network.

Since $s - |s| = 2\min\{0, s\}$ and $\max\{0, s\} = -\min\{0, -s\}$, the above equations can be rewritten as follows:

$$\begin{aligned}
\frac{dx_{ij}}{dt} &= -\mu\{c_{ij}\sum_{p=1}^{n}(\sum_{q=1}^{n} c_{pq}x_{pq} - u_p - v_p) + (x_{ij})^- \\
&+ \sum_{l=1}^{n}(x_{il} + x_{lj}) - 2\} \qquad (31) \\
\frac{du_i}{dt} &= -\mu\{-\sum_{p=1}^{n}(\sum_{q=1}^{n} c_{pq}x_{pq} - u_p - v_p) + \sum_{l=1}^{n}(u_i + v_l - c_{il})^+\} \qquad (32) \\
\frac{dv_i}{dt} &= -\mu\{-\sum_{p=1}^{n}(\sum_{q=1}^{n} c_{pq}x_{pq} - u_p - v_p) + \sum_{l=1}^{n}(u_l + v_i - c_{li})^+\}, \qquad (33)
\end{aligned}$$

© 2001 by CRC Press LLC

where $(s)^+ = \max\{0, s\}$ and $(s)^- = \min\{0, s\}$.

Based also on the energy function (27), the discrete-time version of the primal-dual assignment network is defined in the following difference equations: for $i, j = 1, 2, \cdots, n$;

$$\begin{aligned} x_{ij}^{(k+1)} &= x_{ij}^{(k)} - h\{c_{ij} \sum_{p=1}^{n} (\sum_{q=1}^{n} c_{pq} x_{pq}^{(k)} - u_p^{(k)} - v_p^{(k)}) + (x_{ij}^{(k)})^- \\ &+ \sum_{l=1}^{n} (x_{il}^{(k)} + x_{lj}^{(k)}) - 2\} \end{aligned} \tag{34}$$

$$\begin{aligned} u_i^{(k+1)} &= u_i^{(k)} + h\{\sum_{p=1}^{n} (\sum_{q=1}^{n} c_{pq} x_{pq}^{(k)} - u_p^{(k)} - v_p^{(k)}) \\ &- h \sum_{l=1}^{n} (u_i^{(k)} + v_l^{(k)} - c_{il})^+\} \end{aligned} \tag{35}$$

$$\begin{aligned} v_i^{(k+1)} &= v_i^{(k)} + h\{\sum_{p=1}^{n} (\sum_{q=1}^{n} c_{pq} x_{pq}^{(k)} - u_p^{(k)} - v_p^{(k)}) \\ &- h \sum_{l=1}^{n} (u_l^{(k)} + v_i^{(k)} - c_{li})^+\}, \end{aligned} \tag{36}$$

where $h > 0$ is a design parameter to be given. For convenience, let

$$\xi^{(k)} = \sum_{p=1}^{n} (\sum_{q=1}^{n} c_{pq} x_{pq}^{(k)} - u_p^{(k)} - v_p^{(k)}), \tag{37}$$

$$y_{ij}^{(k)} = \sum_{l=1}^{n} (x_{il}^{(k)} + x_{lj}^{(k)}), \tag{38}$$

$$\gamma_i^{(k)} = \sum_{l=1}^{n} (u_i^{(k)} + v_l^{(k)} - c_{il})^+, \tag{39}$$

$$\delta_i^{(k)} = \sum_{l=1}^{n} (u_l^{(k)} + v_i^{(k)} - c_{li})^+. \tag{40}$$

Then eqns. (34)-(36) can be rewritten as: for $i, j = 1, 2, \cdots, n$;

$$x_{ij}^{(k+1)} = x_{ij}^{(k)} - h \left[\xi^{(k)} c_{ij} + (x_{ij}^{(k)})^- + y_{ij}^{(k)} - 2\right] \tag{41}$$

$$u_i^{(k+1)} = u_i^{(k)} + h \left[\xi^{(k)} - \gamma_i^{(k)}\right] \tag{42}$$

$$v_i^{(k+1)} = v_i^{(k)} + h \left[\xi^{(k)} - \delta_i^{(k)}\right] \tag{43}$$

Figure 5 illustrates the architectures of the primal-dual assignment networks defined in auxiliary equations (37)-(40) (Figure 5(a)), differential equations (31-(33) (Figure 5(b)), and difference equations (22)-(24) (Figure 5(c)). It shows that

© 2001 by CRC Press LLC

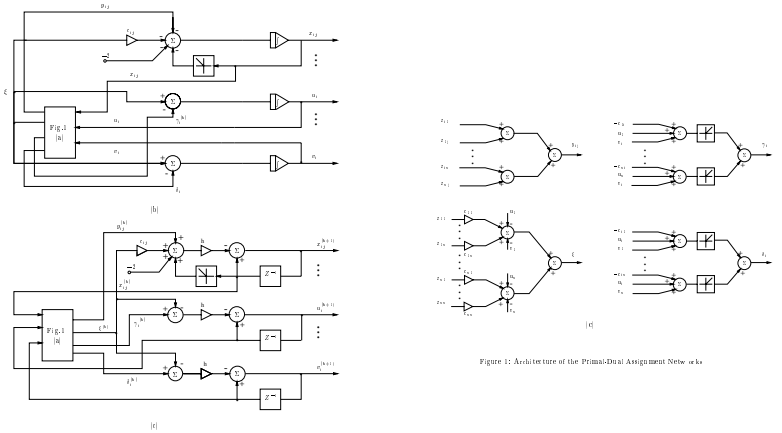

Figure 5. Architectures of the primal-dual assignment networks (Wang, J. and Xia, Y., Analysis and Design of Primal-Dual Assignment Networks, *IEEE Transactions on Neural Networks*, ©1998 IEEE)

the primal-dual assignment networks are composed of a number of adders, limiters, and time delays or integrators only.

**3. Global Convergence**

The global convergence property of the continuous-time counterpart is proven in Xia [1996]. In this section, we shall show that the discrete-time assignment network with a constant design parameter is globally convergent to an exact solution to the primal-dual assignment problem. First, some lemmas are needed to be introduced.

**Lemma 3**. Let $\varphi(x) = \frac{1}{2}(x^2 - |x|x), x \in R$. Then $\varphi'(x) = x - |x|$. For any $x, y \in R$,

$$\varphi(x) \leq \varphi(y) + \varphi'(y)(x - y) + (x - y)^2. \tag{44}$$

**Proof**. Consider two cases as follows.

(i) For any $y \geq 0, \varphi(y) = 0$ and $\varphi'(y) = 0$. Thus,

a) if $x \geq 0, \varphi(x) \leq (x - y)^2$, so (44) holds;

b) if $x < 0, \varphi(x) = x^2 \leq (x - y)^2$, so (44) also holds.

(ii) For $y < 0$, $y^2 + 2y(x - y) + (x - y)^2 = x^2$, so (44) holds.

**Lemma 4**. Let $\varphi(x) = \frac{1}{2}[(a_0x + b_0)^2 - |a_0x + b_0|(a_0x + b_0)], x \in R$. For

© 2001 by CRC Press LLC

any $x, y \in R$,

$$\varphi(x) \leq \varphi(y) + \varphi'(y)(x-y) + a_0^2(x-y)^2. \tag{45}$$

**Proof**. Let $z_1 = a_0 x + b_0, z_2 = a_0 y + b_0$, then $\varphi(x) = \varphi_1(z_1) = \frac{1}{2}(z_1^2 - |z_1|z_1)$ and $\varphi(y) = \varphi_1(z_2) = \frac{1}{2}(z_2^2 - |z_2|z_2)$. Thus, by Lemma 3 we have

$$\varphi(x) = \varphi_1(z_1) \leq \varphi_1(z_2) + \varphi_1'(z_2)(z_1 - z_2) + (z_1 - z_2)^2.$$

Hence

$$\varphi(x) \leq \varphi(y) + \varphi'(y)(x-y) + a_0^2(x-y)^2.$$

**Lemma 5**. Let $\Phi(x) = \frac{1}{2}x^T(x - |x|), x \in R^n$. For any $x, y \in R^n$,

$$\Phi(x) \leq \Phi(y) + \nabla\Phi(y)^T(x-y) + \|x-y\|_2^2. \tag{46}$$

where $\nabla\Phi(y)$ is the gradient of $\Phi(y)$.

**Proof**. Note that

$$\Phi(x) = \frac{1}{2}\sum_{i=1}^{n} x_i(x_i - |x_i|).$$

Then $\forall x, y \in R^n$, by Lemma 1 we have

$$\Phi(x) \leq \sum_{i=1}^{n}\left[\frac{1}{2}y_i(y_i - |y_i|) + (y_i - |y_i|)(x_i - y_i) + (x_i - y_i)^2\right].$$

Thus we have (46).

**Lemma 6**. Let $\Phi(y) = \frac{1}{2}(c - A^T y)^T\{(c - A^T y) - |c - A^T y|)\}$, where $A \in R^{n\times m}$ and $y \in R^m$. For any $y, z \in R^m$,

$$\Phi(z) \leq \Phi(y) + \nabla\Phi(y)^T(z-y) + (z-y)^T AA^T(z-y). \tag{47}$$

**Proof**. By Lemma 4 and Lemma 5, we have (47).

**Lemma 7**. Let $w = (x, u, v)^T, w' = (x', u', v')^T \in R^{n^2+2n}$, and $E(x, u, v)$ be defined in Section III. For any $w, w'$

$$E(w) \leq E(w') + \nabla E(w')^T(w - w') + (w - w')^T H(w - w'), \tag{48}$$

where

$$H = \begin{bmatrix} A^T A + cc^T + I & -cb^T \\ -bc^T & AA^T + bb^T \end{bmatrix},$$

$c = [c_{11}, c_{12}, \ldots, c_{1n}, c_{21}, c_{22}, \ldots, c_{2n}, \ldots, c_{n1}, c_{n2}, \ldots, c_{nn})^T, b = (1, \cdots, 1)^T \in R^{2n}$, and $A$ is the $2n \times n^2$ constraint matrix in the assignment problem whose $(i, j)$ column is $e_i + e_{n+j}$, $e_p$ is a vector in $R^{2n}$ with the $p$-th element being 1 and others being 0, for $i, j, p = 1, \cdots, n$.

© 2001 by CRC Press LLC

**Proof**. Let

$$E_1(w) = \frac{1}{2}\left\{\sum_{i=1}^{n}\left[\sum_{j=1}^{n} c_{ij}x_{ij} - (u_i + v_i)\right]\right\}^2 + \frac{1}{2}\left\{\sum_{i=1}^{n}\left[\sum_{j=1}^{n} x_{ij} - 1\right]^2 + \sum_{j=1}^{n}\left[\sum_{i=1}^{n} x_{ij} - 1\right]^2\right\}$$

and

$$E_2(w) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_{ij}^2 - |x_{ij}|x_{ij}) + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left[c_{ij} - (u_i + v_j)\right]\left[c_{ij} - (u_i + v_j) - |c_{ij} - (u_i + v_j)|\right].$$

By using the second-order Taylor formula, we have

$$E_1(w) = E_1(w') + \nabla E_1(w')^T(w - w') + (w - w')^T\nabla^2 E_1(w')(w - w'),$$

where

$$\nabla^2 E_1(w') = \begin{bmatrix} cc^T + A^TA & -cb^T \\ -bc^T & bb^T \end{bmatrix}.$$

In addition, we see from Lemma 3 and Lemma 4 that

$$E_2(w) \le E_2(w') + \nabla E_2(w')^T(w - w') + (w - w')^T H_1(w - w').$$

where

$$H_1 = \begin{bmatrix} I & 0 \\ 0 & AA^T \end{bmatrix},$$

Since $E(w) = E_1(w) + E_2(w)$, we obtain (48).

**Lemma 8**. $E(w)$ defined in Lemma 4 is continuously differentiable and convex on $R^{n^2+2n}$, and for $\forall w, w' \in R^{n^2+2n}$

$$(w - w')^T\nabla E(w') \le E(w) - E(w').$$

**Proof**. See Ortega [1970].

**Lemma 9**. Let $A$ be defined in Lemma 7. The maximum eigenvalue of $AA^T$ is $2n$.

**Proof**. Since $A$ is a $2n \times n^2$ matrix whose $(i, j)$ column is $e_i + e_{j+n}$ for $i, j = 1, 2, \ldots, n$, then

$$AA^T = \sum_{i=1}^{n}\sum_{j=1}^{n}(e_i + e_{j+n})(e_i + e_{j+n})^T$$

© 2001 by CRC Press LLC

$$
\begin{aligned}
&= \sum_{i=1}^{n}\sum_{j=1}^{n}(e_i e_i^T + e_{j+n}e_{j+n}^T + e_i e_{j+n}^T + e_{j+n}e_i^T) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}(e_i e_i^T + e_{j+n}e_{j+n}^T) + \sum_{i=1}^{n}\sum_{j=1}^{n}(e_i e_{j+n}^T + e_{j+n}e_i^T) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}(e_i e_i^T + e_{j+n}e_{j+n}^T) + \sum_{i=1}^{n}\sum_{j=1}^{n}(e_i e_{j+n}^T + e_{j+n}e_i^T).
\end{aligned}
$$

It is easy to see that

$$
\begin{aligned}
\sum_{i=1}^{n}\sum_{j=1}^{n}(e_i e_i^T + e_{j+n}e_{j+n}^T) &= n\sum_{i=1}^{n} e_i e_i^T + n\sum_{i=n+1}^{2n} e_{i+n}e_{i+n}^T \\
&= 2n\sum_{i=1}^{2n} e_i e_i^T = 2nI_{2n\times 2n},
\end{aligned}
$$

where $I_{2n\times 2n}$ is a $2n \times 2n$ identity matrix. In addition, we have

$$
\sum_{i=1}^{n}\sum_{j=1}^{n}(e_i e_{j+n}^T + e_{j+n}e_i^T) = \sum_{i=1}^{n} e_i \sum_{j=1}^{n} e_{j+n}^T + \sum_{j=1}^{n} e_{j+n}\sum_{i=1}^{n} e_i^T.
$$

Let $\eta_1 = \sum_{i=1}^{n} e_i$ and $\eta_2 = \sum_{j=1}^{n} e_{j+n}$, then $\eta_1^T\eta_2 = \eta_2^T\eta_1 = 0$ and $\eta_1^T\eta_1 = \eta_2^T\eta_2 = n$. Thus $(\eta_1\eta_2^T + \eta_2\eta_1^T)(\eta_1 + \eta_2) = n(\eta_1 + \eta_2)$. Hence $n$ is an eigenvalue of $\eta_1\eta_2^T + \eta_2\eta_1^T$.

Assuming that $n$ is not the maximum eigenvalue of $\eta_1\eta_2^T + \eta_2\eta_1^T$, then there exists $\epsilon > 0$ such that

$$
\begin{pmatrix} 0 & U \\ U & 0 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = (n+\epsilon)\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}
$$

where $v_1, v_2 \in R^n$ and

$$
U = \begin{pmatrix} 1 & 1 & \ldots, 1 \\ 1 & 1 & \ldots, 1 \\ \vdots & \vdots & \ddots, \vdots \\ 1 & 1 & \ldots, 1 \end{pmatrix}_{n\times n}.
$$

Thus

$$
\begin{aligned}
Uv_1 &= (n+\epsilon)v_2 \\
Uv_2 &= (n+\epsilon)v_1,
\end{aligned}
$$

and $U^2v_2 = (n+\epsilon)Uv_1 = (n+\epsilon)^2v_2$. Since the maximum eigenvalue of $U^2$ is $n^2$,

$$
n^2 < (n+\epsilon)^2 \le n^2,
$$

© 2001 by CRC Press LLC

which is contradictory. Therefore, the maximum eigenvalue of $\eta_1\eta_2^T + \eta_2\eta_1^T$ is $n$ and thus the maximum eigenvalue of $AA^T$ is $2n$. The proof is complete.

Using the above lemmas, we shall establish the result of global convergence for the discrete-time assignment network.

**Theorem 5**. If $h < \frac{1}{2\lambda_H}$ where $\lambda_H$ is a maximum eigenvalue of the matrix $H$ defined in Lemma 7, then sequence $\{x^{(k)}, u^{(k)}, v^{(k)}\}$ generated by the discrete-time assignment network is globally convergent to an optimal solution of the primal-dual assignment problem.

**Proof**. Let $\{w^{(k)}\} = \{x^{(k)}, u^{(k)}, v^{(k)}\}$. First, by Lemma 7 we have $E(w^{(k+1)}) \leq E(w^{(k)}) + \nabla E(w^{(k)})^T(w^{(k+1)} - w^{(k)}) + (w^{(k+1)} - w^{(k)})^T H(w^{(k+1)} - w^{(k)})$. Since $w^{(k+1)} - w^{(k)} = -h\nabla E(w^{(k)})$,

$$0 \leq E(w^{(k)}) + \nabla E(w^{(k)})^T(-h\nabla E(w^{(k)})) + h^2\nabla E(w^{(k)})^T H\nabla E(w^{(k)}),$$

hence

$$-E(w^{(k)}) \leq -h\|\nabla E(w^{(k)})\|_2^2 + h^2\lambda_H\|\nabla E(w^{(k)})\|_2^2.$$

Therefore, from Lemma 8 we obtain

$$\begin{array}{rcl} \|w^{(k+1)} - w^*\|_2^2 & = & \|w^{(k)} - w^*\|_2^2 - 2h(w^{(k)} - w^*)^T\nabla E(w^{(k)}) \\ & + & h^2\|\nabla E(w^{(k)})\|_2^2 \\ & \leq & \|w^{(k)} - w^*\|_2^2 - 2h^2E(w^{(k)}) + h^2\|\nabla E(w^{(k)})\|_2^2 \\ & \leq & \|w^{(k)} - w^*\|_2^2 - 2h^2\|\nabla E(w^{(k)})\|_2^2 \\ & + & 2h^3\lambda_H\|\nabla E(w^{(k)})\|_2^2 + h^2\|E(w^{(k)})\|_2^2 \\ & \leq & \|w^{(k)} - w^*\|_2^2 - h^2\|\nabla E(w^{(k)})\|_2^2(1 - 2h\lambda_H) \end{array}$$

where $w^* = (x^*, u^*, v^*)^T$ is an optimal solution to the primal and dual assignment problem. Since

$$H = \begin{bmatrix} A^TA + I & 0 \\ 0 & AA^T \end{bmatrix} + \begin{bmatrix} c \\ -b \end{bmatrix}[c^T, -b^T],$$

and $H$ is a symmetric positive semi-definite matrix, thus $\lambda_H > 0$. So, when $\nabla E(w^{(k)}) \neq 0$ (i.e., $w^{(k)}$ is not an optimal solution), it follows that

$$\|w^{(k+1)} - w^*\|_2 < \|w^{(k)} - w^*\|_2. \tag{49}$$

Thus, $\{w^{(k)}\}$ is bounded. On the other hand, using the above second inequality we have

$$h^2\|\nabla E(w^{(k)})\|_2^2(1 - 2h\lambda_H) \leq \|w^{(k)} - w^*\|_2^2 - \|w^{(k+1)} - w^*\|_2^2,$$

so

$$\sum_{k=1}^{\infty}\|\nabla E(w^{(k)})\|_2^2 < +\infty.$$

© 2001 by CRC Press LLC

Thus $\lim_{k\longrightarrow\infty}\|\nabla E(w^{(k)})\|_2 = 0$. Since $\{w^{(k)}\}$ is bounded, there is a sequence $\{k_i\}$ such that

$$\lim_{i\longrightarrow\infty} w^{(k_i)} = \hat{w}.$$

Then

$$\lim_{i\longrightarrow\infty}\|\nabla E(w^{(k_i)})\|_2 = \|\nabla E(\hat{w})\|_2 = 0,$$

and thus $E(\hat{w}) = 0$. So $\hat{w} = (\hat{x}, \hat{u}, \hat{v})^T$ is an optimal solution to the primal and dual assignment problem. In view of (30), the sequence $\{w^{(k)}\}$ has only one limit point, so

$$\lim_{k\longrightarrow\infty} w^{(k)} = \hat{w}.$$

The above analytical result shows that the constant design parameter of the discrete-time assignment network is bounded in a small range. The following theorem will illustrate that for any fixed initial step $h_0$ there is a number $l > 0$ such that $\frac{1}{2^l}h_0 < \frac{1}{\lambda_H}$, and hence the assignment network is definitely convergent to optimal solution to the primal-dual assignment problem in finite steps.

**Theorem 6**. If $h < 1/[2(4n + \|c\|_2^2 + 1)]$, then sequence $\{w^{(k)}\}$ generated by the discrete-time assignment network is globally convergent to an optimal solution of the primal-dual assignment problem.

**Proof**. Since

$$H = \begin{bmatrix} A^TA + I & 0 \\ 0 & AA^T \end{bmatrix} + \begin{bmatrix} c \\ -b \end{bmatrix}[c^T, -b^T],$$

and $H$ is the sum of two symmetric matrices, according to Courant Fischer Min-max Theorem [Wang, 1995],

$$\lambda_H \leq \lambda + \|c\|_2^2 + 2n$$

where $\lambda$ is the maximum eigenvalue of

$$\begin{bmatrix} A^TA + I & 0 \\ 0 & AA^T \end{bmatrix}.$$

In view that $A^TA$ and $AA^T$ have the same nonzero eigenvalues, according to Lemma 9, we have $\lambda = 2n + 1$. Hence

$$\lambda_H \leq 4n + \|c\|_2^2 + 1.$$

Then from Theorem 5, we can complete the proof.

## IV. SIMULATION RESULTS

In order to demonstrate the effectiveness and efficiency of the proposed neural networks, in this section, we discuss the simulation results through four examples.

© 2001 by CRC Press LLC

The simulation is conducted on matlab, the ordinary differential equation solver engaged is ode45s.

Example 1. Consider the following quadratic program (the equivalent to the one in Kennedy and Chua [1988]) with the only optimal solution $x^* = (5.0, 5.0)$:

$$\begin{aligned} \text{Minimize} \quad & x_1^2 + x_2^2 + x_1 x_2 - 30x_1 - 30x_2 \\ \text{subject to} \quad & \frac{5}{12}x_1 - x_2 \le \frac{35}{12}, \\ & \frac{5}{2}x_1 + x_2 \le \frac{35}{2}, \\ & -x_1 \le 5, \\ & x_2 \le 5. \end{aligned}$$

This problem is equivalent to

$$\begin{aligned} \text{Minimize} \quad & x_1^2 + x_2^2 + x_1 x_2 - 30x_1 - 30x_2 \\ \text{subject to} \quad & \frac{5}{12}x_1 - x_2 + x_3 = \frac{35}{12}, \\ & \frac{5}{2}x_1 + x_2 + x_4 = \frac{35}{2}, \\ & -5 \le x_1 \le 7, -5 \le x_2 \le 5, 0 \le x_3 \le 10, 0 \le x_4 \le 35. \end{aligned}$$

We use the system (11) to solve the above problem. Figure 6 shows the transient behavior of the primal-dual network which globally converges to the optimal solution.

Example 2. Consider the convex quadratic program (16) where

$$D = \begin{bmatrix} -5/12 & 1 \\ -5/2 & -1 \\ 1 & 0 \\ 0 & -1 \end{bmatrix}, A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, b = \begin{bmatrix} -35/12 \\ -35/2 \\ -5 \\ -5 \end{bmatrix}, c = \begin{bmatrix} -30 \\ -30 \end{bmatrix}$$

Its exact solution is $(5, 5)^T$. We use the systems (15) and (19) to solve the above problem. All simulation results show that the solution trajectory always converges to unique point $z^* = (5.000, 5.000, 0, 6.000, 0, 9.000)^T$ which corresponds to the optimal solution $(5, 5)^T$. For example, let $B = 5I$, and starting point is $(-10, 10, 0, 0, 0, 0)^T$. Figure 7 shows the transient behavior of the continuous-time neural network for this starting point. In the discrete-time case, taking step size $h = 0.08$, respective trajectories of the initial point $(-10, 10, 0, 0, 0, 0)^T$ are shown in Figure 8.

Example 3. Consider the classical linear complementary problem which was taken from Hertog [1994]. $M$ is an $10 \times 10$ upper triangular matrix

$$M = \begin{bmatrix} 1 & 2 & 2 & \dots & 2 \\ 0 & 1 & 2 & \dots & 2 \\ 0 & 0 & 1 & \dots & 2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

© 2001 by CRC Press LLC

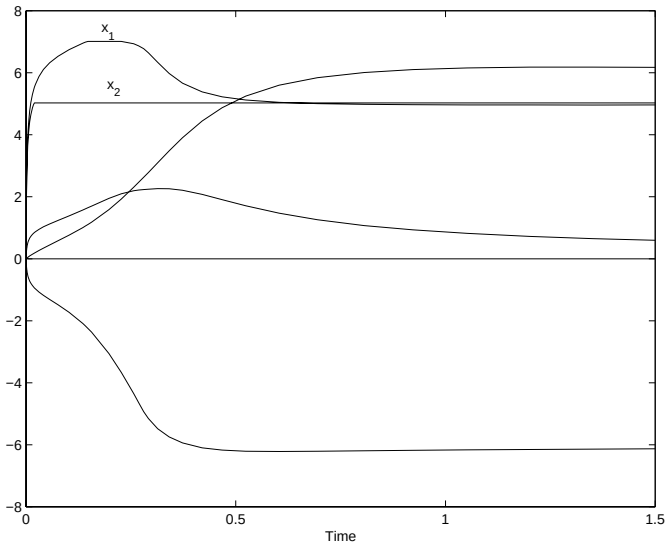


Figure 6. Transient behaviors of the primal-dual network of (14) in Example 1

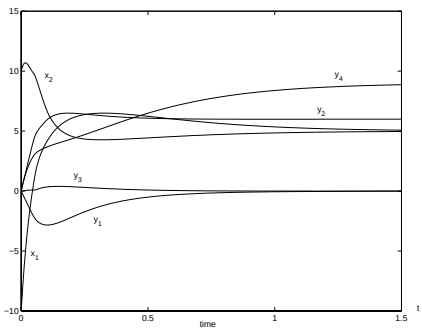


Figure 7. Transient behaviors of the primal-dual network of (15) in Example 2

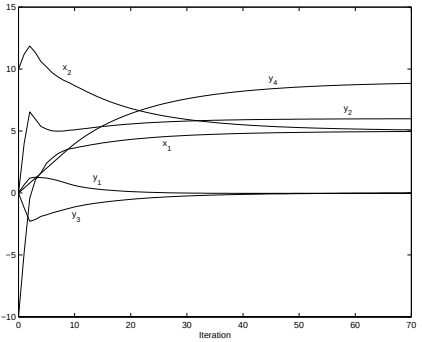


Figure 8. Transient behaviors of the primal-dual network of (19) in Example 2

© 2001 by CRC Press LLC

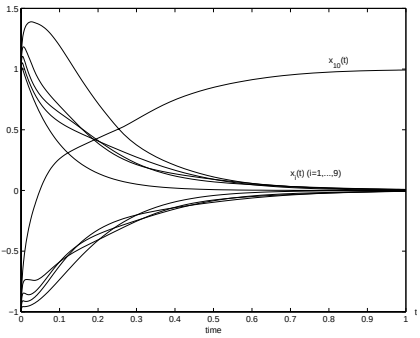


Figure 9. Transient behaviors of the recurrent neural network of (15) in Example 3

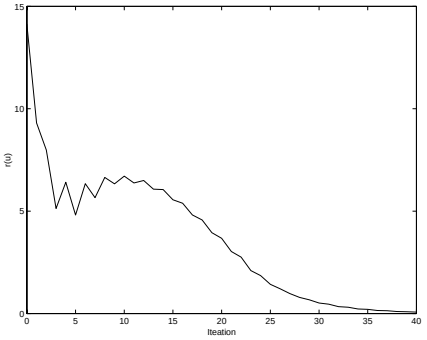


Figure 10. Transient behaviors of the recurrent neural network of (19) in Example 3

and $q = (-1, ..., -1)^T \in R^1 0$. The solution of the problem is $u^* = (0, ..., 0, 1)^T \in R^1 0$. We use the systems (15) and (19) to solve the above problem. The simulation results show that the trajectory of the system always globally converges to the solution $(0, ..., 1)^T$. For example, let $B = 5I$, and the initial point is $(1, -1, ..., 1, -1)^T \in R^{10}$, respectively. Figure 9 shows the transient behaviors of the continuous-time neural network for this initial point. In the discrete-time case, taking $n = 80$ and step size $h = 0.016$, then the error $r(u) = \|P_\Omega(u - \beta(Mu + q)) - u\|$ along the trajectory of the zero initial point is shown in Figure 10.

Example 4: Consider the sorting problem used in Wang [1995, 1997] (Example 1): rank 10 items {-1.3, 1.7, 0.5, 2.2, -2.6, 1.5, -0.6, 0.9, -1.2, 1.1} in ascending order. Let $s_j = 11 - j$ for $j = 1, 2, \ldots, 10$. Accordingly, the cost coefficient matrix can be defined as follows.

$$[c_{ij}] = \begin{pmatrix} -13.0 & -11.7 & -10.4 & -9.1 & -7.8 & -6.5 & -5.2 & -3.9 & -2.6 & -1.3 \\ 17.0 & 15.3 & 13.6 & 11.9 & 10.2 & 8.5 & 6.8 & 5.1 & 3.4 & 1.7 \\ 5.0 & 4.5 & 4.0 & 3.5 & 3.0 & 2.5 & 2.0 & 1.5 & 1.0 & 0.5 \\ 22.0 & 19.8 & 17.6 & 15.4 & 13.2 & 11.0 & 8.8 & 6.6 & 4.4 & 2.2 \\ -26.0 & -23.4 & -20.8 & -18.2 & -15.6 & -13.0 & -10.4 & -7.8 & -5.2 & -2.6 \\ 15.0 & 13.5 & 12.0 & 10.5 & 9.0 & 7.5 & 6.0 & 4.5 & 3.0 & 1.5 \\ -6.0 & -5.4 & -4.8 & -4.2 & -3.6 & -3.0 & -2.4 & -1.8 & -1.2 & -0.6 \\ 9.0 & 8.1 & 7.2 & 6.3 & 5.4 & 4.5 & 3.6 & 2.7 & 1.8 & 0.9 \\ -12.0 & -10.8 & -9.6 & -8.4 & -7.2 & -6.0 & -4.8 & -3.6 & -2.4 & -1.2 \\ 11.0 & 9.9 & 8.8 & 7.7 & 6.6 & 5.5 & 4.4 & 3.3 & 2.2 & 1.1 \end{pmatrix}.$$

The sorting problem can be formulated as the assignment problem [Wang, 1995]. The decision variable is defined as that $x_{ij} = 1$ if item $i$ with numerical key $r_i$ is in the $j$-th position of the sorted list. The cost coefficients of the assignment problem for sorting are defined as $c_{ij} = r_i s_j$ where $r_i$ and $s_j$ denote respectively the numerical key of the $i$-th item to be sorted and the nonzero weighting parameter for the $j$-th position in the desired list.

Simulations have been conducted with the initial values of all variables to be zero for both continuous-time and discrete-time assignment networks. Figure 11 illustrates the transient behavior of the energy function of the continuous-time assignment network with $\mu = 10^8$ and variable time-steps. Figure 12 depicts the transient behaviors of the discrete-time assignment network with three different

© 2001 by CRC Press LLC

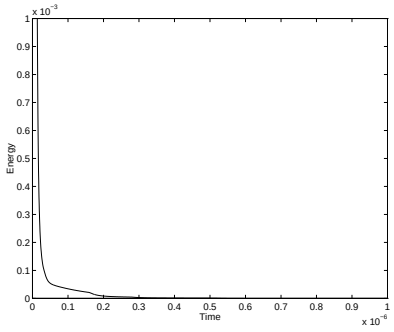


Figure 11. Transient behaviors of the continuous-time network of (14) in Example 4 (Wang, J. and Xia, Y., Analysis and design of primal-dual assignment networks, *IEEE Transactions on Neural Networks,* ©1998 IEEE)

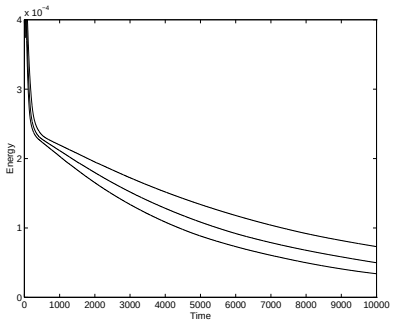


Figure 12. Transient behaviors of the discrete-time network of (14) in Example 4 (Wang, J. and Xia, Y., Analysis and design of primal-dual assignment networks, *IEEE Transactions on Neural Networks,* ©1998 IEEE)

step lengths $h$. Specifically, the line decreasing at the fastest rate corresponds to $h = 0.25$, the slowest line $h = 0.15$, in-between $h = 0.20$. Note that every step size $h > 1/(2\lambda_H)$. All the values of the energy function converge to zero.

## V. CONCLUDING REMARKS

Neural networks have been proposed for optimization in a variety of application areas such as design and layout of very large scale integrated (VLSI) circuits. The nature of parallel and distributed information processing makes recurrent neural networks viable for solving complex optimization problems in real time. One of the salient features of neural networks is their suitability for hardware implementation, in which the convergence rate is not increasing statistically as the size of the problem increases.

Although great progress has been made in using neural networks for optimization, many theoretical and practical problems remain unsolved. Many avenues are open for future work. For example, the existing neural networks have not yet been shown to be capable of solving nonconvex optimization problems. Neither could the existing neural networks be guaranteed to obtain the optimial solutions to NP-hard combinatorial optimization problems. Further investigations should aim at the indepth analysis of the dynamics of recurrent neural networks for solving nonconvex and discrete optimization problems, the wide applications of recurrent neural networks to practical problems for real-time design and planning, and the hardware prototyping of recurrent neural networks for optimization.

© 2001 by CRC Press LLC

## REFERENCES


Baiocchi, C. and Capelo, A., *Variational and Quasi-variational Inequalities: Applications to Free Boundary Problems*, John Wiley and Sons, New York, 1988.

Bazaraa, M. S., Jarvis, J. J., and Sherali, H. D., *Linear Programming and Network Flows* (2nd Ed.), John Wiley & Sons, New York, NY, 1990.

Bertsekas, D. P., *Constrained Optimization and Largrange Multiplier Methods*, Academic Press, New York, 1982.

Bouzerdoum, A. and Pattison, T. R., Neural network for quadratic optimization with bound constraints, *IEEE Transactions on Neural Networks*, 4, 293, 1993.

Chua, L. O. and Lin, G. N., Nonlinear Programming without Computation, *IEEE Transactions on Circuits and Systems*, 31, 182, 1984.

Cichocki, A. and Unbehauen, R., Switched-Capacitor Neural Networks for Differential Optimization, *International Journal of Circuit Theory and Applications*, 19, 161, 1991.

Cichocki, A. and Unbehauen, R., *Neural Networks for Optimization and Signal Processing*, John Wiley, New York, 1993.

Cottle, R. W., Pang, J.-S., and Stone, R. E., *The Linear Complementarity Problem.* Academic Press, Boston, 1992.

Dennis, J. B., *Mathematics Programming and Electrical Networks.* Chapman and Hall, London, 1959.

Gafni, E. M. and Bertsekas, D. P., Two-metric projection methods for constrained optimization, *SIAM J. Control and Optimization*, 22, 936, 1984.

Glazos, M. P., Hui, S., and Zak, S. H., Sliding models in solving convex programming problems, *SIAM J. Control and Optimization*, 36(2), 680, 1998.

Hertog, D., *Interior Point Approach to Linear, Quadratic and Convex Programming: Algorithms and Complexity,* Kluwer Academic Publishers, Boston, MA, 1994.

Hopfield, J. J. and Tank, D. W., Neural Computation of Decisions in Optimization Problems, *Biological Cybernetics*, 52(3), 141 ,1985.

Kinderlehrer, D. and Stampcchia, G., *An Introduction to Variational Inequalities and Their Applications*, Italy, Academic Press, 1980.


© 2001 by CRC Press LLC

Kennedy, M. P. and Chua, L. N., Unifying the Tank and Hopfield Linear Programming Circuit and the Canonical Nonlinear Programming Circuit of Chua and Lin, *IEEE Transactions on Circuits and Systems,* 34(2), 210, 1987.

Kennedy, M. D. and Chua, L. N., Neural networks for nonlinear programming, *IEEE Transactions on Circuits and Systems,* 35(5), 554, 1988.

Lillo, W. E., Loh, M. H., Hui, S., and Zăk, S. H., On solving constrained optimization problems with neural networks: A penalty method approach, *IEEE Transactions on Neural Networks,* 4(6), 931, 1993.

Maa, C. Y. and Shanblatt, M. A., A Two-Phase Optimization Neural Network, *IEEE Transactions on Neural Networks,* 3(6), 1003, 1992.

Maa, C. Y. and Shanblatt, M. A., Linear and quadratic programming neural network analysis, *IEEE Transactions on Neural Networks,* 3, 580, 1992.

Marcotte, P., Application of Khobotov's algorithm to variational inequalities and network equilibrium problems, *Information Systems in Operations Research,* 29, 258, 1991.

Miller R. K. and Michel, A. N., *Ordinary Differential Equations,* Academic Press, New York, 1982.

More, J. J. and Toroaldo, G., On the solution of large quadratic programming problems with bound constraints, *SIAM J. Optimization,* 1(1), 93, 1991.

Ortega, J. M. and Rheinboldt, W. G., *Iterative Solution of Nonlinear Equations in Several Variables,* Academic Press, New York, 1970.

Pang, J. S., A posteriori error bounds for the linearly-constrained variational inequality problem, *Mathematics of Operations Research,* 12, 474, 1987.

Pyne, I. B., Linear programming on an electronic analogue computer. *Transactions of the American Institute of Electrical Engineering,* 75, 139, 1956.

Rodríguez-Vázquez, A., Domínguez-Castro, R., Rueda, A., Huertas, J. L., Sánchez-Sinencio, E., Nonlinear switched-capacitor 'neural networks' for optimization problems, *IEEE Transactions on Circuits and Systems*, 37(3), 384, 1990.

Rybashov, M. V., The gradient method of solving convex programming problems on electronic analogue computers. *Automation Remote Control,* 26(11), 1886, 1965.

© 2001 by CRC Press LLC

Solodov, M. V. and Tseng, P., Modified projection-type methods for monotone variational inequalities, *SIAM J. Control and Optimization*, 2, 1814, 1996.

Sudharsanan, S. and Sundareshan, M., Exponential stability and a systematic synthesis of a neural network for quadratic minimization, *Neural Networks*, 4(5), 599, 1991.

Tank, W. D. and Hopfield, J. J., 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, 1986.

Vandenberghe, L., Moor, B. L., and Vandewalle, J., The generalized linear complementarity problem applied to the complete analysis of resistive piecewise-linear circuits, *IEEE Transactions on Circuits and Systems*, 36(11), 1391, 1989.

Wang, J., A deterministic annealing neural network for convex programming, *Neural Networks*, 7(4), 629, 1994.

Wang, J., Analysis and design of an analog sorting network, *IEEE Transactions on Neural Networks,* 6(4), 962, 1995.

Wang, J., Primal and dual assignment networks, *IEEE Transactions on Neural Networks,* 8, 784, 1997.

Wang, J. and Xia, Y., Analysis and Design of Primal-Dual Assignment Networks, *IEEE Transactions on Neural Networks,* 9, 183, 1998.

Wu, X., Xia, Y., Li,J., and Chen, W. K., A high performance neural network for solving linear and quadratic programming problems, *IEEE Transactions on Neural Networks*, 7, 643, 1996.

Xia, Y., A new neural network for solving linear programming problems and its applications," *IEEE Transactions on Neural Networks,* 7, 525, 1996.

Xia, Y., A new neural network for solving linear and quadratic programming problems, *IEEE Transactions on Neural Networks,* 7, 1544, 1996.

Xia, Y., Neural network for solving extended linear programming problems, *IEEE Transactions on Neural Networks,* 8, 525, 1997.

Xia Y. and Wang J., Neural network for solving linear programming problems with bounded variables, *IEEE Transactions on Neural Networks,* 6, 515, 1995.

Xia, Y. and Wang, J., A General method for designing optimization neural networks with global convergence, *IEEE Transactions on Neural Networks*, 9, 1998.

© 2001 by CRC Press LLC

Zhang, S. and Constantinides, A. G., Lagrange Programming Neural Networks, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39(7), 441, 1992.

Zhang, S., Zhu, X., and Zou, L. H., Second-Order Neural Networks for Constrained Optimization, *IEEE Transactions on Neural Networks*, 3(6), 1021, 1992.

© 2001 by CRC Press LLC