Brief Papers

Solving the Assignment Problem Using Continuous-Time and Discrete-Time Improved Dual Networks

Xiaolin Hu, *Member*, *IEEE*, and Jun Wang, *Fellow*, *IEEE*

***Abstract*— The assignment problem is an archetypal combinatorial optimization problem. In this brief, we present a continuous-time version and a discrete-time version of the improved dual neural network (IDNN) for solving the assignment problem. Compared with most assignment networks in the literature, the two versions of IDNNs are advantageous in circuit implementation due to their simple structures. Both of them are theoretically guaranteed to be globally convergent to a solution of the assignment problem if only the solution is unique.**

***Index Terms*— Analog circuits, assignment problem, linear programming, quadratic programming, sorting problem.**

I. Introduction

The assignment problem is concerned with assigning n entities to n slots for achieving minimum cost or maximum profit. It is known to be a polynomial combinatorial optimization problem. Its applications cover pattern classification, machine learning, operations research, and so on.

For solving the assignment problem, there exist many efficient iterative algorithms such as the Hungarian method, the auction method, and the signature method. Inspired by the Hopfield network for solving optimization problems, many artificial neural networks have been developed for solving the assignment problem (e.g., [1]–[4]). One of the major advantages of neural networks is that they can be implemented in parallel analog circuits to achieve super high speed, which is very attractive in real-time applications. However, most of these methods require an "annealing" procedure which is sensitive to the solution quality. In addition, this time-varying procedure would pose difficulties in circuit implementation.

Manuscript received May 23, 2011; revised February 3, 2012; accepted February 5, 2012. Date of publication February 21, 2012; date of current version May 2, 2012. The work of X. Hu was supported in part by the National Natural Science Foundation of China, under Grant 60805023 and Grant 61134012, the National Basic Research Program (973 Program) of China, under Grant 2012CB316301, and the Basic Research Foundation of Tsinghua National Laboratory for Information Science and Technology. The work of J. Wang was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Grant CUHK417209E and Grant CUHK417608E.

X. Hu is with the State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology, and Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: xlhu@tsinghua.edu.cn).

J. Wang is with the Department of Mechanical and Automation Engineering, Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: jwang@mae.cuhk.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2012.2187798

In [5], an assignment neural network is proposed without any time-varying procedure. It features elegant theoretical results with constant parameters. Its major demerit lies in its complex structure. For instance, it entails more neurons and interconnections than the network in [4]. But this is a tradeoff between the efficiency and structural complexity, as argued in [5]. In this brief, we show that this tradeoff is unnecessary. Based on the *improved dual neural network* (IDNN) proposed in [6], we design a very simple assignment network with constant parameters only.

The IDNN is an efficient neural network solver for a special class of quadratic programming (QP) problems, and it has been successfully applied to solve the k-winners-take-all (k-WTA) problem [6]. The k-WTA network was then extended to encompass discontinuous activation functions [7], [8]. The original IDNN and its k-WTA extensions [7], [8] are continuous-time solvers, which could be implemented on analog circuits. Analog circuits can have extremely fast computing capability but they are not as precise or robust as digital circuits [9]. With the rapid development of digital technologies, discrete-time neural networks seem to be a good compromise between speed and scalability. In [10], a continuous-time assignment network proposed in [2] is discretized and implemented on digital hardware, which achieves good results. In [11], a discrete-time k-WTA network is proposed based on the IDNN. In this brief, in addition to a continuous-time assignment network based on the IDNN, we also present a discrete-time counterpart.

II. Problem Formulation

Suppose that there are n entities to be assigned to n slots and assigning entity i to slot j induces a cost c_{ij}. Then, what is the best assignment in terms of minimum total cost? This is a typical *assignment problem*. Mathematically, it can be formulated as a linear zero–one programming problem (LZOP, for short)

$$\begin{aligned} \text{minimize} \quad & f_1 = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1,\ldots,n \\ & \sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1,\ldots,n \\ & x_{ij} \in \{0,1\} \qquad \forall i,j = 1,\ldots,n \end{aligned} \tag{1}$$

where c_{ij} and x_{ij} are, respectively, the cost variable and the decision variable associated with assigning entity i to slot j. The variable $x_{ij} = 1$ means assigning entity i to slot j, and $x_{ij} = 0$ means not assigning entity i to slot j. Since any

entity should be, and must be, assigned to only one slot, and any slot should be, and must be, assigned one entity, a feasible assignment should correspond to a matrix $\mathbf{x} = \{x_{ij}\}$ with only one element equal to 1 in every column and row.

Lemma 1: The LZOP problem is equivalent to the following quadratic zero–one programming (QZOP) problem:

$$\begin{aligned} \text{minimize} \quad & f_2 = \frac{q}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}^2 + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1,\ldots,n \\ & \sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1,\ldots,n \\ & x_{ij} \in \{0,1\} \qquad \forall i,j = 1,\ldots,n \end{aligned} \tag{2}$$

where $q > 0$ is a constant.

Proof: This can be proved by showing that $\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}^2$ is a constant in the feasible region. Because $x_{ij} \in \{0,1\}$, $x_{ij} = x_{ij}^2$. Then $\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} = n$, which completes the proof. ∎

It is known that if the LZOP problem has a unique solution, it is equivalent to a linear programming (LP) problem by replacing the binary constraints with the intervals $x_{ij} \in [0,1], \forall i, j = 1,\ldots,n$, which is called the LP problem hereafter. In what follows, we formulate the continuous counterpart of the QZOP problem.

Theorem 1: If the QZOP problem has a unique solution, then there exists a sufficiently small positive constant q such that the QZOP problem is equivalent to a QP problem by replacing the binary constraints with the intervals $x_{ij} \in [0,1], \forall i, j = 1,\ldots,n.$, which is called the QP problem hereafter.

Proof: What is needed to show is only that the unique solution of the QZOP problem, denoted by $\mathbf{x}^*$, is also a solution of the QP problem because the objective function of the QP is strictly convex and it has at most one solution.

Denote the feasible region of the QP problem by $\mathcal{X}$ and the feasible region of the QZOP problem by $\mathcal{V}$. Note that any point in $\mathcal{X}$ is called a *doubly stochastic matrix* and any point in $\mathcal{V}$ is called a *permutation matrix*. According to the well-known Birkhoff–von Neumann theorem, a square matrix is doubly stochastic if and only if it is a convex combination of permutation matrices. Namely, any $\mathbf{x} \in \mathcal{X}$ can be expressed as $\mathbf{x} = \sum_k \theta_k \mathbf{v}^{(k)}$, where $\mathbf{v}^{(k)} \in \mathcal{V}$, $\sum_k \theta_k = 1, 0 \le \theta_k \le 1$. Denote the unique solution of the QZOP problem by $\mathbf{x}^*$. Then

$$\begin{aligned} \langle \mathbf{x}^*, \mathbf{x}^* - \mathbf{v}^{(k)} \rangle &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}^* \left(x_{ij}^* - v_{ij}^{(k)} \right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} (x_{ij}^*)^2 - x_{ij}^* v_{ij}^{(k)} \\ &= n - \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}^* v_{ij}^{(k)} > 0 \end{aligned}$$

for $\mathbf{v}^{(k)} \in \mathcal{V}, \mathbf{v}^{(k)} \neq \mathbf{x}^*$, where $\langle \cdot, \cdot \rangle$ stands for the Frobenius inner product of two matrices. Because $\mathbf{x}^*$ is also the unique solution of the LP problem, according to the equivalence between convex optimization problem and variational inequality [12], we have

$$\langle \nabla f_1(\mathbf{x}^*), \mathbf{x} - \mathbf{x}^* \rangle = \langle \mathbf{c}, \mathbf{x} - \mathbf{x}^* \rangle \ge 0 \qquad \forall \mathbf{x} \in \mathcal{X}.$$

Now we show that for any $\mathbf{x} \in \mathcal{X}$ but $\mathbf{x} \neq \mathbf{x}^*$, the last equality above cannot hold. Otherwise, there exists such a point $\bar{\mathbf{x}}$ so that $\langle \mathbf{c}, \bar{\mathbf{x}} - \mathbf{x}^* \rangle = 0$. Then $\langle \mathbf{c}, \mathbf{x} - \mathbf{x}^* \rangle = \langle \mathbf{c}, \mathbf{x} - \bar{\mathbf{x}} \rangle \ge 0, \forall \mathbf{x} \in \mathcal{X}$, indicating that $\bar{\mathbf{x}} \neq \mathbf{x}^*$ is also a solution of the LP problem, which contradicts the uniqueness of the solution. Hence

$$\langle \mathbf{c}, \mathbf{x} - \mathbf{x}^* \rangle > 0 \qquad \forall \mathbf{x} \in \mathcal{X}, \mathbf{x} \neq \mathbf{x}^*.$$

Consequently

$$\langle \mathbf{c}, \mathbf{v}^{(k)} - \mathbf{x}^* \rangle > 0 \qquad \forall \mathbf{v}^{(k)} \in \mathcal{V}, \mathbf{v}^{(k)} \neq \mathbf{x}^*$$

as $\mathcal{V} \subset \mathcal{X}$. Let

$$0 < q \le \frac{\langle \mathbf{c}, \mathbf{v}^{(k)} - \mathbf{x}^* \rangle}{\langle \mathbf{x}^*, \mathbf{x}^* - \mathbf{v}^{(k)} \rangle} \qquad \forall \mathbf{v}^{(k)} \in \mathcal{V}, \mathbf{v}^{(k)} \neq \mathbf{x}^*.$$

It follows that

$$\begin{aligned} \langle \nabla f_2(\mathbf{x}^*), \mathbf{v}^{(k)} - \mathbf{x}^* \rangle &= \langle q\mathbf{x}^* + \mathbf{c}, \mathbf{v}^{(k)} - \mathbf{x}^* \rangle \ge 0 \\ & \qquad \forall \mathbf{v}^{(k)} \in \mathcal{V}, \mathbf{v}^{(k)} \neq \mathbf{x}^*. \end{aligned}$$

For any $\mathbf{x} \in \mathcal{X}$

$$\begin{aligned} \langle \nabla f_2(\mathbf{x}^*), \mathbf{x} - \mathbf{x}^* \rangle &= \left\langle \nabla f_2(\mathbf{x}^*), \sum_k \theta_k \mathbf{v}^{(k)} - \mathbf{x}^* \right\rangle \\ &= \sum_k \theta_k \langle \nabla f_2(\mathbf{x}^*), \mathbf{v}^{(k)} - \mathbf{x}^* \rangle \ge 0 \end{aligned}$$

where $0 \le \theta_k \le 1, \sum_k \theta_k = 1$. Hence, $\mathbf{x}^*$ is a solution of the QP problem, which completes the proof. ∎

From the proof, it can be seen that q should be small enough and its upper bound is

$$\min_k \frac{\langle \mathbf{c}, \mathbf{v}^{(k)} - \mathbf{x}^* \rangle}{\langle \mathbf{x}^*, \mathbf{x}^* - \mathbf{v}^{(k)} \rangle} \tag{3}$$

which is positive. In practice, the optimal solution $\mathbf{x}^*$ is unknown, and it is time consuming to find all feasible solutions $\mathbf{v}^{(k)}$ to the LZOP problem or the QZOP problem. So, it is suggested to set q to be very small. Throughout the rest of this brief, it is assumed that the solution of the assignment problem (1) is unique. Then, with small q, all the four problems, namely, LZOP, QZOP, LP, and QP, are equivalent.

III. Two Networks

A. Architecture

We rewrite the QP problem in the vector form as

$$\begin{aligned} \text{minimize} \quad & f_2 = \frac{1}{2}\|\mathbf{y}\|^2 + \mathbf{p}^T\mathbf{y} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{y} = \mathbf{1}_{2n}, \quad \mathbf{0}_{n^2} \le \mathbf{y} \le \mathbf{1}_{n^2} \end{aligned}$$

where $\mathbf{p}$ and $\mathbf{y}$ are vectors by stacking the first column to the last column of the matrix $\mathbf{c}/q = \{c_{ij}/q\}$ and $\mathbf{x} = \{x_{ij}\}$, respectively, $\mathbf{1}$ and $\mathbf{0}$ are two vectors with all elements equal

to 1 and 0 (the dimensions are indicated by the subscripts), respectively, and

$$\mathbf{A} = \begin{pmatrix} \mathbf{I}_n & \mathbf{I}_n & \cdots & \mathbf{I}_n \\ \mathbf{1}_n^T & \mathbf{0}_n^T & \cdots & \mathbf{0}_n^T \\ \mathbf{0}_n^T & \mathbf{1}_n^T & \cdots & \mathbf{0}_n^T \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}_n^T & \mathbf{0}_n^T & \cdots & \mathbf{1}_n^T \end{pmatrix}$$

with $\mathbf{I}_n$ standing for the $n \times n$ dimensional identity matrix. Throughout this brief, $\|\cdot\|$ stands for the l_2 norm of a vector. According to [6], the problem can be solved by the following IDNN:

$$\tau \frac{d\mathbf{z}}{dt} = -\mathbf{A}\tilde{\mathbf{y}} + \mathbf{1}_{2n} \tag{4}$$

where $\tau > 0$ is a constant, $\tilde{\mathbf{y}} = \mathbf{g}(\mathbf{A}^T\mathbf{z} - \mathbf{p})$, and $\mathbf{g} = (g(s), \ldots, g(s))^T$. $g(s)$ is the activation function with saturation, which is equal to s if $0 \le s \le 1$, 0 if $s < 0$, and 1 otherwise. The scalar form of the network is as follows:

1) state equations

$$\tau \frac{du_i}{dt} = -\sum_{j=1}^{n} g(u_i + v_j - c_{ij}/q) + 1, \qquad i = 1, \ldots, n$$

$$\tau \frac{dv_i}{dt} = -\sum_{j=1}^{n} g(u_j + v_i - c_{ji}/q) + 1, \qquad i = 1, \ldots, n;$$

2) output equations

$$x_{ij} = g(u_i + v_j - c_{ij}), \quad i, j = 1, \ldots, n.$$

Clearly, this network would entail n neurons for representing u_i and n neurons for representing v_i.

Next, we propose a discrete-time version of the IDNN for solving the assignment problem

$$\mathbf{z}(k+1) = \mathbf{z}(k) - \beta(\mathbf{A}\tilde{\mathbf{y}}(k) - \mathbf{1}_{2n}) \tag{5}$$

where $\tilde{\mathbf{y}}(k) = \mathbf{g}(\mathbf{A}^T\mathbf{z}(k) - \mathbf{p})$. The scalar form is as follows:

1) state equations

$$u_i(k+1) = u_i(k) - \beta\left(\sum_{j=1}^{n} g(u_i(k) + v_j(k) - \frac{c_{ij}}{q}) - 1\right), \qquad i = 1, \ldots, n$$

$$v_i(k+1) = v_i(k) - \beta\left(\sum_{j=1}^{n} g(u_j(k) + v_i(k) - \frac{c_{ji}}{q}) - 1\right), \qquad i = 1, \ldots, n;$$

2) output equations

$$x_{ij}(k) = g(u_i(k) + v_j(k) - c_{ij}), \quad i, j = 1, \ldots, n$$

where k labels the steps and $\beta > 0$ is the step size. Also, this network would entail n neurons for representing u_i and n neurons for representing v_i.

The two networks share the same overall structure, as illustrated in Fig. 1, though concrete implementation entails different techniques, i.e., analog versus digital circuit techniques. The block diagram for realizing the neuron u_i or v_i can be found in [13].

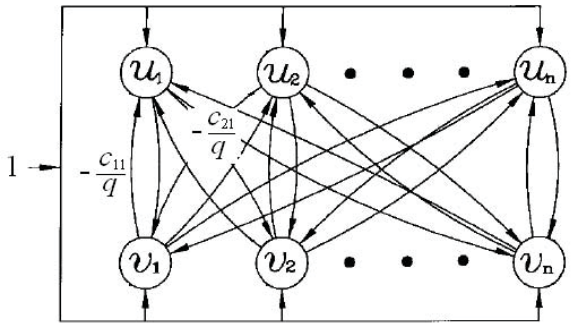


Fig. 1. Abstract structure of the assignment networks (4) and (5).

TABLE I
MODEL COMPARISON

	Neurons	Time-varying param	Activation fun
Primal NN [4]	n^2	Yes	Continous
Dual NN [4]	$2n$	Yes	Continuous
PDNN [5]	$n^2 + 2n$	No	Continuous
Heaviside NN [14]	$2n$	No	Discontinuous
IDNN in this brief	$2n$	No	Continuous

B. Comparison With Existing Neural Networks

In [4], two neural networks are proposed for solving the assignment problem (1). One is called the *primal neural network*, and the other is called the *dual neural network*. It is seen that the primal neural network entails n^2 neurons and $O(n^2)$ connections, whereas the dual neural network entails $2n$ neurons and $O(n^2)$ connections (see [4] for details). The latter model is simpler in structure, nearly as simple as the IDNN (4). But both of them involve a temperature parameter $\alpha \exp(-t/T)$, which is hard to realize in hardware.

In [5], a continuous-time version and a discrete-time version of the *primal–dual neural network* are proposed for solving the assignment problem. They entail constant parameters only, which is superior to the networks in [4], but both of them consist of $n^2 + 2n$ neurons and $O(n^2)$ connections.

Recently, an assignment neural network with the Heaviside step activation function was proposed with $2n$ neurons [14]. In addition, it does not need the parameter q. But because the activation function is discontinuous, its implementation entails different techniques, which might be harder compared with implementing a continuous activation function.

Since the assignment problem can be equivalently written as an LP or QP problem, all LP networks and QP networks are capable of solving it, e.g., [15], [16]. However, it is easy to see that all of them require $n^2 + 2n$ neurons for solving the assignment problem.

The properties of some typical networks discussed above are summarized in Table I. It is seen that the IDNNs (both the continuous- and discrete-time versions) presented in this brief are among the best.

IV. Stability and Convergence Results

A. Preliminaries

Let $\mathcal{R}^n$ denote the n-dimensional real space and $\mathcal{Z}_+$ denote the set of nonnegative integers.

Definition 1:

1) The *equilibrium point* of (4) is a point $\mathbf{z}^* \in \mathcal{R}^{2n}$ such that $\mathbf{Ag}(\mathbf{A}^T\mathbf{z}^* - \mathbf{p}) = \mathbf{1}_{2n}$.
2) The equilibrium point of (4) is *stable in the sense of Lyapunov* if for all $\epsilon > 0$ there exists $\delta = \delta(\epsilon)$ such that if $\|\mathbf{z}(0)\| < \delta$, then $\|\mathbf{z}(t)\| < \epsilon, t \geq 0$.
3) A continuous trajectory $\mathbf{s}(t) \in \mathcal{R}^n$ of (4) is *globally convergent* to the set $\mathcal{S} \subset \mathcal{R}^n$ if it exists for all $t \in [0, +\infty)$ and $\lim_{t\to+\infty} \|\mathbf{s}(t) - \bar{\mathbf{s}}\| = 0$, where $\bar{\mathbf{s}} \in \mathcal{S} \subset \mathcal{R}^n$.

All of the above definitions apply to the discrete-time IDNN (5) by substituting t with k, and $t > 0$ with $k \in \mathcal{Z}_+$.

It is seen that the equilibrium sets of the neural networks (4) and (5) are the same, which is denoted by $\mathcal{E}$ hereafter.

Lemma 2 [12]: For any $\mathbf{x}, \mathbf{y} \in \mathcal{R}^n$

$$(\mathbf{x} - \mathbf{y})^T(\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})) \geq \|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})\|.$$

The following result follows from [6, Th. 2] and Theorem 1 directly.

Lemma 3: If q is sufficiently small [the upper bound is indicated in (3)], the unique solution of the assignment problem (1), denoted by $\mathbf{y}^*$, can be expressed as $\mathbf{y}^* = \mathbf{g}(\mathbf{A}^T\mathbf{z}^* - \mathbf{p})$, where $\mathbf{z}^* \in \mathcal{E}$.

B. Main Results

Theorem 2: If q is sufficiently small, then any equilibrium point of the continuous-time network (4) is stable in the sense of Lyapunov, and the corresponding output trajectory globally converges to the unique solution of the assignment problem (1).

Proof: Since it is assumed that the assignment problem has a unique solution, according to Lemma 3, the unique solution corresponds to the equilibrium point of (4). Because (4) is a special case of the network proposed in [6], the results follow from [6, Th. 4] directly. ∎

In what follows, we analyze the stability and convergence result of the discrete-time network (5). A procedure similar to that in [11] is adopted.

Theorem 3: Any equilibrium point of the discrete-time network (5) is stable in the sense of Lyapunov, and the corresponding output trajectory globally converges to the unique solution of the assignment problem (1), if q is sufficiently small and the step size $\beta < 2/\lambda_{\max}$, where $\lambda_{\max}$ denotes the maximum eigenvalue of $\mathbf{A}^T\mathbf{A}$.

Proof: First of all, it is seen that $\lambda_{\max} > 0$ because $\mathbf{A}^T\mathbf{A}$ is positive semidefinite but not equal to zero. Since the right-hand side of the dynamic system (5) is defined over the entire $\mathcal{R}^{2n}$, there exists a unique solution $\mathbf{z}(k)$ with the initial point $\mathbf{z}_0$ as k increases from 0 to $+\infty$ [17].

Let $\mathbf{z}^*$ denote the equilibrium point of (5). Then, according to Lemma 3, $\mathbf{y}^* = \mathbf{g}(\mathbf{A}^T\mathbf{z}^* - \mathbf{p})$ and $\mathbf{Ay}^* - \mathbf{1}_{2n} = \mathbf{0}_{2n}$. Define a function $V(k) = \|\mathbf{z}(k) - \mathbf{z}^*\|^2$. Since $V(k)$ is radially unbounded and

$$\begin{aligned}
&V(k+1) - V(k)\\
&= \|\mathbf{z}(k+1) - \mathbf{z}^*\|^2 - \|\mathbf{z}(k) - \mathbf{z}^*\|^2\\
&= \|\mathbf{z}(k) - \beta(\mathbf{A}\tilde{\mathbf{y}}(k) - \mathbf{1}_{2n}) - \mathbf{z}^* + \beta(\mathbf{Ay}^* - \mathbf{1}_{2n})\|^2\\
&\quad - \|\mathbf{z}(k) - \mathbf{z}^*\|^2\\
&= \|\mathbf{z}(k) - \mathbf{z}^* - \beta\mathbf{A}(\tilde{\mathbf{y}}(k) - \mathbf{y}^*)\|^2 - \|\mathbf{z}(k) - \mathbf{z}^*\|^2\\
&= -2\beta(\mathbf{z}(k) - \mathbf{z}^*)^T\mathbf{A}(\tilde{\mathbf{y}}(k) - \mathbf{y}^*) + \beta^2\|\mathbf{A}(\tilde{\mathbf{y}}(k) - \mathbf{y}^*)\|^2\\
&= -2\beta((\mathbf{A}^T\mathbf{z}(k) - \mathbf{p}) - (\mathbf{A}^T\mathbf{z}^* - p))^T(\tilde{\mathbf{y}}(k) - \mathbf{y}^*)\\
&\quad + \beta^2\|\mathbf{A}(\tilde{\mathbf{y}}(k) - \mathbf{y}^*)\|^2\\
&\leq -2\beta\|\tilde{\mathbf{y}}(k) - \mathbf{y}^*\|^2 + \beta^2\lambda_{\max}\|\tilde{\mathbf{y}}(k) - \mathbf{y}^*\|^2\\
&= \beta(\beta\lambda_{\max} - 2)\|\tilde{\mathbf{y}}(k) - \mathbf{y}^*\|^2 \leq 0
\end{aligned}$$

the equilibrium point $\mathbf{z}^*$ is stable in the sense of Lyapunov [17]. In above equation, Lemma 2 is used. In addition, any level set $\{\mathbf{z}|V(k) \leq l\}$ is bounded, where $l > 0$. So, $\{\mathbf{z}|V(k) \leq V(0)\}$ is bounded. It follows from the LaSalle's invariance principle for discrete-time dynamic systems that every state trajectory $\mathbf{z}(k)$ tends to the largest invariant set contained in

$$\mathcal{K} = \{\mathbf{z} \in \mathcal{R}^{2n} \mid \|\tilde{\mathbf{y}}(k) - \mathbf{y}^*\| = 0\}$$

as $k \to +\infty$ [17]. Clearly, the largest invariant set in $\mathcal{K}$ is $\mathcal{K}$ itself here, because every point in this set will make $\mathbf{z}(k+1) = \mathbf{z}(k)$. On the other hand, every equilibrium point must be a member of $\mathcal{K}$. Hence, $\mathcal{K} = \mathcal{E}$. Then the state trajectory $\mathbf{z}(k)$ of the network globally converges to $\mathcal{E}$. From Lemma 3, the output trajectory $\tilde{\mathbf{y}}(k)$ globally converges to the unique solution of the assignment problem. ∎

Note that the upper bound of the step size indicated in Theorem 3 is conservative, and in practice larger step sizes can be used to accelerate the convergence. See Section V.

In very large-scale integrated implementation, time delays may become a critical problem for stability [18]–[22]. But the analysis with time delays is out of the scope of this brief.

V. Simulation Results

To verify the results presented in the last section, we numerically simulated the proposed networks (4) and (5) for solving some problems in MATLAB.

Example 1: For generating an assignment problem (1), what one needs to set is only the cost matrix $\mathbf{c}$. We randomly generated some $n \times n$ cost matrices with every element between 0 and 1. Fig. 2(a) and (b) shows the state trajectories of the continuous-time network (4) and the discrete-time network (5), respectively, for solving an assignment problem with $n = 10$. The parameters were set as follows: $\tau = 10^{-6}$, $\beta = 1.9/\lambda_{\max}(\mathbf{A}^T\mathbf{A})$, and $q = 0.001$. The outputs of the two networks converge to the same point, which is the correct solution of the assignment problem, verified by solving the LP problem with the MATLAB build-in function "linprog." The two networks start from the same initial point. It is seen that their trajectories look similar. In fact, this is the case in most simulations.

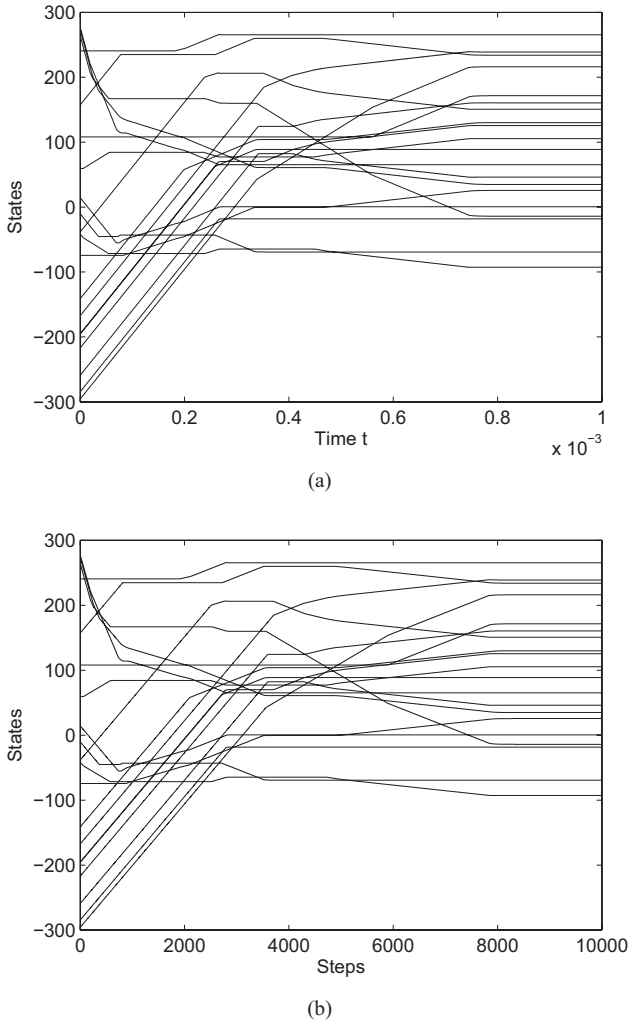


Fig. 2. State trajectories of (a) network (4) and (b) network (5).

In Theorem 1, q is given an upper bound. But it is unclear how small is enough. We investigated this issue numerically. Only the continuous-time network (with $\tau = 10^{-6}$) was simulated, as the discrete-time network has similar dynamic behaviors. The idea was to compare the convergence time of the network with different q values. Let $\mathbf{x}^*$ be the *ground truth* obtained by MATLAB function "linprog." For saving CPU time in running these simulations, we terminated the program when

$$\sum_i \sum_j |\text{round}(x_{ij}(t)) - x_{ij}^*| \leq 10^{-6}$$

where round(x) is equal to 0 if $x < 0.5$ and 1 otherwise. The time at which this was achieved was recorded as the convergence time. For every q value, 30 different runs with random initial points in $[-50, 50]$ were executed. The statistics of the convergence time for three problem sizes is plotted in Fig. 3. It is seen that, as q decreases, the convergence time increases. Therefore, q should not be too small. This poses some difficulty in choosing an appropriate q.

Next, we investigated the relationship between the problem size and the convergence time of the network (4). Let $q = 0.01/80$, and select several $n \leq 80$. For each n, 30 different runs with random initial points in [–50, 50] were executed. The same stopping criterion was adopted as before. The statistics of the convergence time with $\tau = 10^{-6}$ is plotted in Fig. 4(a). It is seen that the convergence time decreases as n increases. This is due to the parallel nature of the network. If q is allowed to vary with the problem size, e.g., $q = 0.01/n$, then the convergence time will be reduced [see Fig. 4(b)].

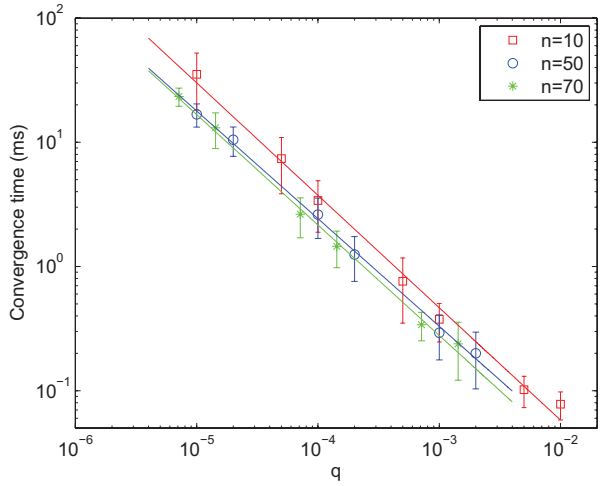
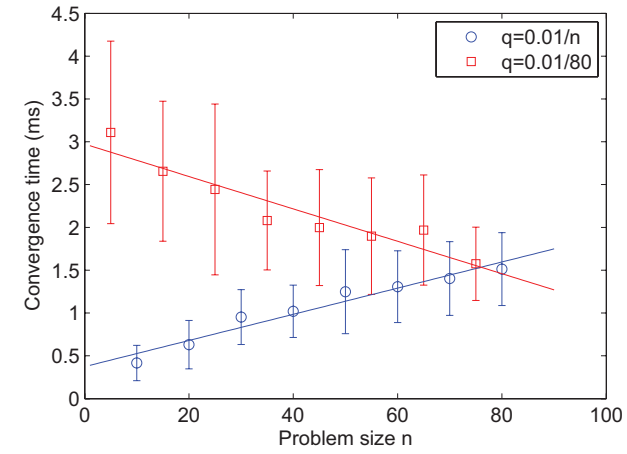


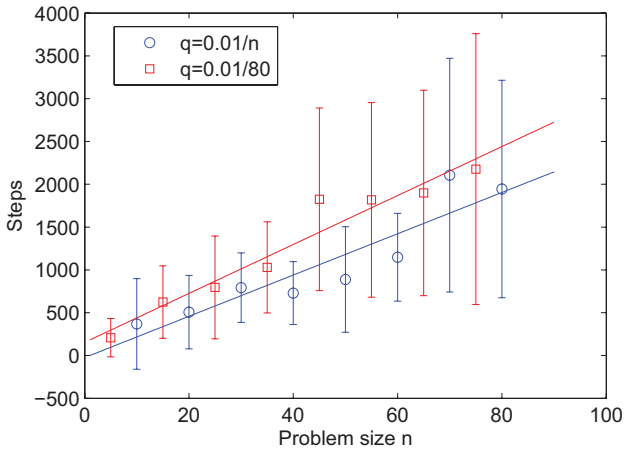
Fig. 3. Convergence time of (4) with different q values. Note that the coordinates are in logarithmic scale. The squares, circles, and asterisks denote the means for $n = 10$, 50, and 70, respectively, and the bars below and above them stand for 1 standard deviation each. The continuous lines are linear regressions of the logarithm of the means versus the logarithm of q.

Simulating the discrete-time network (5) is time consuming because the step size indicated in Theorem 3 is rather conservative while there is no intelligence for choosing the appropriate step sizes as the MATLAB code simulators by which we have simulated (4). Here we introduce a simple rule. Let $\beta_0 = 1.9/\lambda_{\max}(\mathbf{A}^T\mathbf{A})$ and $\beta = \alpha\beta_0$. Initially, $\alpha = 1000$. Every 500 updates, we check $\|\mathbf{A}\tilde{\mathbf{y}}(k) - \mathbf{1}_{2n}\|_1$. If this quantity has not decreased over 20%, then $\alpha = \alpha/10$. This process continues until $\alpha = 1$. In addition, when $\alpha = 1$, after every 500 updates, it has 50% percent chance to jump to 1000 (this is to perturb the system to have a better starting point which may lead to faster convergence). The stopping criterion was adopted the same as above. With this rule, the steps needed for convergence were significantly reduced, and the statistics is presented in Fig. 4(b). It is seen that the convergence steps increase approximately linearly with the problem size no matter whether q is fixed or varied. Anyway, the problem-size-dependent q has resulted in faster convergence.

Example 2: Sorting is a fundamental operation accounting for 25% of data processing time [23]. It is desirable to develop some fast and robust hardware units for doing this job. It is known that it is equivalent to an assignment problem in the form of (1) with $c_{ij} = r_i s_j$, where r_i denotes the value of item i in the unsorted sequence and s_j denotes the nonzero weighting parameter for jth position in the sorted sequence (e.g., if $s_1 > s_2 > \cdots$, then the desired order is from small to large). The decision variable $x_{ij} = 1$ if item i with value r_i is in the jth position of the sorted sequence.



(a)

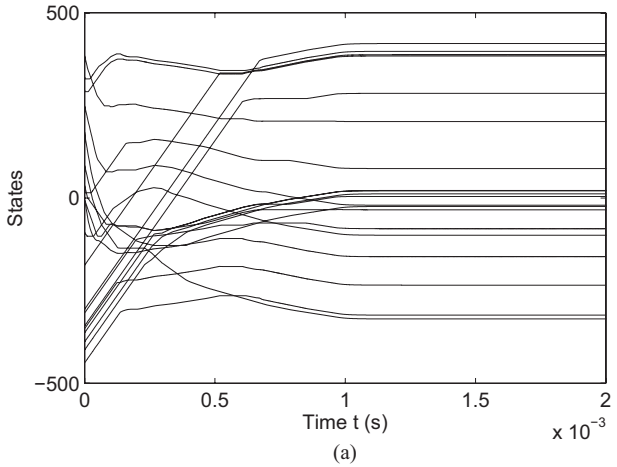


(b)

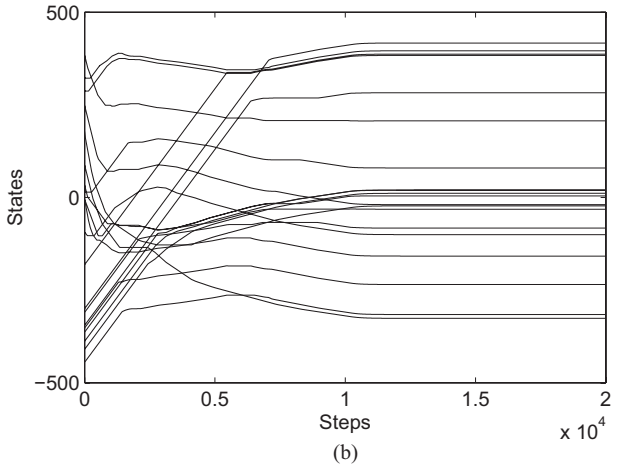
Fig. 4. Convergence time of (a) network (4) and (b) network (5) for different problem sizes. The circles and bars denote the means and the standard deviations, respectively. The continuous lines are the linear regressions of the means versus n.

Consider sorting the sequence $\{1.4, 3.5, -3.5, 5, 10, -1, 0, 8, 7.5, 7.6\}$. Let $r_i = 11 - i$ for $i = 1, 2, \ldots, 10$, which implies sorting from small to large. Let $q = 0.1$, $\tau = 10^{-6}$, $\beta = 1.9/\lambda_{\max}(\mathbf{A}^T\mathbf{A})$ and simulate the two networks (4) and (5) (without the acceleration rule) to solve the problem. Fig. 5(a) and (b), respectively, demonstrate the transient behavior of the two networks with a random initial point. Again, it is seen that they look very similar. The corresponding final outputs are both

$$\begin{pmatrix} 0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1 \\ 0&1&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0 \end{pmatrix}$$



(a)



(b)

Fig. 5. State trajectories of (a) network (4) and (b) network (5) for solving the sorting problem.

which is the correct solution. The sorted sequence is $\{r_3, r_6, r_7, r_1, r_2, r_4, r_9, r_{10}, r_8, r_5\}$.

VI. Conclusion

In this brief, we applied an IDNN for solving the problem and obtained two assignment networks, a continuous-time version and a discrete-time version, with $2n$ neurons and without time-varying parameters. Both networks are theoretically guaranteed to be globally convergent to the solution of the problem if only the problem has a unique solution.

The main disadvantage of the proposed assignment networks is that they introduce a parameter that should be set appropriately. If it was too large, the network may converge to incorrect states, if it was too small, the convergence will be slow. Unfortunately, at the current stage, it has to be selected by trial and error.

References

[1] S. P. Eberhardt, T. Daud, D. A. Kerns, T. X. Brown, and A. P. Thakoor, "Competitive neural architecture for hardware solution to the assignment problem," *Neural Netw.*, vol. 4, no. 4, pp. 431–442, 1991.

[2] J. Wang, "Analogue neural network for solving the assignment problem," *Electron. Lett.*, vol. 28, no. 11, pp. 1047–1050, May 1992.

[3] J. J. Kosowsky and A. L. Yuille, "The invisible hand algorithm: Solving the assignment problem with statistical physics," *Neural Netw.*, vol. 7, no. 3, pp. 477–490, 1994.

[4] J. Wang, "Primal and dual assignment networks," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 784–790, May 1997.

[5] J. Wang and Y. Xia, "Analysis and design of primal-dual assignment networks," *IEEE Trans. Neural Netw.*, vol. 9, no. 1, pp. 183–194, Jan. 1998.

[6] X. Hu and J. Wang, "An improved dual neural network for solving a class of quadratic programming problems and its k-winners-take-all application," *IEEE Trans. Neural Netw.*, vol. 19, no. 12, pp. 2022–2031, Dec. 2008.

[7] Q. Liu, C. Dang, and J. Cao, "A novel recurrent neural network with one neuron and finite-time convergence for k-winners-take-all operation," *IEEE Trans. Neural Netw.*, vol. 21, no. 7, pp. 1140–1148, Jul. 2010.

[8] J. Wang, "Analysis and design of a k-winners-tak-all model with a single state variable and the heaviside step activation function," *IEEE Trans. Neural Netw.*, vol. 21, no. 9, pp. 1496–1506, Sep. 2010.

[9] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*. New York: Wiley, 1993.

[10] D. L. Hung and J. Wang, "Digital hardware realization of a recurrent neural network for solving the assignment problem," *Neurocomputing*, vol. 51, pp. 447–461, Apr. 2003.

[11] Q. Liu, J. Cao, and J. Liang, "A discrete-time recurrent neural network with one neuron for k-winners-take-all operation," in *Proc. 6th Int. Symp. Neural Netw.*, May 2009, pp. 272–278.

[12] D. Kinderlehrer and G. Stampcchia, *An Introduction to Variational Inequalities and Their Applications*. New York: Academic, 1980.

[13] X. Hu and J. Wang, "Solving the assignment problem with the improved dual neural network," in *Proc. 8th Int. Symp. Neural Netw.*, May–Jun. 2011, pp. 547–556.

[14] Q. Liu and J. Wang, "A one-layer dual recurrent neural network with a heaviside step activation function for linear programming with its linear assignment application," in *Proc. Int. Conf. Artif. Neural Netw.*, Jun. 2011, pp. 253–260.

[15] X. Hu and B. Zhang, "An alternative recurrent neural network for solving variational inequalities and related optimization problems," *IEEE Trans. Syst., Man, Cybern. B*, vol. 39, no. 6, pp. 1640–1645, Dec. 2009.

[16] L. Cheng, Z.-G. Hou, and M. Tan, "A delayed projection neural network for solving linear variational inequalities," *IEEE Trans. Neural Netw.*, vol. 20, no. 6, pp. 915–925, Jun. 2009.

[17] W. M. Haddad and V. Chellaboina, *Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach*. Princeton, NJ: Princeton Univ. Press, 2008.

[18] H. Zhang, Z. Wang, and D. Liu, "Global asymptotic stability of recurrent neural networks with multiple time-varying delays," *IEEE Trans. Neural Netw.*, vol. 19, no. 5, pp. 855–873, May 2008.

[19] T. Li, A. Song, S. Fei, and T. Wang, "Delay-derivative-dependent stability for delayed neural networks with unbounded distributed delay," *IEEE Trans. Neural Netw.*, vol. 21, no. 8, pp. 1365–1371, Aug. 2010.

[20] Z. Zeng, T. Huang, and W. X. Zheng, "Multistability of recurrent neural networks with time-varying delays and the piecewise linear activation function," *IEEE Trans. Neural Netw.*, vol. 21, no. 8, pp. 1371–1377, Aug. 2010.

[21] Z. Zeng and T. Huang, "New passivity analysis of continuous-time recurrent neural networks with multiple discrete delays," *J. Indust. Manag. Opt.*, vol. 7, no. 2, pp. 283–289, 2011.

[22] Y. Shen and J. Wang, "Robustness analysis of global exponential stability of recurrent neural networks in the presence of time delays and random disturbances," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 1, pp. 87–96, Jan. 2012.

[23] D. E. Knuth, *The Art of Computer Programming, Sorting and Searching*. Reading, MA: Addison-Wesley, 1973.

Estimator Design for Discrete-Time Switched Neural Networks With Asynchronous Switching and Time-Varying Delay

Dan Zhang, Li Yu, Qing-Guo Wang, and Chong-Jin Ong

***Abstract*— This brief deals with the estimator design problem for discrete-time switched neural networks with time-varying delay. One main problem is the asynchronous-mode switching between the neuron state and the estimator. Our goal is to design a mode-dependent estimator for the switched neural networks under average dwell time switching such that the estimation error system is exponentially stable with a prescribed l_2 gain (in the H_∞ sense) from the noise signal to the estimation error. A new Lyapunov functional is constructed that may increase during the mismatched switchings. New results on the stability and l_2 gain analysis are then obtained. The admissible estimator gains are computed by solving a set of linear matrix inequalities. The relations among the switching law, the maximal delay upper bound, and the optimal H_∞ disturbance attenuation level are established. The effectiveness of the proposed design method is finally illustrated by a numerical example.**

***Index Terms*— Asynchronous switching, average dwell time, state estimation, switched neural networks, time-varying delay.**

I. Introduction

Neural networks have received increasing research attention in the last decades due to their successful applications in various areas, including image processing, pattern recognition, associative memory, and optimization problems [1]. Many neural networks may contain inherent time delays in signal transmission, which cause oscillation and instability. In recent years, a great number of results have been reported for various neural networks with time delays. The existence of equilibrium point, global asymptotic/exponential stability, and the existence of periodic solutions have been intensively studied (see [2]–[8]).

In reality, neural networks often exhibit a special characteristic of network-mode switching, which gives rise to the so-called switched neural networks. Namely, the switched neural networks are a class of neural networks whose parameters are operated by a switching signal, see [9]–[14] for more details. In these results, the well-developed techniques

Manuscript received August 1, 2011; accepted January 21, 2012. Date of publication February 24, 2012; date of current version May 2, 2012. This work was supported by the National Natural Science Funds of China under Grant 60974017 and the Graduate Innovation Research Funding of Zhejiang Province under Grant YK2010031.

D. Zhang is with the Department of Automation, Zhejiang Provincial United Key Laboratory of Embedded Systems, Zhejiang University of Technology, Hangzhou 310032, China, and also with the Department of Mechanical Engineering, National University of Singapore, 119260, Singapore (e-mail: jason_zhang19850626@hotmail.com).

L. Yu is with the Department of Automation, Zhejiang Provincial United Key Laboratory of Embedded Systems, Zhejiang University of Technology, Hangzhou 310032, China (e-mail: lyu@zjut.edu.cn).

Q.-G. Wang is with the Department of Electrical and Computer Engineering, National University of Singapore, 119260, Singapore (e-mail: elewqg@nus.edu.sg).

C.-J. Ong is with the Department of Mechanical Engineering, National University of Singapore, 119260, Singapore (e-mail: mpeongcj@nus.edu.sg).

Digital Object Identifier 10.1109/TNNLS.2012.2186824