An Improved Dual Neural Network for Solving a Class of Quadratic Programming Problems and Its k-Winners-Take-All Application

Xiaolin Hu and Jun Wang, *Fellow, IEEE*

***Abstract*—This paper presents a novel recurrent neural network for solving a class of convex quadratic programming (QP) problems, in which the quadratic term in the objective function is the square of the Euclidean norm of the variable. This special structure leads to a set of simple optimality conditions for the problem, based on which the neural network model is formulated. Compared with existing neural networks for general convex QP, the new model is simpler in structure and easier to implement. The new model can be regarded as an improved version of the dual neural network in the literature. Based on the new model, a simple neural network capable of solving the k-winners-take-all (k-WTA) problem is formulated. The stability and global convergence of the proposed neural network is proved rigorously and substantiated by simulation results.**

***Index Terms*—Global asymptotic stability, k-winners-take-all (k-WTA), optimization, quadratic programming (QP), recurrent neural network.**

I. Introduction

SOLVING optimization-related problems by using recurrent neural networks has attracted much attention since the pioneering work of Tank and Hopfield in the 1980s [1], [2]. The models and applications of this category of neural networks have continually enjoyed a prosperous development during the past two decades, e.g., see [3]–[6], [8]–[19], and references therein. The primary motivation of developing recurrent neural networks for solving optimization problems (as well as other problems such as matrix algebra problems [21], which is another hot topic in neural network community) is not based on developing new numerical algorithms that run on conventional digital computers. The working principle of this type of neural networks is fundamentally different from those of iterative numerical algorithms such as interior point algorithms and linear matrix inequality (LMI) algorithms. These neural networks are essentially governed by a set of dynamic equations, which have at least two advantages. On one hand, they give hope to build some brain-like computing models that mimic some working principles of the biological counterparts. Indeed, the pioneering model in this field, Hopfield neural network, originated from neurobiology. On the other hand, their network structures make it possible to design some specific analog circuits that are capable of real-time computing. Such specific circuits can solve problems in running time at the orders of magnitude much faster than the conventional algorithms executed on general purpose digital computers.

In designing a recurrent neural network for solving a specific problem, two major factors refer to its convergence property and structural complexity. Great efforts have been made to reduce the model complexity while preserving the desired convergence property. For instance, for quadratic programming (QP), a series of neural networks have been proposed in [6], [10], [15], and [20] with lower model complexity. In this paper, we are concerned with solving a special class of QP problems by designing a neural network with lower model complexity than existing ones.

The QP problem we are concerned with is as follows:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}x^Tx + p^Tx \\ &\text{subject to} && Ax \le b, \qquad Cx = d, \qquad x \in \Omega \end{aligned} \tag{1}$$

where $x \in \Re^n$ is the decision vector, $p \in \Re^n, A \in \Re^{m\times n}, b \in \Re^m, C \in \Re^{r\times n}, b \in \Re^r$ are constants, and $\Omega \subseteq \Re^n$ is a closed convex set. In this paper, unless specified otherwise, Ω is either a box set defined as

$$\Omega = \{x \in \Re^n | l_i \le x_i \le h_i, \forall i = 1, \cdots, n\} \tag{2}$$

where $l_i, h_i \in \Re$ are constants, or a sphere set defined as

$$\Omega = \{x \in \Re^n | \|x - s\| \le \kappa, \kappa > 0\} \tag{3}$$

where $s \in \Re^n$ and $\kappa \in \Re$ are constants. Throughout this paper, $\|\cdot\|$ denotes the 2-norm of a vector.

A general QP problem is often written in the form of (1) with the objective function $(1/2)x^TQx + p^Tx$ where $Q^T = Q$. Clearly, problem (1) represents a special case of the general QP problem with Q being an identity matrix. We remark that when Ω is a box set and Q is a diagonal matrix with all diagonal elements positive (e.g., in some scenarios, we need to

Manuscript received March 13, 2007; revised September 04, 2007 and February 02, 2008; accepted March 27, 2008. Current version published November 28, 2008. This work was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Grants G_HK010/06 and CUHK417608E, by the National Natural Science Foundation of China under Grants 60805023, 60621062, and 60605003, by the National Key Foundation R&D Projects under Grants 2003CB317007, 2004CB318108, and 2007CB311003, and by the Basic Research Foundation of Tsinghua National Laboratory for Information Science and Technology (TNList).

X. Hu is with the State Key Lab of Intelligent Technology and Systems, Tsinghua National Lab for Information Science and Technology (TNList), and Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: xiaolin.hu@gmail.com).

J. Wang is with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong, China (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/TNN.2008.2003287

minimize the weighted sum of the square of the variables), the problem can be transformed into (1). In fact, define a new variable $\tilde{x} = Q^{1/2}x$, then $x = Q^{-1/2}\tilde{x}$. Substitution of x into the general QP problem will yield

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\tilde{x}^T\tilde{x} + p^TQ^{-1/2}\tilde{x} \\ \text{subject to} \quad & AQ^{-1/2}\tilde{x} \le b, \qquad CQ^{-1/2}\tilde{x} = d \\ & Q^{1/2}l \le \tilde{x} \le Q^{1/2}h \end{aligned}$$

which has the same form as (1).

Problem (1) has many applications. One of them refers to the calculation of the minimum Euclidean distance from a point to a convex set. The problem is formally stated as follows:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|x - \bar{x}\|^2 \\ \text{subject to} \quad & x \in \mathcal{X} \end{aligned} \tag{4}$$

where $\bar{x}$ denotes the given point and $\mathcal{X}$ denotes the set. If $\mathcal{X}$ is defined by a certain combination of sphere sets, linear equalities, and linear inequalities, for instance

$$\mathcal{X} = \{x \in \Re^n | Ax \le b, Cx = d, x \in \Omega\} \tag{5}$$

where the parameters are the same as in (1), problem (4) falls into the category of problem (1). This minimum distance problem is often encountered in robot navigation, where the environment is modeled geometrically by lines and curves, especially in map-based navigation [22]. Online solution to this type of problems is a basic requirement in such scenarios, and the recurrent neural network approach is of particular interest as it is well suited for online computation [1], [2].

Throughout this paper, we assume that the feasible set of problem (1) is nonempty, which ensures that there exists a unique solution to the problem because it is a strictly convex optimization problem.

II. Model Description and Comparisons

A. Model Description

The recurrent neural network proposed in this paper for solving (1) is governed by the following:

- state equation

$$\begin{cases} \dfrac{dy}{dt} = -\lambda\{y - (y + Ax - b)^+\} \\ \dfrac{dz}{dt} = -\lambda\{Cx - d\}; \end{cases} \tag{6a}$$

- output equation

$$x = g_\Omega(-A^Ty + C^Tz - p); \tag{6b}$$

where $\lambda > 0$ is a constant scaling factor and $g_\Omega(\cdot)$ and $(\cdot)^+$ are two activation functions. $g_\Omega(\cdot)$ is defined by

$$g_\Omega(x) = \arg\min_{y\in\Omega} \|x - y\|. \tag{7}$$

Specifically, if Ω is a box set defined in (2), then $g_\Omega(x) = (g_{\Omega_1}(x_1), \cdots, g_{\Omega_n}(x_n))^T$ with

$$g_{\Omega_i}(x_i) = \begin{cases} l_i, & x_i < l_i \\ x_i, & l_i \le x_i \le h_i \\ h_i, & x_i > h_i \end{cases}$$

where $\Omega_i = [l_i, h_i]$. If Ω is a sphere set defined in (3), then

$$g_\Omega(x) = \begin{cases} x, & \|x - s\| \le \kappa \\ s + \dfrac{x - s}{\|x - s\|}, & \|x - s\| > \kappa. \end{cases}$$

The other activation function $(\cdot)^+$ is a special case of $g_\Omega(\cdot)$ in which Ω is set to $\Re^m_+$, the nonnegative quadrant of $\Re^m$.

The architecture of the network is illustrated in Fig. 1. The matrices A and C in the figure are denoted by $\{a_{ij}\}_{m\times n}$ and $\{c_{ij}\}_{r\times n}$, respectively; and other parameters in (6) are self-evident in the figure. The model comprises some amplifiers (realizing the nonlinear activation functions $g_\Omega(\cdot)$ and $(\cdot)^+$), integrators, summators, multipliers, and interconnections. Among them, the number of amplifiers, integrators, and interconnections is crucial in determining the structural complexity of the neural network. Let us discuss one by one.

- In this neural network, there are $n + m$ amplifiers in total. Different from most neural networks in the literature, some amplifiers [those corresponding to $g_\Omega(\cdot)$] in the proposed model are integrated in others [those corresponding to $(\cdot)^+$]. These amplifiers can be realized by some simple circuit units [5], [8], [9].
- From Fig. 1, it is seen that $m + r$ integrators are needed. In analog circuit implementation, integrators are realized by capacitors [1], [2]; and the fewer integrators we have, the better. Note that in the proposed neural network, the number of integrators is not equal to the number of amplifiers, which does not occur in many models such as in [1], [2], [4], [8], [9], and [15].
- The number of interconnections in the model can be counted in the following way. From Fig. 1(a), there are $(m+r+2)$ connections inside this module (the inputs and outputs are not counted) and there are n such modules. So, there are $n(m+r+2)$ connections in these modules. From Fig. 1(b), there are $(n+6)$ connections inside this module (the inputs and outputs are not counted) and there are m such modules. So, there are $m(n+6)$ connections in these modules. From Fig. 1(c), there are $(n + 2)$ connections inside this module (the inputs and outputs are not counted) and there are r such modules. So, there are $r(n + 2)$ connections in these modules. Connecting the outputs y_j's of modules (b) and the outputs z_k's of modules (c) to the input layers of modules (a) requires $n(m + r)$ connections. Connecting the outputs g_{Ω_i}'s to the input layers of modules (c) and (d) requires $n(m + r)$ connections as well. In total, there should be $4n(m + r) + 2n + 2r + 6m$ connections in the neural network.

B. Model Comparisons

In view that (1) is a special case of general QP problem, many existing neural networks in the literature (e.g., [1]–[7], [10]–[13], [15], and [16]) can solve it. Among them, the models devised more than ten years ago such as in [1]–[6] are not competitive with latecomers in terms of either convergence property or structural complexity. The model in [12] can be regarded as a variant of that in [13], but with higher structural complexity.

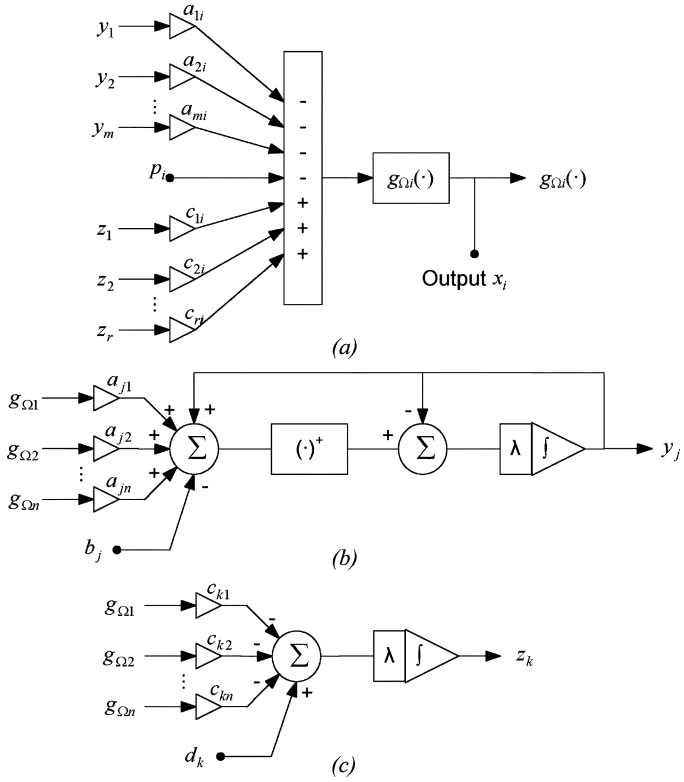

Fig. 1. Architecture of the proposed neural network (6). (a) outputs each component of $g_{\Omega_i}(-A^Ty+C^Tz-p)(i=1,\cdots,n)$, which is fed into (b) and (c). (b) and (c) accept $g_\Omega(\cdot)$ and output y_j $(j=1,\cdots,m)$ and z_k $(k=1,\cdots,r)$, respectively, which are then fed into (a) as an input. The output of (a) is the output of the entire network.

The model in [16] is very much complicated in structure; indeed, its value lies in its proved exponential convergence, not in its structure. Different from other models mentioned above, the model in [11] is not based on differential equation theory, but on differential inclusion theory, which makes it a little difficult for comparison with the new model (6). However, one point is clear: its convergence depends on some parameters that are often inconvenient to choose. This is a shortcoming in view that there are no such parameters in model (6). In the following, we compare the proposed model (6) with the models in [10], [15], and [13] for solving (1).

The model in [7] and [10] is called the dual neural network. When specialized for solving (1), its dynamic equations become as follows:

- state equation

$$\frac{du}{dt}=-\lambda\left\{JJ^Tu-g_{\mathcal{K}_1}(JJ^Tu-u-Jp)-Jp\right\}; \quad (8a)$$

- output equation

$$x=J^Tu-p; \quad (8b)$$

where $\lambda>0$, $J=(A^T,C^T,I_n)^T$ with I_n being an $n\times n$ identity matrix, and $\mathcal{K}_1=\{u\in\Re^{m+r+n}|\underline{\xi}\le u\le\overline{\xi}\}$ with

$$\underline{\xi}=\begin{pmatrix}-\infty\\ d\\ l\end{pmatrix}\quad \overline{\xi}=\begin{pmatrix}b\\ d\\ h\end{pmatrix}.$$

Apparently, there are $m+r+n$ activation functions in this network. A closer look at the definition of $\mathcal{K}_1$ tells that actually only $m+n$ activation functions are required, because a projection to a point d is always equal to d and to ensure that this happens there is no need to use a projection function. Then, the realization of the neural network entails $m+n$ amplifiers. By drawing the diagram of the neural network as we do for neural network (6), it is not difficult to know that the neural network requires $m+n$ integrators and $O((m+r+n)^2+n(m+r+n))$ interconnections (the first term corresponds to the state equation and the second term corresponds to the output equation), in addition to some summators and multipliers. Here, and in what follows, when we estimate the number of interconnections in a neural network, the constants and first-order terms with respect to n,m,r are neglected.

The model in [15] is a simplified version of (8). When specialized for solving (1), its equations become as follows:

- state equation

$$\frac{du}{dt}=-\lambda\left\{EME^Tu-g_{\mathcal{K}_2}(EME^Tu-u+Es)+Es\right\}; \quad (9a)$$

- output equation

$$x=ME^Tu+s; \quad (9b)$$

where $\lambda>0$, $M=I_n-C^T(CC^T)^{-1}C$, $s=C^T(CC^T)^{-1}(Cp+d)-p$, $E=(A^T,I_n)^T$ with I_n being an $n\times n$ identity matrix, and $\mathcal{K}_2=\{u\in\Re^{m+n}|\underline{\xi}\le u\le\overline{\xi}\}$ with

$$\underline{\xi}=\begin{pmatrix}-\infty\\ l\end{pmatrix}\quad \overline{\xi}=\begin{pmatrix}b\\ h\end{pmatrix}.$$

By drawing the diagram of the neural network, it is not difficult to know that the neural network requires $m+n$ amplifiers, $m+n$ integrators, and $O((m+n)^2+n(m+n))$ interconnections (the first term corresponds to the state equation and the second term corresponds to the output equation) in addition to some summators and multipliers.

The third model refers to the extended projection neural network formulated in [13] for solving general convex optimization problems. When specialized for solving (1), its state equation becomes

$$\frac{d}{dt}\begin{pmatrix}x\\ y\\ z\end{pmatrix}=-\lambda\begin{pmatrix}x-g_\Omega(-A^Ty+C^Tz-p)\\ y-(y+Ax-b)^+\\ Cx-d\end{pmatrix} \quad (10)$$

where $\lambda>0$, and its output is simply x. The neural network entails $m+n$ amplifiers, $m+n+r$ integrators, and $O(n(m+n))$ interconnections in addition to some summators and multipliers.

For comparison purpose, we list the numbers of amplifiers, integrators, and interconnections of these neural networks in Table I. Note that r is always smaller than or equal to n [otherwise, problem (1) may have no solution]. We have the following observations: 1) the numbers of amplifiers in these neural networks are all the same; 2) the proposed neural network requires fewest integrators; and 3) the proposed neural network and the extended projection neural network require fewest interconnections. These observations then lead to the conclusion that among

TABLE I
COMPARISONS OF SEVERAL SALIENT NEURAL NETWORKS IN LITERATURE FOR SOLVING (1)

Neural network model	Number of amplifiers	Number of integrators	Number of interconnections
Dual neural network (8)	$n+m$	$n+m+r$	$O((m+r+n)^2+n(m+r+n))$
Simplified dual neural network (9)	$n+m$		$O((m+n)^2+n(m+n))$
Extended projection neural network (10)	$n+m$	$n+m+r$	$O(n(m+n))$
Improved dual neural network (6)	$n+m$	$m+r$	$O(n(m+r))$

these models the proposed neural network (6) is simplest in structure. Moreover, it should be noted that the neural network (9) entails computation of matrix inverse, which makes it less suitable for solving problems with time-varying coefficient C in the equality constraint $Cx=d$. Other models in the table do not have this limitation.

Before ending this section, we would like to point out the connection between the new model (6) and the dual neural network (8), which can be reasoned by some basic algebraic manipulations together with the observation: $g_{(-\infty,b]}(v)=v-(v-b)^+$ for any $v\in\Re^m$.

Theorem 1: If $\Omega=\Re^n$, then the neural networks (6) and (8) are equivalent.

In this sense, the proposed neural network is an improved version of the dual neural network for solving the special QP problem (1).

III. CONVERGENCE ANALYSIS

We first introduce some basic properties of the activation function $g_\Omega(\cdot)$ defined in (7).

Lemma 1 [23, pp. 9–10]: For any $u\in\Re^n$ and any $v\in\Omega$, where Ω is a closed and convex set in $\Re^n$

$$(g_\Omega(u)-u)^T(v-g_\Omega(u))\geq 0.$$

For any $u,v\in\Re^n$

$$\|g_\Omega(u)-g_\Omega(v)\|\leq\|u-v\|.$$

By assumption, there always exists a unique optimal solution to problem (1). In what follows, the unique optimal solution is denoted by x^*.

Theorem 2: $x^*=g_\Omega(-A^Ty^*+C^Tz^*-p)$, where $((y^*)^T,(z^*)^T)^T$ denotes the equilibrium point of the state equation (6a).

Proof: According to the Karush–Kuhn–Tucker (KKT) conditions (see, e.g., [24]), x^* is a solution to problem (1) if and only if there exist $y^*\in\Re^m$ and $z^*\in\Re^r$ such that

$$\begin{cases} g_\Omega(-p-A^Ty^*+C^Tz^*)=x^* \\ (y^*+Ax^*-b)^+=y^* \\ Cx^*-d=0. \end{cases}$$

Substitution of the first equation into the other two equations above yields

$$\begin{cases} \left(y^*+Ag_\Omega(-A^Ty^*+C^Tz^*-p)-b\right)^+=y^* \\ Cg_\Omega(-A^Ty^*+C^Tz^*-p)-d=0 \end{cases} \tag{11}$$

which gives the desired results. □

Lemma 2: Consider the following function:

$$f(y,z)=w^Tg_\Omega(w)-\frac{1}{2}\|g_\Omega(w)\|^2+b^Ty-d^Tz \tag{12}$$

where $w=-A^Ty+C^Tz-p$. Then:

1) f is continuously differentiable with respect to y and z, and the partial derivatives are given by $\partial f/\partial y=-Ag_\Omega(w)+b$ and $\partial f/\partial z=Cg_\Omega(w)-d$;
2) f is convex in $\Re^{m+r}$.

Proof:

1) Note that $f(y,z)$ can be written as

$$f(y,z)=\frac{1}{2}\left\{\|w\|^2-\|w-g_\Omega(w)\|^2\right\}+b^Ty-d^Tz.$$

To show the continuous differentiability of f, we only need to show that the function $\phi(w)=\|w-g_\Omega(w)\|^2$ is continuously differentiable with respect to w. Define another function $\tilde{\phi}:\Re^n\times\Re^n\to\Re$ as $\tilde{\phi}(w,v)=\|w-v\|^2$, then

$$\phi(w)=\min_{v\in\Omega}\tilde{\phi}(w,v).$$

Notice that $\tilde{\phi}$ is continuously differentiable in both w and v, and the minimum solution of the right-hand side (RHS) of above equation is uniquely attained at $v^*=g_\Omega(w)$ for any fixed $w\in\Re^n$, then it follows from [25, Ch. 4, Th. 1.7] that $\phi(w)$ is differentiable and

$$\nabla\phi(w)=\nabla_w\tilde{\phi}(w,v^*)\Big|_{v^*=g_\Omega(w)}=2\left(w-g_\Omega(w)\right).$$

(This is a special case of [18, Lemma 3].) The partial derivatives $\partial f/\partial y$ and $\partial f/\partial z$ then readily follow from the above equation.

2) For any $y_1,y_2\in\Re^m$ and $z_1,z_2\in\Re^r$, we have

$$\begin{aligned}&(y_1-y_2)^T\left(\frac{\partial f}{\partial y_1}-\frac{\partial f}{\partial y_2}\right)+(z_1-z_2)^T\left(\frac{\partial f}{\partial z_1}-\frac{\partial f}{\partial z_2}\right)\\ &\quad=(y_1-y_2)^T\left(-Ag_\Omega(w_1)+Ag_\Omega(w_2)\right)\\ &\qquad+(z_1-z_2)^T\left(Cg_\Omega(w_1)-Cg_\Omega(w_2)\right)\\ &\quad=\left(-A^Ty_1+A^Ty_2+C^Tz_1-C^Tz_2\right)^T\\ &\qquad\times\left(g_\Omega(w_1)-g_\Omega(w_2)\right)\\ &\quad=(w_1-w_2)^T\left(g_\Omega(w_1)-g_\Omega(w_2)\right)\end{aligned}$$

where $w_1=-A^Ty_1+C^Tz_1-p$ and $w_2=-A^Ty_2+C^Tz_2-p$. By using Lemma 1, it is easy to show that

$$(w_1-w_2)^T\left(g_\Omega(w_1)-g_\Omega(w_2)\right)\geq\|g_\Omega(w_1)-g_\Omega(w_2)\|^2\geq 0$$

which implies the convexity of f [24]. □

Theorem 3: $((y^*)^T,(z^*)^T)^T$ is an equilibrium point of the state equation (6a) if and only if it is a solution to the following problem:

$$\begin{aligned} &\text{minimize} \quad f(y,z) \\ &\text{subject to} \quad y \geq 0 \end{aligned} \tag{13}$$

where $f(y,z)$ is defined in (12).

Proof: From Lemma 2, f is a convex and continuously differentiable function. Then, by the KKT conditions, $((y^*)^T,(z^*)^T)^T$ is the solution to (13) if and only if

$$\begin{cases} \left(y^* - \left.\dfrac{\partial f}{\partial y}\right|_{(y^*,z^*)}\right)^+ = y^* \\ \left.\dfrac{\partial f}{\partial z}\right|_{(y^*,z^*)} = 0. \end{cases}$$

Substitution of $\partial f/\partial y$ and $\partial f/\partial z$ into the above equations results in (11). The theorem is then proved. □

It is noted that the state equation of the proposed neural network (6) can be equivalently written as

$$\frac{du}{dt} = -\lambda\left\{u - g_{\mathcal{S}}\left(u - \nabla f(u)\right)\right\} \tag{14}$$

where $u = (y^T,z^T)^T$, $\mathcal{S} = \Re_+^m \times \Re^r$, and $f(u)$ is defined in (12). Equation (14) represents the well-known projection neural network for solving the convex optimization problem (13) studied extensively in the literature (see, e.g., [9], [26], and [27]). However, in our case, $f(u)$ is continuously differentiable but not twice continuously differentiable. As a result, many existing stability results cannot be applied here directly. However, with some modifications in the proofs, we can still obtain some significant results for (14), and consequently, for (6a).

Lemma 3: There exists a unique continuous state trajectory $(y(t)^T,z(t)^T)^T$ for (6a) with any initial point $(y(t_0)^T,z(t_0)^T)^T = (y_0^T,z_0^T)^T$ for $t \in [t_0,\tau)$.

Proof: Let the RHS of (14) be denoted by $L(u)$, where $u = (y^T,z^T)^T$. Then, for any $u_1,u_2 \in \Re^{m+r}$

$$\begin{aligned} &\|L(u_1) - L(u_2)\| \\ &\quad = \lambda\left\|u_1 - g_{\mathcal{S}}\left(u_1 - \nabla f(u_1)\right) - u_2 + g_{\mathcal{S}}\left(u_2 - \nabla f(u_2)\right)\right\| \\ &\quad \leq \lambda\left[\|u_1 - u_2\| + \left\|g_{\mathcal{S}}\left(u_1 - \nabla f(u_1)\right) - g_{\mathcal{S}}\left(u_2 - \nabla f(u_2)\right)\right\|\right] \\ &\quad \leq \lambda\left[2\|u_1 - u_2\| + \|\nabla f(u_1) - \nabla f(u_2)\|\right] \\ &\quad \leq \lambda\left[2\|u_1 - u_2\| + (\|A\| + \|C\|)\right. \\ &\qquad \times \left\|g_\Omega(-A^Ty_1 + C^Tz_1 - p)\right. \\ &\qquad\quad \left.\left. - g_\Omega(-A^Ty_2 + C^Tz_2 - p)\right\|\right] \\ &\quad \leq \lambda\left[2 + (\|A\| + \|C\|)\left\|(-A^T, C^T)\right\|\right]\|u_1 - u_2\|. \end{aligned}$$

In above reasoning, Lemmas 1 and 2 are used. Hence, $L(u)$ is Lipschitz continuous in $\Re^{m+r}$, and there exists a unique continuous solution $(y(t)^T,z(t)^T)^T$ for (6a) with any initial point for $t \in [t_0,\tau)$. □

The heretofore analyses pave a way to present the main result in this section.

Theorem 4: The state vector of neural network (6) is stable in the sense of Lyapunov and globally convergent to an equilibrium point. The output of the neural network is globally convergent to the unique optimum solution x^* of problem (1).

Proof: By Lemma 3, with any initial point $(y_0^T,z_0^T)^T$, (6a) has a unique continuous solution trajectory $(y(t)^T,z(t)^T)^T$ for $t \in [t_0,\tau)$. Consider the following Lyapunov function:

$$V\left(u(t)\right) = \frac{1}{\lambda}\psi\left(u(t)\right) + \frac{1}{2\lambda}\left\|u(t) - u^*\right\|^2 \qquad \forall t \geq t_0 \tag{15}$$

where $u(t) = (y(t)^T,z(t)^T)^T$, $u^* = ((y^*)^T,(z^*)^T)^T$ is an equilibrium point of (14), and

$$\psi\left(u(t)\right) = f\left(u(t)\right) - f(u^*) - \left(u(t) - u^*\right)^T\nabla f(u^*)$$

where $f(u)$ is defined in (12). Since $f(u)$ is convex, $\psi(u) \geq 0$ [24] and $V(u) \geq \|u - u^*\|^2/(2\lambda)$. Then, the time derivative of V along the trajectory of (14) is

$$\frac{dV}{dt} = \left(\nabla f(u) - \nabla f(u^*) + u - u^*\right)^T\left(g_{\mathcal{S}}\left(u - \nabla f(u)\right) - u\right).$$

Similar to [27, Th. 2.1], we can show that

$$\begin{aligned} &\left(\nabla f(u) - \nabla f(u^*) + u - u^*\right)^T\left(g_{\mathcal{S}}\left(u - \nabla f(u)\right) - u\right) \\ &\quad \leq -(u - u^*)^T\left(\nabla f(u) - \nabla f(u^*)\right) \\ &\qquad - \left\|g_{\mathcal{S}}\left(u - \nabla f(u)\right) - u\right\|^2 \end{aligned}$$

which implies

$$\frac{dV}{dt} \leq -\left\|g_{\mathcal{S}}\left(u - \nabla f(u)\right) - u\right\|^2 \leq 0.$$

Therefore, the state vector in (6a) is stable in the sense of Lyapunov. Moreover, the above inequality indicates that

$$\left\{u(t)|t_0 \leq t < \tau\right\} \subset \mathcal{S}_0 = \left\{u \in \Re^{m+r}|V(u) \leq V\left(u(t_0)\right)\right\}.$$

Since $V \geq \|u - u^*\|^2/(2\lambda)$, $\mathcal{S}_0$ is bounded, which follows that $u(t)$ is bounded. Hence, $\tau = \infty$. The boundedness of $u(t)$ also indicates that there exists a convergent subsequence $\{u(t_k)\}$ such that

$$\lim_{k\to\infty} u(t_k) = \hat{u}$$

where $\hat{u}$ is an equilibrium point of (6a). Finally, define a Lyapunov function again

$$\begin{aligned} \hat{V}_1(u) = \frac{1}{\lambda}\left(f\left(u(t)\right) - f(\hat{u}) - \left(u(t) - \hat{u}\right)^T\nabla f(\hat{u})\right) \\ + \frac{1}{2\lambda}\left\|u(t) - \hat{u}\right\|^2 \qquad \forall t \geq t_0. \end{aligned}$$

It is easy to see that $\hat{V}(u)$ decreases along the trajectory of (6a) and satisfies $\hat{V}(\hat{u}) = 0$. Therefore, for any $\varepsilon > 0$, there exists $q > 0$ such that, for all $t \geq t_q$

$$\left\|u(t) - \hat{u}\right\|^2/(2\lambda) \leq \hat{V}\left(u(t)\right) \leq \hat{V}\left(u(t_q)\right) < \varepsilon.$$

Therefore, $\lim_{t\to\infty} u(t) = \hat{u}$. It follows that (6a) is globally convergent to one of its equilibrium points. As a result, from Theorem 2, the output trajectory globally converges to the unique solution x^* of problem (1). □

Corollary 1: In problem (1), let $a_i, c_j \in \Re^{1\times n}$ denote the row vectors of A and C, respectively $(i = 1,\cdots,m, j = 1,\cdots,r)$.

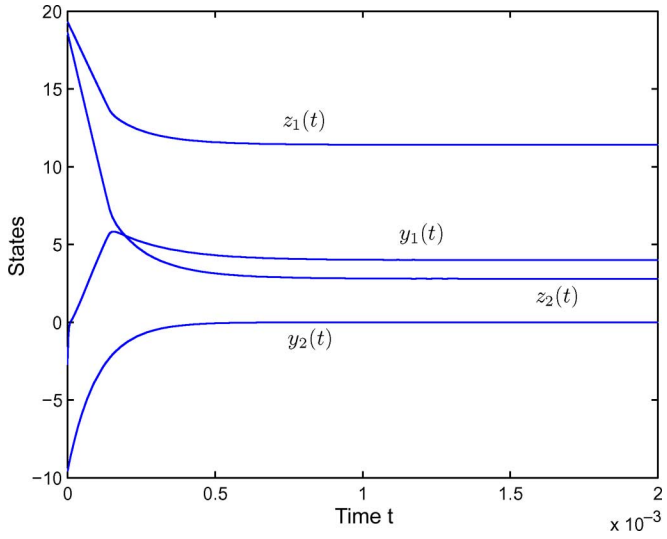

Fig. 2. Transient behavior of the states of neural network (6) started from a random initial point with $\lambda = 10^4$ for solving the first problem in Example 1.

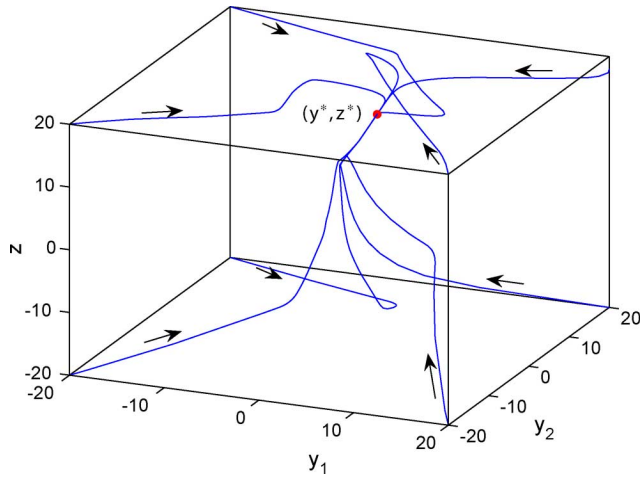

Fig. 3. State trajectories of neural network (6) started from eight corners of the box $[-20, 20]^3$ with $\lambda = 10^4$ for solving the second problem in Example 1 (the second equality constraint is deleted from the first problem).

If the vectors a_i for $i \in \{i|a_i x^* = b_i\}$ and c_j for $j = 1, \cdots, r$ are linearly independent, then the state vector of neural network (6) is globally asymptotically stable in the sense of Lyapunov at its unique equilibrium point.

Proof: The condition in the corollary actually implies the uniqueness of Lagrangian multipliers y^* and z^* associated with the solution x^*. Then, the result follows from Theorem 4 immediately. □

IV. Illustrative Examples

In this section, we illustrate the performance of the proposed neural network by solving two problems. The simulations are conducted in MATLAB.

Example 1: Consider a 4-D QP problem (1) with

$$p = (3, 0, 2, 6)^T \quad A = \begin{pmatrix} 2 & 6 & -1 & 0 \\ -2 & 0 & 1 & 3 \end{pmatrix} \quad b = \begin{pmatrix} 10 \\ 10 \end{pmatrix}$$
$$C = \begin{pmatrix} 1 & 0 & 0 & -2 \\ 2 & 0 & 0 & -4 \end{pmatrix} \quad d = \begin{pmatrix} 6 \\ 12 \end{pmatrix} \quad \Omega = [0, 10]^4.$$

We use the proposed neural network (6) to solve the problem. Simulation results show that the state vector of the neural network is convergent to one of its equilibrium points from any initial point and the output is always convergent to the unique solution $(6, 0, 2, 0)^T$. For example, Fig. 2 illustrates the transient states in one simulation where the scaling factor λ is set to 10^4. The trajectories converge to $((y^*)^T, (z^*)^T)^T = (4.0022, -0.0000, 11.4039, 2.7955)^T$, which corresponds to $x^* = (5.9906, 0, 2.0022, 0)^T$, very close to the exact solution.

Note that for solving above problem the state equation of the neural network has multiple (infinity number of) equilibrium points. This is because the two rows in parameter C are linearly dependent. Now, we remove the second equality constraint and solve the problem again. Simulations show that the state equation is then globally asymptotically stable at its unique equilibrium point $(4, 0, 17)^T$, a phenomenon consistent with Corollary 1. Fig. 3 illustrates the state trajectories $y_1(t), y_2(t), z(t)$ from eight corners of the box $[-20, 20]^3$, all of which converge to this unique equilibrium point.

Example 2: The minimum distance problem (4) has an application in designing recurrent neural network for solving variational inequalities (VIs) and related optimization problems. In general, a nonlinear VI assumes the form

$$F(x^*)^T (x - x^*) \geq 0 \qquad \forall x \in \mathcal{X} \tag{16}$$

where $\mathcal{X}$ is a closed convex set in $\Re^n$ and $F(x)$ is a continuous vector-valued function from $\Re^n$ to $\Re^n$. The problem is to find $x^* \in \mathcal{X}$ such that (16) holds. For the relationship between VIs, complementarity problems and optimization problems, readers are referred to [28] for an excellent survey. For solving VIs via recurrent neural networks, recent studies suggest a very competitive model, the projection neural network model [8], [9], [14], [27]

$$\frac{dx}{dt} = -\lambda \{x - g_{\mathcal{X}}(x - F(x))\} \tag{17}$$

where $\lambda > 0$ is a constant and $g_{\mathcal{X}} : \Re^n \to \mathcal{X}$ is defined in (7). It is seen that the above system includes system (14) as a special case.

Aiming at easily realizing the activation function $g_{\mathcal{X}}(\cdot)$, $\mathcal{X}$ is always assumed to be a box or a sphere set (see Section II-A for the discussion of the two cases), though most stability and convergence results for system (17) are valid for general closed and convex set $\mathcal{X}$, e.g., an $\mathcal{X}$ defined in (5). The definition of the activation function in (7) indicates that it can be determined by solving a minimum distance problem in the form of (4). This observation motivates us to discretize the continuous-time system (17) as

$$x^{(k+1)} = x^{(k)} - \eta \left\{ x^{(k)} - g_{\mathcal{X}} \left(x^{(k)} - F\left(x^{(k)}\right) \right) \right\} \tag{18}$$

where $\eta > 0$ is a constant step size, and to solve a minimum distance problem at each step k by using neural network (6). Specifically, the minimum distance problem is in the form of (4) with $\bar{x} = x^{(k)} - F(x^{(k)})$, and consequently, we need to substitute p in the neural network (6) with $-x^{(k)} + F(x^{(k)})$.

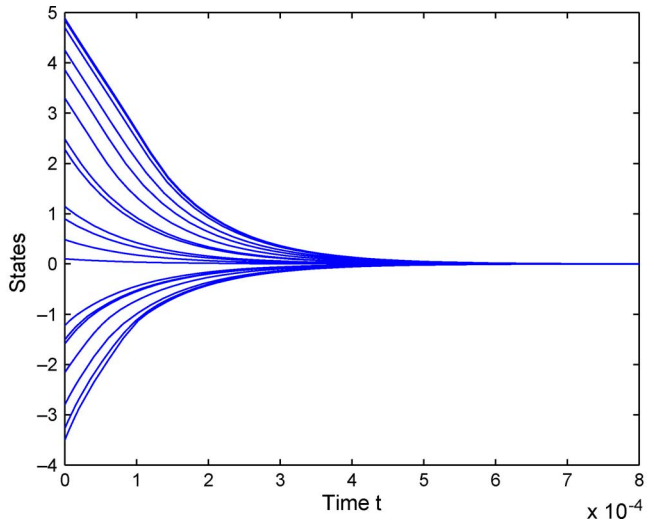

Fig. 4. Transient behavior of the states of neural network (6) started from 20 random initial points with $\lambda = 10^4$ for computing the projection of $(2.5, 0)^T$ onto $\mathcal{X}$ in Example 2.

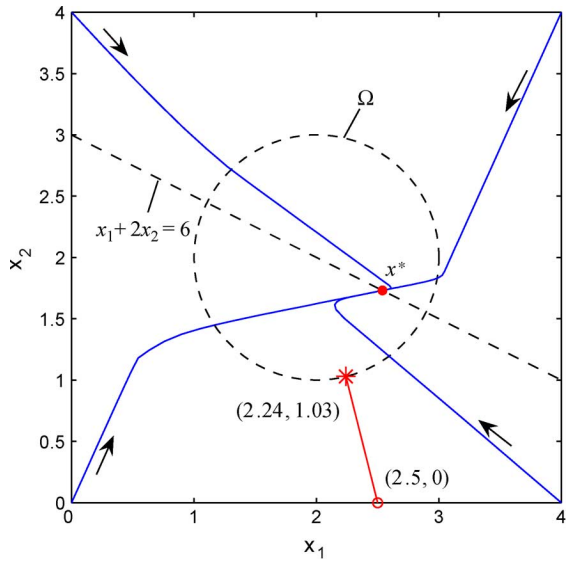

Fig. 5. State trajectories of neural network (18) started from four corners of the box $[0, 4]^2$ with $\eta = 1$ in Example 2.

In general, if (17) is proved to be globally convergent to the solutions of the VI (16) with some choice of λ, then (18) is globally convergent to the solutions of the VI with sufficiently small η. Then, we obtain a hierarchical neural network, in which the high-level network computes solutions to the VI and the low-level network computes the projection of a point onto the feasible set. This hierarchical approach is of particular interest if F is pesudomonotone but not monotone in $\mathcal{X}$ because other candidates such as in [13] for solving constrained VIs cannot deal with this type of the VI.

In this example, we use the discrete-time projection neural network (18) together with neural network (6) to solve a VI in (16) with

$$F(x) = \begin{pmatrix} -5x_2 + 5 \\ -5x_1 + 6x_2 - 5 \end{pmatrix}$$
$$\mathcal{X} = \{x \in \Re^2 | x_1 + 2x_2 \leq 6, x \in \Omega\}$$

where $\Omega = \{x \in \Re^2 | (x_1 - 2)^2 + (x_2 - 2)^2 \leq 1\}$ is a circular plate on the $x_1 - x_2$ plane. Note that $F(x)$ is not monotone, but pseudomonotone in $\mathcal{X}$; moreover, its Jacobian matrix is symmetric. According to [14, Th. 2], the continuous-time projection neural network (17) should be globally convergent to a solution of the VI. Consequently, the discrete-time projection neural network (18) should be globally convergent to a solution provided that η is sufficiently small. However, because of the inequality constraint in $\mathcal{X}$, neural network (18) cannot be applied directly to solve the problem. To determine the projection of a point onto $\mathcal{X}$ (i.e., the lower part of the circular plate Ω divided by a dashed line in Fig. 5), neural network (6) is utilized. For example, we determine the projection of a point $(2.5, 0)^T$ (denoted by a small circle in Fig. 5) onto $\mathcal{X}$ by using neural network (6) with $\lambda = 10^4$. Fig. 4 illustrates the trajectories of the state trajectories started from 20 different initial points. It is seen that all trajectories converge to zero. From the output equation, the projection point is calculated as $(2.24, 1.03)^T$ (denoted by an asterisk in Fig. 5).

With the aid of neural network (6), the discrete-time projection neural network (18) is able to solve the VI. Fig. 5 shows the state trajectories on the phase plane started from the corners of the square $[0, 4]^2$. It is observed that all the trajectories globally converge to $(2.539, 1.731)^T$, the unique solution of the problem.

V. A k-Winners-Take-All Network

In this section, the neural network (6) is tailored for solving the k-winners-take-all (k-WTA) problem

$$x_i = f(v_i) = \begin{cases} 1, & \text{if } v_i \in \{k \text{ largest elements of } v\} \\ 0, & \text{otherwise} \end{cases} \quad (19)$$

where $v \in \Re^n$ stands for the input vector and $x \in \{0, 1\}^n$ stands for the output vector. The k-WTA operation accomplishes a task of selecting k largest inputs among n inputs in a network. It has been shown to be a computationally powerful operation compared with standard neural network models with threshold logic gates [29]. In addition, it has important applications in machine learning as well, such as k-neighborhood classification and k-means clustering.

Many attempts have been made to design neural networks to perform the k-WTA operation (e.g., [29]–[34], and [36]). However, most of them require $O(n^2)$ interconnections among amplifiers. In [35], a neural circuit with $O(n)$ interconnections was devised for doing this operation. However, the model is somewhat difficult to use because there are many parameters to be chosen (the parameter-choosing procedure is decomposed into five steps). Recently, the k-WTA operation was formulated into a QP problem and a simplified dual neural network was developed to solve the problem [15]. Assume that the difference between the kth and $(k+1)$th largest inputs is at least $2a$, where $a > 0$, then the k-WTA problem is equivalent to the following QP problem [15, Th. 4]:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}x^T x - \frac{1}{2a}v^T x \\ \text{subject to} \quad & e^T x = k, \qquad x \in [0, 1]^n \end{aligned} \quad (20)$$

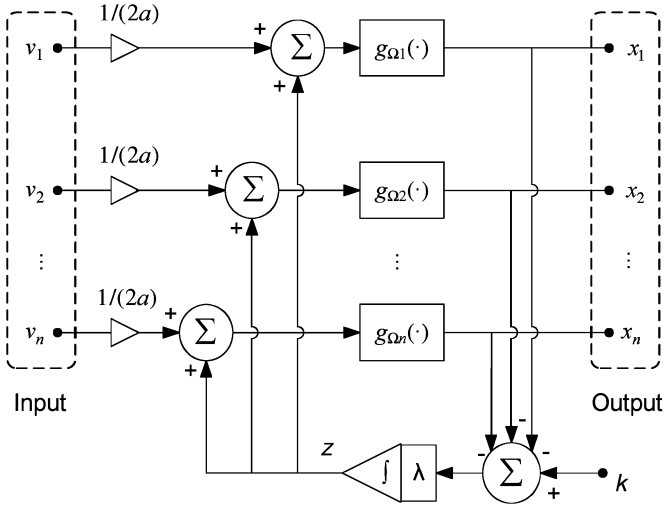

Fig. 6. Architecture of the k-WTA neural network (22).

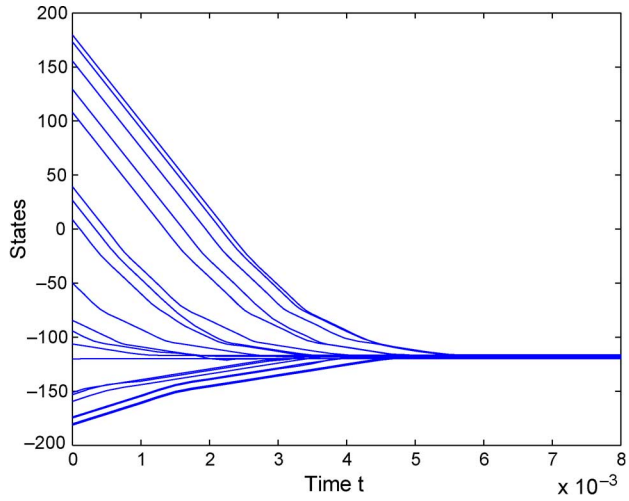

Fig. 7. State trajectories of the k-WTA network (22) with $\lambda = 10^4$, $a = 0.1$ started from 20 random initial points within $[-200, 200]$ in Example 3.

where e is a column vector with all entries being 1s. Then, a k-WTA neural network was tailored from (9) [15] as follows:

- state equation

$$\frac{du}{dt} = -\lambda \{Mu - g_\Omega(Mu - u + s) + s\}; \tag{21a}$$

- output equation

$$x = Mu + s; \tag{21b}$$

where $\lambda > 0$, $M = (I_n - ee^T/n)/a$, $s = Mv + (k/n)e$, and $\Omega = [0,1]^n$. From [15], this network has n amplifiers, n integrators, and $O(n)$ interconnections.

Comparing (20) with (1), and following from the neural network (6), we have a k-WTA network described by the following:

- state equation

$$\frac{dz}{dt} = -\lambda \left\{ \sum_{i=1}^{n} x_i - k \right\}; \tag{22a}$$

- output equation

$$x_i = g_{\Omega_i}\left(z + \frac{v_i}{2a}\right), \qquad i = 1, \cdots, n; \tag{22b}$$

where $z \in \Re$, $\lambda > 0$ is a scaling factor, and $\Omega_i = [0,1]$ for $i = 1, \cdots, n$. The structure of the network is illustrated in Fig. 6. It is clear from the figure that realization of the network requires n amplifiers and just one integrator. Moreover, it is observed that there is no interconnection between any pair of amplifiers, and consequently, the total number of interconnections is of order $O(n)$.

In addition to having fewer integrators than network (21), by comparing [15, Fig. 7] and Fig. 6 in this paper, the new model (22) is much simpler in structure and thus more favorable for circuit implementation.

The following stability results for (22) follow from Theorem 4 and Corollary 1 by noticing the special structure of problem (20).

Corollary 2: The state vector of neural network (22) is globally asymptotically stable at its unique equilibrium point, and the output trajectory of the neural network is globally convergent to the unique solution of problem (20).

Example 3: First, consider a k-WTA problem with $n = 10$, $k = 2$, and ten inputs $v = (15.3, 23.3, 14.7, 5.6, 21.0, 29.5, 24.2, 21.1, 14.5, 3.4)^T$ randomly generated within the interval $[0, 30]$. Fig. 7 depicts the state trajectories of the k-WTA network with the resolution parameter $a = 0.1$ and scaling constant $\lambda = 10^4$, where each trajectory starts from a different initial state and converges to a value $[-119.9, -116.5]$. Any value in this interval corresponds to the unique output of the network $x^* = (0, 0, 0, 0, 0, 1, 1, 0, 0, 0)^T$, which is the correct solution.

The following numerical results show the relationship between the difference sizes of the problem and the convergence times of the network. Let $n = 5, 10, 15, \cdots, 100$ and $k = n/5$. For each pair of n and k, 30 sets of n inputs v_i are randomly generated within the interval $[0, 30]$ and the k-WTA network with the same initial state are simulated. The convergence time of the network is defined as the time when the RHS of (22a) becomes smaller than 0.001. For each pair of n and k, 30 convergence times are obtained and averaged. Fig. 8 shows a curve of the average convergence time with respect to different values of n. It is interesting to see that the average convergence time decreases as the problem size increases. This property can be deemed as another merit of the proposed network, though a solid theoretical analysis is still lacking at this stage. The similar phenomenon was observed for another k-WTA network based on the QP formulation (20) in [36].

Example 4: In (19) let $k = 2$, $n = 4$, and the inputs $v_i = 10\sin[2\pi(t + 0.2(i-1))]$, $\forall i = 1, 2, 3, 4$, where $t = 0$; i.e., $v = (0, 9.511, 5.878, -5.878)^T$. The parameter settings come from [15]. In the equivalent QP problem (20), let the resolution parameter $a = 0.1$. Simulation results show that the neural network is always convergent to the solution $(0, 1, 1, 0)^T$, which implies that v_2 and v_3 should be selected. Fig. 9 depicts the state trajectory of the neural network with $\lambda = 10^4$ but different values of a from a random initial point within the first 0.001 time units. It is seen that greater a implies faster convergence. However, when a decreases from 0.1 to 0.001, the trajectories

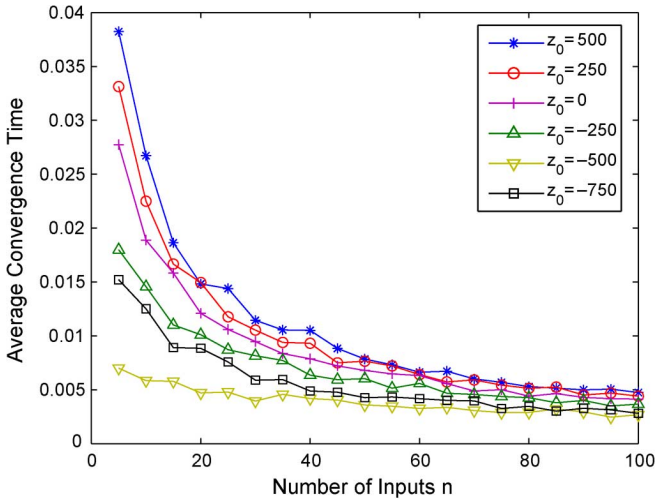

Fig. 8. Average convergence time of the k-WTA network (22) with respect to different problem sizes in Example 3, with $\lambda = 10^4$, $a = 0.02$, and six different initial points z_0.

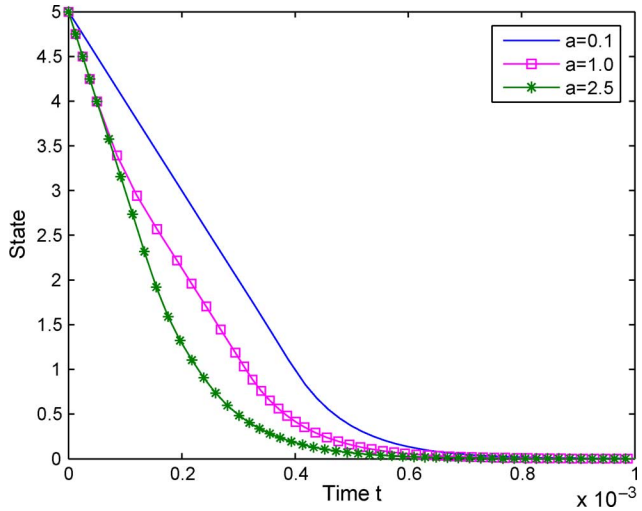

Fig. 9. State trajectory of the k-WTA network (22) with $\lambda = 10^4$ and different values of a but started from the same initial point $z_0 = 5$ in Example 4, within the first 0.001 time units.

are almost the same as that for $a = 0.1$. Now let t range from 0 to 1 continuously, which leads to four sinusoidal input signals v_i for $i = 1, \cdots, 4$ (see the top graph in Fig. 10). The other four graphs in Fig. 10 record the outputs of the k-WTA network at each time instant t. It is readily checked that the outputs are correct.

VI. Concluding Remarks

In this paper, an improved dual neural network is presented for solving a class of QP problem. Compared with its predecessor, the dual neural network, the new network is simpler in structure, which is due to the special structure of such problems. Specifically, if the bound constraint is presented, the neural network can be regarded as a two-layer neural network. If the bound constraint is absent, the neural network reduces to the dual neural network. The global convergence of the neural

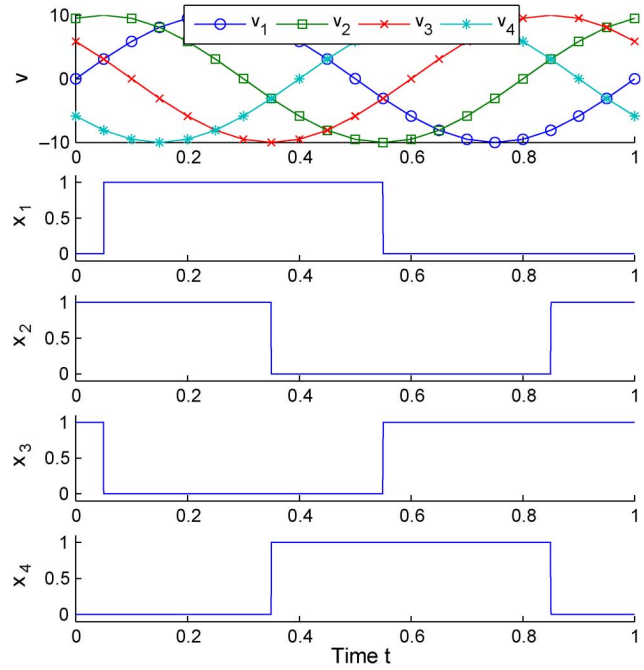

Fig. 10. Inputs and outputs of the k-WTA network (22) in Example 4.

network can be guaranteed by rather weak conditions. An interesting application for the k-WTA operation is discussed, which results in a simple k-WTA network. Numerical results are shown to demonstrate the performance of the neural networks. In particular, the simulation results in an example show that the convergence time of the k-WTA network decreases when the problem size increases on average, which indicates that its convergence rate with respect to the problem size deserves further investigation.

References

[1] J. J. Hopfield and D. W. Tank, "Computing with neural circuits: A model," *Science*, vol. 233, no. 4764, pp. 625–633, Aug. 1986.

[2] D. W. Tank and J. J. Hopfield, "Simple neural optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits Syst.*, vol. CAS-33, no. 5, pp. 533–541, May 1986.

[3] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Trans. Circuits Syst.*, vol. CAS-35, no. 5, pp. 554–562, May 1988.

[4] J. Wang, "A deterministic annealing neural network for convex programming," *Neural Netw.*, vol. 7, no. 4, pp. 629–641, 1994.

[5] M. Forti and A. Tesi, "New conditions for global stability of neural networks with application to linear and quadratic programming problems," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 42, no. 7, pp. 354–366, Jul. 1995.

[6] Y. Xia, "A new neural network for solving linear and quadratic programming problems," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1544–1547, Nov. 1996.

[7] Y. Xia and J. Wang, "A dual neural network for kinematic control of redundant robot manipulators," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 31, no. 1, pp. 147–154, Feb. 2001.

[8] X. Liang and J. Si, "Global exponential stability of neural networks with globally Lipschitz continuous activations and its application to linear variational inequality problem," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 349–359, Mar. 2001.

[9] Y. Xia, H. Leung, and J. Wang, "A projection neural network and its application to constrained optimization problems," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 49, no. 4, pp. 447–458, Apr. 2002.

[10] Y. Zhang and J. Wang, "A dual neural network for convex quadratic programming subject to linear equality and inequality constraints," *Phys. Lett. A*, vol. 298, pp. 271–278, 2002.
[11] M. Forti, P. Nistri, and M. Quincampoix, "Generalized neural network for nonsmooth nonlinear programming problems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 9, pp. 1741–1754, Sep. 2004.
[12] X. Gao, "A novel neural network for nonlinear convex programming," *IEEE Trans. Neural Netw.*, vol. 15, no. 3, pp. 613–621, May 2004.
[13] Y. Xia, "An extended projection neural network for constrained optimization," *Neural Comput.*, vol. 16, pp. 863–883, 2004.
[14] X. Hu and J. Wang, "Solving pseudomonotone variational inequalities and pseudoconvex optimization problems using the projection neural network," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1487–1499, Nov. 2006.
[15] S. Liu and J. Wang, "A simplified dual neural network for quadratic programming with its KWTA application," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1500–1510, Nov. 2006.
[16] Y. Yang and J. Cao, "Solving quadratic programming problems by delayed projection neural network," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1630–1634, Nov. 2006.
[17] X. Hu and J. Wang, "Solving generally constrained generalized linear variational inequalities using the general projection neural networks," *IEEE Trans. Neural Netw.*, vol. 18, no. 6, pp. 1697–1708, Nov. 2007.
[18] X. Hu and J. Wang, "A recurrent neural network for solving a class of general variational inequalities," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 37, no. 3, pp. 528–539, Jun. 2007.
[19] X. Hu and J. Wang, "Design of general projection neural networks for solving monotone linear variational inequalities and linear and quadratic optimization problems," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 37, no. 5, pp. 1414–1421, Oct. 2007.
[20] Q. Liu and J. Wang, "A one-layer recurrent neural network with a discontinuous hard-limiting activation function for quadratic programming," *IEEE Trans. Neural Netw.*, vol. 19, no. 4, pp. 558–570, Apr. 2008.
[21] F. M. Ham and I. Kostanic, *Principles of Neurocomputing for Science and Engineering*. New York: McGraw-Hill, 2001.
[22] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 237–267, Feb. 2002.
[23] D. Kinderlehrer and G. Stampcchia, *An Introduction to Variational Inequalities and Their Applications*. New York: Academic, 1980.
[24] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*. New York: Wiley, 1993.
[25] A. Auslender, *Optimisation: Méthodes Numériques*. Paris, France: Masson, 1976.
[26] A. Nagurney and D. Zhang, *Projected Dynamical Systems and Variational Inequalities With Applications*. Boston, MA: Kluwer, 1996.
[27] Y. Xia, "Further results on global convergence and stability of globally projected dynamical systems," *J. Optim. Theory Appl.*, vol. 122, no. 3, pp. 627–649, 2004.
[28] P. T. Harker and J. S. Pang, "Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms and applications," *Math. Program.*, vol. 48, pp. 161–220, 1990.
[29] W. Maass, "On the computational power of winner-take-all," *Neural Comput.*, vol. 12, pp. 2519–2535, 2000.
[30] E. Majani, R. Erlanson, and Y. Abu-Mostafa, "On the k-winners-take-all network," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed. San Mateo, CA: Morgan-Kaufmann, 1989, vol. 1, pp. 634–642.
[31] G. L. Dempsey and E. S. McVey, "Circuit implementation of a peak detector neural network," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 40, no. 9, pp. 585–591, Sep. 1993.
[32] J. Wang, "Analogue winner-take-all neural networks for determining maximum and minimum signals," *Int. J. Electron.*, vol. 77, no. 3, pp. 355–367, 1994.
[33] J. P. F. Sum, C. S. Leung, P. K. S. Tam, G. H. Young, W. K. Kan, and L. W. Chan, "Analysis for a class of winner-take-all model," *IEEE Trans. Neural Netw.*, vol. 10, no. 1, pp. 64–71, Jan. 1999.
[34] B. A. Calvert and C. Marinov, "Another k-winners-take-all analog neural network," *IEEE Trans. Neural Netw.*, vol. 11, no. 4, pp. 829–838, Jul. 2000.
[35] C. A. Marinov and J. J. Hopfield, "Stable computational dynamics for a class of circuits with $O(N)$ interconnections capable of KWTA and rank extractions," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 5, pp. 949–959, May 2005.
[36] Q. Liu and J. Wang, "Two k-winners-take-all networks with discontinuous activation functions," *Neural Netw.*, vol. 21, no. 2–3, pp. 406–413, 2008.

Xiaolin Hu received the B.E. and M.E. degrees in automotive engineering from Wuhan University of Technology, Wuhan, China, and the Ph.D. degree in automation and computer-aided engineering from The Chinese University of Hong Kong, Hong Kong, China, in 2001, 2004, and 2007, respectively.

Currently, he is a Postdoctoral Fellow at Tsinghua National Lab of Information Science and Technology, State Key Lab of Intelligent Technology and Systems, and Department of Computer Science and Technology, Tsinghua University, Beijing, China. His current research interests are theories and applications of artificial neural networks. His general interests also include evolutionary computation, computational neuroscience, and multimedia processing.

Jun Wang (S'89–M'90–SM'93–F'07) received the B.S. degree in electrical engineering and the M.S. degree in systems engineering from Dalian University of Technology, Dalian, China, and the Ph.D. degree in systems engineering from Case Western Reserve University, Cleveland, OH, in 1982, 1985, and 1991, respectively.

Currently, is a Professor in the Department of Mechanical and Automation Engineering, Chinese University of Hong Kong, Hong Kong. He held various academic positions at Dalian University of Technology, Case Western Reserve University, and University of North Dakota. He also held various short-term visiting positions at USAF Armstrong Laboratory (1995), REKEN Brain Science Institute (2001), Universite catholique de Louvain (2001), Chinese Academy of Sciences (2002), and Huazhong University of Science and Technology (2006–2007). He has been holding a Cheung Kong Chair Professorship in computer science and engineering at Shanghai Jiao Tong University since 2008. His current research interests include neural networks and their applications.

Dr. Wang has been an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS since 1999 and theIEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS since 2003, and a member of the Editorial Advisory Board of the *International Journal of Neural System* since 2006. He also served as an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS (2002–2005), a guest editor of the special issues of *European Journal of Operational Research* (1996), *International Journal of Neural Systems* (2007), and *Neurocomputing* (2008). He was an organizer of several international conferences such as the General Chair of the 13th International Conference on Neural Information Processing (2006) and the 2008 IEEE World Congress on Computational Intelligence held in Hong Kong. He served as the President of Asia Pacific Neural Network Assembly in 2006.