A Simplified Dual Neural Network for Quadratic Programming With Its KWTA Application

Shubao Liu and Jun Wang, *Senior Member, IEEE*

***Abstract*—The design, analysis, and application of a new recurrent neural network for quadratic programming, called simplified dual neural network, are discussed. The analysis mainly concentrates on the convergence property and the computational complexity of the neural network. The simplified dual neural network is shown to be globally convergent to the exact optimal solution. The complexity of the neural network architecture is reduced with the number of neurons equal to the number of inequality constraints. Its application to k-winners-take-all (KWTA) operation is discussed to demonstrate how to solve problems with this neural network.**

***Index Terms*—Global stability, k-winners-take-all (KWTA), quadratic programming, recurrent neural networks.**

I. Introduction

In the past half-century, tremendous research efforts have been put into constrained optimization, which plays an important role in current research and development. As the outcome, many efficient algorithms have been developed, e.g., the simplex methods for linear programming, active set methods, and interior point methods for nonlinear optimization. These methods have been discussed in detail in many books, such as [1] and [2]. Also several stochastic parallel optimization methods, such as simulated annealing and evolutionary algorithms, have been proposed to deal with the general global optimization; see [3] and [4]. All these methods have been successfully applied to parametric constrained optimization problems. But many problems are inherently dynamic, i.e., the parameters of the problems are time-varying. Usually the problems in these circumstances can be formulated as dynamic constrained optimization problems. Many of these sample problems in robot control, signal processing, and biomedical engineering are described in [5]. The real-time solution requirement of dynamic optimization constitutes a major challenge for the traditional parametric optimization methods.

Recurrent neural networks (RNNs) have the features of high parallelism, adaptivity, and circuit implementability. These characteristics can be utilized to solve dynamic constrained optimization problems. Remarkable advances have been made in the area of RNNs during the past twenty years, in both theory and applications.

Manuscript received November 14, 2004; revised January 10, 2006.

S. Liu is with the Division of Engineering, Brown University, Providence, RI 02912 USA (e-mail: shubao_liu@brown.edu).

J. Wang is with the Department of Automation and Computer-Aided Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong, China (e-mail: jwang@acae.cuhk.edu.hk).

Color versions of Figs. 10–12 are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2006.881046

This paper is concerned with recurrent neural networks for solving dynamic convex quadratic programming problems with equality and inequality constraints. The interest in such problems arises from two observations. First, many engineering problems can be turned to this formulation because the two-norm or least-square optimization is a convenient choice for both engineering plausibility and mathematical tractability. For example, in robot motion control, the desired position may change with time, and the obstacle may move; in wireless communication, the channels change rapidly with time, because of both natural communicating conditions (like multipath fading, mobile terminal moving, etc.) and random asynchronous access of other interfering users [9]. Therefore, both the motion planning and control of kinematically redundant manipulators and the multiuser detection of wireless communication can be formulated as a convex quadratic optimization problem with time-varying parameters [20], [12]. Second, in many methods, general nonlinear programming can be solved by a series of quadratic optimization.

In the past two decades, neural networks for optimization and their engineering applications have been widely investigated. Various results have been reported extensively in the literature. Many of them have been successfully applied in engineering optimization. A continuous-time recurrent neural network is a dynamical system and its stable state is called an attractor. Dynamical systems are called dissipative if their dynamics converge to attractors. When a dissipative system has an energy (Lyapunov) functional, the attractors are fixed points; otherwise, more complex attractors may appear. Various continuous-time recurrent neural networks have been designed for computing by utilizing their stable states. This computing paradigm is called computing with attractors [8]. Tank and Hopfield proposed the first working recurrent neural network implemented on analog circuits [14], [15], which opened the avenue of solving optimization problems by using recurrent neural networks. Over years, various neural network models have been developed for solving quadratic programming problems. According to their design method, these neural networks can be categorized as the penalty-parameter neural network [16], the two-layer Lagrange neural network [17], the two-layer primal-dual neural network [19], and the one-layer dual neural network [20], [21]. It is known that the neural network model [16] contains finite penalty parameters and generate approximate solutions only. The dimensionality of Lagrange network is much larger than that of original problems, due to the introduction of slack and surplus variables. As a much flexible tool for exactly solving quadratic programming problems, the primal-dual neural network [19], [22] was developed with the feature that it handles the primal quadratic program and its dual problem simultaneously by minimizing the duality gap

using the gradient method. Unfortunately, the dynamic equations of the primal-dual neural network are usually complicated, and may contain second-order nonlinear terms. Moreover, the network size is usually larger than or equal to the dimensionality of the primal quadratic program and its dual problem. In [20] and [21], a neural network called dual neural network is presented to solve convex quadratic problems utilizing only the dual variables. In this paper, we will give a new neural network called simplified dual neural network based on dual neural network for solving quadratic programs problems. It can further reduce the architecture complexity while preserving the desired convergence property.

The remainder of this paper is organized as follows. In Section II, we present the simplified dual neural network for solving convex quadratic programming problems. In Section III, the neural network is proved to be globally stable and convergent to the exact optimal solution. A numerical example is simulated to show its convergence behavior. Its application to k-winners-take-all (KWTA) operation is discussed in detail in Section IV. Section V concludes this paper.

II. Model Description

A general dynamic convex quadratic optimization problem can be expressed as

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}x^T W(t)x + c(t)^T x \\ &\text{subject to} \quad A(t)x = b(t), \qquad l(t) \le E(t)x \le h(t) \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ is the decision variable, $W(t) \in \mathbb{S}^n_{++}$,[1] $c(t) \in \mathbb{R}^n$, $A(t) \in \mathbb{R}^{m\times n}$, $b(t) \in \mathbb{R}^m$ $(m < n)$, $E(t) \in \mathbb{R}^{p\times n}$, $l(t), h(t) \in \mathbb{R}^p$. Here, if $l(t) = -\infty$ or $h(t) = \infty$, the inequality constraints are one-sided; if $l(t) = -\infty$ and $h(t) = \infty$, the inequality constraints disappear. So this formulation is general.

From now on, we simply denote $W(t)$ as W, $b(t)$ as b, and so on. Without loss of generality, we suppose A is a full rank matrix (i.e., rank$(A) = m$). When rank$(A) = m' < m$ (i.e., rank deficient), we can find a maximum linearly independent subset of row vectors in A. These row vectors constitute a new matrix A' (rank$(A') = m$) with the corresponding new vector b'. Then the equality constraint is equivalent to

$$A'x = b'$$

where A' is a full rank matrix.

Consider (1) as the primal problem; then its dual problem is

$$\begin{aligned} &\text{maximize} \quad b^T y - \frac{1}{2}x^T W x + l^T v - h^T w \\ &\text{subject to} \quad Wx + c - A^T y - E^T v + E^T w = 0 \end{aligned} \tag{2}$$

where $y \in \mathbb{R}^m$, $v, w \in \mathbb{R}^p$ are dual decision variables.

Define $u := v - w$. The equality constraint in (2) becomes

$$Wx + c - A^T y - E^T u = 0.$$

[1]Hereafter, $W(t) \in \mathbb{S}^n_{++}$ means that $W(t)$ is symmetric and positive definite.

According to the Karush–Kuhn–Tucker (KKT) conditions for convex optimization [2], the following set of equations has the same solution as the primal problem (1) and its dual (2):

$$\begin{aligned} &Wx + c - A^T y - E^T u = 0 \\ &Ax = b \\ &\begin{cases} (Ex)_i = l_i, & \text{if } u_i > 0 \\ (Ex)_i = h_i, & \text{if } u_i < 0 \\ l_i \le (Ex)_i \le h_i, & \text{if } u_i = 0. \end{cases} \end{aligned}$$

That is

$$Wx + c - A^T y - E^T u = 0 \tag{3}$$

$$Ax = b \tag{4}$$

$$Ex = g(Ex - u) \tag{5}$$

where $g(z)$ is a piecewise linear function, defined as

$$g(z_i) = \begin{cases} l_i, & \text{if } z_i < l_i \\ z_i, & \text{if } l_i \le z_i \le h_i, \\ h_i, & \text{if } z_i > h_i. \end{cases} \quad i = 1, \ldots, p$$

If $l = -\infty$, $g(z_i)$ degenerates to

$$g(z_i) = \begin{cases} z_i, & \text{if } z_i \le h_i \\ h_i, & \text{otherwise}, \end{cases} \quad i = 1, \ldots, p.$$

If $h = \infty$, $g(z_i)$ degenerates to

$$g(z_i) = \begin{cases} z_i, & \text{if } z_i \ge l_i; \\ l_i, & \text{otherwise}, \end{cases} \quad i = 1, \ldots, p.$$

If $l = -\infty$ and $h = \infty$, $g(z_i)$ degenerates to a linear function

$$g(z_i) = z_i.$$

From (3), because W is invertible, we can get

$$x = W^{-1}(A^T y + E^T u - c). \tag{6}$$

Substituting (6) into (4)

$$AW^{-1}(A^T y + E^T u - c) = b.$$

Because A is of full rank and W is invertible, $AW^{-1}A^T$ is invertible. So y can be explicitly expressed by u as

$$y = (AW^{-1}A^T)^{-1}[-AW^{-1}E^T u + AW^{-1}c + b]. \tag{7}$$

Based on (5)–(7), by the projection theorem, the dynamic equation of the proposed neural network for solving the primal problem (1) can be designed as:

- state equation

$$\epsilon\frac{du}{dt} = -Ex + g(Ex - u) \tag{8}$$

- output equation

$$\begin{aligned} x &= W^{-1}(A^T y + E^T u - c) \\ y &= (AW^{-1}A^T)^{-1}[-AW^{-1}E^T u + AW^{-1}c + b] \end{aligned} \tag{9}$$

where $u \in \mathbb{R}^p$ is the state vector and $\epsilon > 0$ is a scaling parameter that controls the convergence rate of the neural network. It can be rewritten in the more compact and explicit form as:

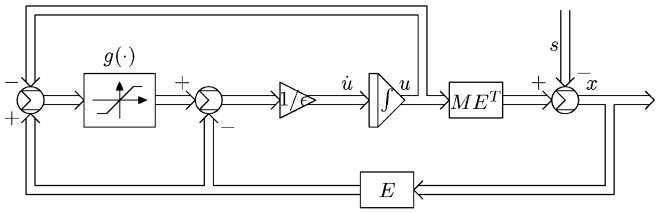


Fig. 1. Block diagram of the simplified dual neural network.

TABLE I
COMPARISON OF ARCHITECTURE COMPLEXITY AMONG VARIOUS NEURAL NETWORKS FOR QUADRATIC PROGRAMMING

Neural Network Type	Number of Neurons	Reference
Simplified Dual Neural Network	p	this paper
Lagrange Neural Network	$n+m+4p$	[17]
Prime-Dual Neural Network	$n+m+2p$	[19]
Dual Neural Network	$m+p$	[21]

- state equation

$$\epsilon\frac{du}{dt} = -EME^Tu + g((EME^T - I)u + Es) - Es \quad (10)$$

- output equation

$$x = ME^Tu + s \quad (11)$$

where $M := W^{-1} - W^{-1}A^T(AW^{-1}A^T)^{-1}AW^{-1}$, $s := W^{-1}(A^T(AW^{-1}A^T)^{-1}(AW^{-1}c+b)-c)$. Hereafter, we will call this neural network as the simplified dual neural network. Because the analytic expression of A, W, F can be obtained in the design stage, the analytic expressions of M and s can be computed beforehand. In view of this, it is not computationally complex although (10) appears complicated.

By defining $z = (EME^T - I)u + Es$, (10) can be rewritten as the following formulation similar to the Hopfield network:

$$\epsilon\frac{dz}{dt} = -EME^Tz + (EME^T - I)g(z) + Es$$

where z denotes the states of the neurons, $g(\cdot)$ is the nonlinear activation function, and $EME^T - I$ is a symmetric connection weight matrix.

The block diagram of the neural network is shown in Fig. 1, from which it can be observed that this is a one-layer network model. An important criterion of measuring the performance of recurrent neural networks is their computational complexity. Computational complexity deals with their computational power under constraints on resources. Here the limited resources include number of neurons, number of connections, and convergence time [7]. In the simplified dual neural network, the number of neurons is equal to the number of inequality constraints, whereas in the original dual neural network, the number of neurons is equal to the number of equality and inequality constraints [20], [21]. A complete comparison with several neural networks for solving the quadratic program problem (1) is shown in Table I. The convergence property is analyzed in the next section.

III. CONVERGENCE ANALYSIS

To study the optimization capability of the simplified dual neural network, its convergence property is first investigated. The relationship between the equilibrium point and the optimal solution is studied. A neural network is said to be globally convergent if starting from any state, the trajectory of the state converges to an equilibrium point. To analyze the convergence of the simplified dual neural network, three lemmas are first introduced.

Lemma 1 [23]: Assume that the set $\Omega \subset \mathbb{R}^m$ is a closed convex set; then the following two inequalities hold:

$$(P_\Omega(\alpha) - \beta)^T(\alpha - P_\Omega(\alpha)) \geq 0 \qquad \forall\alpha \in \mathbb{R}^m, \qquad \beta \in \Omega$$
$$\|P_\Omega(\beta) - P_\Omega(\alpha)\| \leq \|\beta - \alpha\| \qquad \forall\alpha, \qquad \beta \in \mathbb{R}^m$$

where $P_\Omega : \mathbb{R}^m \to \Omega$ is a projection operator defined as $P_\Omega(\gamma) = \min_{\zeta\in\Omega}\|\gamma - \zeta\|$.

Remark 1: It is clear that the set $\Omega := \{u \in \mathbb{R}^p | l \leq u \leq h\}$ and $g(u)$ satisfy the above projection property.

Lemma 2: For $\Lambda = \text{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ with $\lambda_i > 0$ $(i = 1, 2, \ldots, n)$, $P \in \mathbb{R}^{m\times n}$ $(m < n)$, $\text{rank}(P) = m$, the following matrix inequality holds[2] :

$$\Lambda - P^T(P\Lambda^{-1}P)^{-1}P \succeq 0.$$

Proof: Denote $P = [p_1\ p_2\ \cdots\ p_m]^T$. We can find $n-m$ column vectors $\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_{n-m}$, such that

$$p_i^T\Lambda^{-1}\tilde{p}_j = 0$$
$$\forall i, j\ (i = 1, 2, \ldots, m; j = 1, 2, \ldots, n-m).$$

Define $\tilde{P} := [\tilde{p}_1\ \tilde{p}_2\ \cdots\ \tilde{p}_{n-m}]^T$

$$\tilde{P}\Lambda^{-1}P^T = 0_{(n-m)\times m}$$
$$\Rightarrow (\tilde{P}\Lambda^{-1}P^T)^T = 0_{m\times(n-m)}$$
$$\Rightarrow P\Lambda^{-1}\tilde{P}^T = 0_{m\times(n-m)}.$$

Because $\Lambda = \text{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ is of full rank and $\text{rank}(P) = m$, then $p_1, p_2, \ldots, p_m, \tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_{n-m}$ are linearly independent vectors. Define $Q := \begin{bmatrix} P \\ \tilde{P} \end{bmatrix} \in \mathbb{R}^{n\times n}$; then Q is invertible. For $\forall y$, we have

$$\begin{aligned}
&y^T\Lambda y \\
&= y^TQ^T(Q^T)^{-1}(\Lambda^{-1})^{-1}Q^{-1}Qy \\
&= (Qy)^T(Q\Lambda^{-1}Q^T)^{-1}(Qy) \\
&= \left(\begin{bmatrix} P \\ \tilde{P} \end{bmatrix} y\right)^T \left(\begin{bmatrix} P \\ \tilde{P} \end{bmatrix} \Lambda^{-1}[P^T \quad \tilde{P}^T]\right)^{-1} \left(\begin{bmatrix} P \\ \tilde{P} \end{bmatrix} y\right) \\
&= \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix}^T \begin{bmatrix} P\Lambda^{-1}P^T & P\Lambda^{-1}\tilde{P}^T \\ \tilde{P}\Lambda^{-1}P^T & \tilde{P}\Lambda^{-1}\tilde{P}^T \end{bmatrix}^{-1} \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix} \\
&= \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix}^T \begin{bmatrix} P\Lambda^{-1}P^T & 0_{m\times(n-m)} \\ 0_{(n-m)\times m} & \tilde{P}\Lambda^{-1}\tilde{P}^T \end{bmatrix}^{-1} \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix} \\
&= \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix}^T \begin{bmatrix} (P\Lambda^{-1}P^T)^{-1} & 0_{m\times(n-m)} \\ 0_{(n-m)\times m} & (\tilde{P}\Lambda^{-1}\tilde{P}^T)^{-1} \end{bmatrix} \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix} \\
&= y^TP^T(P\Lambda^{-1}P^T)^{-1}Py + y^T\tilde{P}^T(\tilde{P}\Lambda^{-1}\tilde{P}^T)^{-1}\tilde{P}y.
\end{aligned}$$

[2]Hereafter, $H \succeq 0$ means matrix H is positive semidefinite and $H \succ 0$ means matrix H is positive definite.

Move $y^TP^T(P\Lambda^{-1}P^T)^{-1}Py$ to the left side

$$y^T\Lambda y - y^TP^T(P\Lambda^{-1}P^T)^{-1}Py = y^T\tilde{P}^T(\tilde{P}\Lambda^{-1}\tilde{P}^T)^{-1}\tilde{P}y. \tag{12}$$

Because $\Lambda^{-1} \succeq 0$

$$\tilde{P}\Lambda^{-1}\tilde{P}^T \succeq 0.$$

Then

$$\tilde{P}^T(\tilde{P}\Lambda^{-1}\tilde{P}^T)^{-1}\tilde{P} \succeq 0.$$

That is

$$y^T\tilde{P}^T(\tilde{P}\Lambda^{-1}\tilde{P}^T)^{-1}\tilde{P}y \geq 0 \qquad \forall y \in \mathbb{R}^n. \tag{13}$$

From (12) and (13)

$$y^T\Lambda y - y^TP^T(P\Lambda^{-1}P^T)^{-1}Py \geq 0 \qquad \forall y \in \mathbb{R}^n$$

i.e.,

$$\Lambda - P^T(P\Lambda^{-1}P)^{-1}P \succeq 0.$$

□

Lemma 3: Let $W \in \mathbb{S}^n_{++}$, $A \in \mathbb{R}^{m\times n}$ $(m < n)$, $\text{rank}(A) = m$, $E \in \mathbb{R}^{p\times n}$; then

$$E(W^{-1} - W^{-1}A^T(AW^{-1}A^T)^{-1}AW^{-1})E^T \succeq 0.$$

Proof: By the matrix spectrum theorem, because $W \in \mathbb{S}^n_{++}$, it can be decomposed as

$$W = T\Lambda T^T \tag{14}$$

where $\Lambda = \text{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ with $\lambda_i > 0\,(i = 1, 2, \ldots, n)$ as eigenvalues of W and T is an orthogonal matrix. Then, take the inverse of both side of (14)

$$W^{-1} = T\Lambda^{-1}T^T. \tag{15}$$

Define $P := AT \in \mathbb{R}^{m\times n}$ $(m < n)$, by Lemma 2

$$\Lambda - P^T(P\Lambda^{-1}P^T)^{-1}P \succeq 0.$$

$\Lambda^{-1} \succ 0$, so

$$I - P^T(P\Lambda^{-1}P^T)^{-1}P\Lambda^{-1} \succeq 0.$$

Furthermore

$$T\Lambda^{-1}(I - T^TA^T(AT\Lambda^{-1}T^TA^T)^{-1}AT\Lambda^{-1})T^T \succeq 0. \tag{16}$$

In view of (15), (16) is equivalent to

$$W^{-1} - W^{-1}A^T(AW^{-1}A^T)^{-1}AW^{-1} \succeq 0.$$

Then

$$E(W^{-1} - W^{-1}A^T(AW^{-1}A^T)^{-1}AW^{-1})E^T \succeq 0.$$

□

Theorem 1: The simplified dual neural network (10) is stable in the sense of Lyapunov and globally convergent to an equilibrium point u^*.

Proof: At u^*, we have the following:

$$(z - Ex^*)^Tu^* \geq 0, \qquad \forall z \in \Omega \tag{17}$$

where $\Omega := \{u \in \mathbb{R}^p \mid l \leq u \leq h\}$. This inequality can be obtained by considering the following three cases.

- If for some $i \in \{1, 2, \ldots, p\}$, $u_i^* = 0$, $l \leq [Ex^*]_i \leq h$, then $(z_i - [Ex^*]_i)u_i^* = 0$.
- If for some $j \in \{1, 2, \ldots, p\}$, $u_j^* > 0$, $[Ex^*]_j = l$ and $l \leq z_j \leq h$, then $z_j - [Ex^*]_j \geq 0$ and thus $(z_j - [Ex^*]_j)u_j^* \geq 0$.
- If for some $k \in \{1, 2, \ldots, p\}$, $u_k^* < 0$, $[Ex^*]_k = h$ and $l \leq z_k \leq h$, then $z_k - [Ex^*]_k \leq 0$ and thus $(z_k - [Ex^*]_k)u_k^* \geq 0$.

Therefore, by substituting $z = g(EME^Tu + Es - u)$, we can get from (17) that

$$[g(EME^Tu + Es - u) - (EME^Tu^* + Es)]^Tu^* \geq 0. \tag{18}$$

On the other hand, from Lemma 1, it follows that $\forall u \in \mathbb{R}^p$

$$\begin{aligned}&[g(EME^Tu + Es - u) - (EME^Tu^* + Es)]^T\\ &[(EME^Tu + Es - u) - g(EME^Tu + Es - u)] \geq 0.\end{aligned} \tag{19}$$

Combining (18) and (19), we have

$$\begin{aligned}&[g(EME^Tu + Es - u) - (EME^Tu^* + Es)]^T\\ &[u^* + (EME^Tu + Es - u) - g(EME^Tu + Es - u)] \geq 0.\end{aligned} \tag{20}$$

Defining $\tilde{g} := g(EME^Tu + Es - u) - (EME^Tu + Es)$, (20) becomes

$$[\tilde{g} + EME^T(u - u^*)]^T[(u - u^*) + \tilde{g}] \leq 0. \tag{21}$$

From (21), we can get

$$\begin{aligned}&(u - u^*)^T\tilde{g} + \tilde{g}^TEME^T(u - u^*)\\ &\qquad \leq -\|\tilde{g}\|^2 - (u - u^*)^TEME^T(u - u^*).\end{aligned} \tag{22}$$

According to Lemma 3, EME^T is positive semidefinite, i.e.,

$$(u - u^*)^TEME^T(u - u^*) \geq 0. \tag{23}$$

From (22) and (23), we get

$$(u - u^*)^T\tilde{g} + \tilde{g}^TEME^T(u - u^*) \leq 0 \tag{24}$$

and if and only if $u = u^*$, the equality holds.

Now choose the following radially unbounded Lyapunov functional candidate:

$$V(u(t)) = \frac{1}{2}\|Q(u(t) - u^*)\|_2^2 \tag{25}$$

where Q is a symmetric and positive definite matrix with $Q^2 = (I + EME^T)$.

Then from (24), we get

$$\begin{aligned}\frac{dV}{dt} &= (u-u^*)^T Q^2 \frac{du}{dt} \\ &= (u-u^*)^T (I+EME^T)\tilde{g} \\ &= (u-u^*)^T \tilde{g} + \tilde{g}^T EME^T (u-u^*) \\ &\le 0. \end{aligned} \tag{26}$$

By the Lyapunov stability theorem, the simplified dual neural network is globally stable. □

Theorem 2: $x^* = ME^T u^* + s$ is the optimal solution of the quadratic programming problem (1), where u^* is an equilibrium point of dynamic (10).

Proof: Since u^* is an equilibrium point of the dynamic (10)

$$g(EMEu^* + Es - u^*) - (EME^T u^* + Es) = 0.$$

By $x^* = ME^T u^* + s$

$$g(Ex^* - u^*) = Ex^*. \tag{27}$$

Define $y^* := -GE^T u^* + h$, $G := (AW^{-1}A^T)^{-1}(AW^{-1}$, and $h := (AW^{-1}A^T)^{-1}(AW^{-1}c + b)$; then

$$\begin{aligned} x^* &= ME^T u^* + s \\ &= W^{-1}(A^T(Gu^* + h) + E^T u^* - c) \\ &= W^{-1}(A^T y^* + E^T u^* - c). \end{aligned} \tag{28}$$

That is

$$Wx^* + c - A^T y^* - E^T u^* = 0. \tag{29}$$

Substituting G and h into the expression of y^*, we have

$$y^* = (AW^{-1}A^T)^{-1}[(-AW^{-1}E^T)u^* + AW^{-1}c + b].$$

$AW^{-1}A^T$ is invertible; then

$$AW^{-1}A^T y^* = (-AW^{-1}E^T)u^* + AW^{-1}c + b.$$

Leaving only b on the right side

$$AW^{-1}A^T y^* + AW^{-1}E^T u^* - AW^{-1}c = b$$

i.e.,

$$AW^{-1}(A^T y^* + E^T u^* - c) = b.$$

In view of x^* in (28), we have

$$Ax^* = b. \tag{30}$$

Equations (27), (29), and (30) constitute the KKT conditions (3)–(5) of (1). So $x^* = ME^T u^* + s$ is the optimal solution to the quadratic programming problem (1). □

From Theorems 1 and 2, we can conclude that the simplified dual neural network is globally convergent to the exact optimal solution of (1).

To show the convergence behavior of the neural network, let us consider a numerical example

$$\begin{aligned} \text{minimize} \quad & 3x_1^2 + 3x_2^2 + 4x_3^2 + 5x_4^2 + 3x_1x_2 + 5x_1x_3 \\ & + x_2x_4 - 11x_1 - 5x_4 \\ \text{subject to} \quad & 3x_1 - 3x_2 - 2x_3 + x_4 = 0 \\ & 4x_1 + x_2 - x_3 - 2x_4 = 0 \\ & -x_1 + x_2 \le -1, \\ & -2 \le 3x_1 + x_3 \le 4. \end{aligned}$$

Written in the standard form, the corresponding parameters are

$$W = \begin{bmatrix} 6 & 3 & 5 & 0 \\ 3 & 6 & 0 & 1 \\ 5 & 0 & 8 & 0 \\ 0 & 1 & 0 & 10 \end{bmatrix} \quad c = \begin{bmatrix} -11 \\ 0 \\ 0 \\ -5 \end{bmatrix}$$

$$A = \begin{bmatrix} 3 & -3 & -2 & 1 \\ 4 & 1 & -1 & -2 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$E = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 \end{bmatrix} \quad l = \begin{bmatrix} -\infty \\ -2 \end{bmatrix} \quad h = \begin{bmatrix} -1 \\ 4 \end{bmatrix}.$$

The simplified dual neural network for solving this quadratic programming problem needs only two neurons, whereas the Lagrange neural network [17] needs 12 neurons, the primal-dual neural network [19] nine neurons, and the dual neural network [21] four neurons.

The simulation results are shown in Figs. 2–5 with $\epsilon = 10^{-8}$. Figs. 2 and 3 illustrate, respectively, the convergence behaviors of variables u and x. From the initial state $[10, -10]^T$, the simplified dual neural network converges to the optimal solution $[0, -2, 2, -2]^T$ within 10^{-5} s. The norm of the difference between the optimal solution and the neural network output value at time instance 10^{-5} s is less than 3×10^{-10}. Figs. 4 and 5 show the state trajectories of the simplified dual neural network converging to the optimal solution from different initial states.

In this neural network, u is the state variable, x is the output, and the time-varying parameters W, b, c, E, l, and h are the inputs. In an electronic implementation, u is electric signal, which operates on ns − μs time scale, while the time-varying parameters usually change in ms − s time scale. With the two time-scale dynamics, the neural network can solve dynamic optimization problems as parametric problems. This is the rationale that we can use the electronic neural network to solve dynamic optimization problems. There are several advantages for this neural network compared with traditional parametric numerical methods.

1) It is a parallel solution, compared with traditional methods. With the parallelizable ability, it can tackle large-scale problems.
2) With the global convergence property, the neural network solution is convergent to the optimal solution no matter how the inputs change.
3) It can be implemented by electronic circuits, which means that the neural network can tackle practical problems directly with high speed to ensure online operations and produce continuous outputs.

IV. KWTA Application

In this section, an application of the simplified dual neural network for the KWTA operation is described. The KWTA operation is first converted to an equivalent constrained quadratic optimization problem. Then the simplified dual neural network

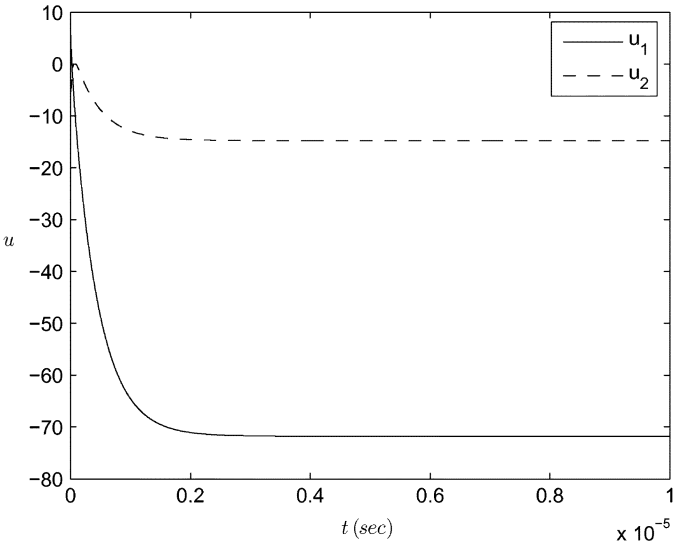


Fig. 2. Transient behaviors of u.

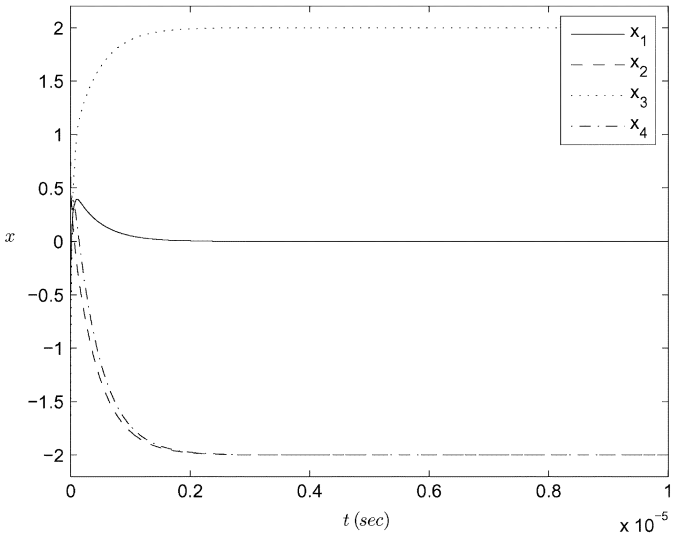


Fig. 3. Transient behaviors of x.

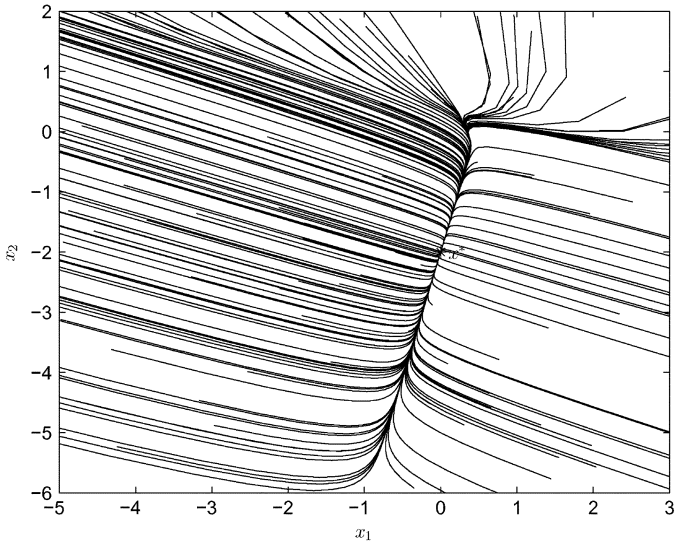


Fig. 4. Trajectories of x_1 and x_2 from different initial states.

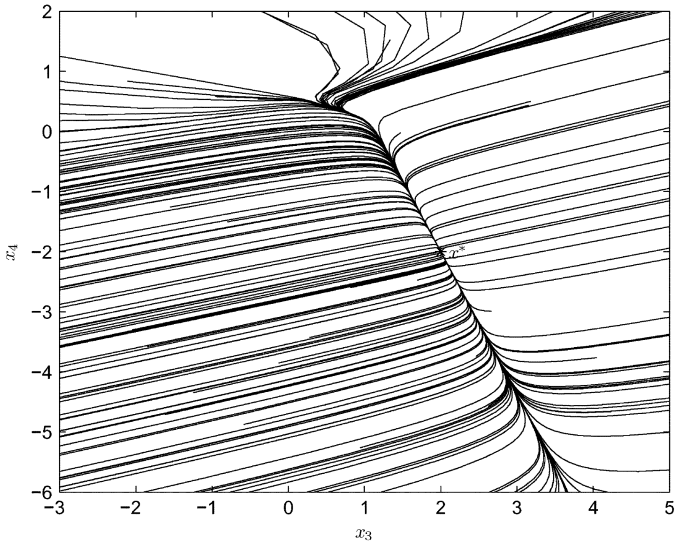


Fig. 5. Trajectories of x_3 and x_4 from different initial states.

is tailored to a KWTA network. The network has good scalability performance with $O(n)$ nodes and $O(n)$ interconnections. Simulation results are presented to show the effectiveness and performance of the KWTA network.

Winner-take-all (WTA) is an operation that identifies the largest value from the input signals. Such an operation has numerous applications including associative memories [25], cooperative models of binocular stereo [26], Fukushima's neocogniton for feature extraction, etc. [27]. In the combinatorial optimization, this operation is also needed.

As an extension of winner-take-all operation, KWTA selects the k largest inputs from n total inputs. It can be considered as a generalized version of winner-take-all operation. It has recently been shown that KWTA is computationally powerful compared with standard neural network models with threshold logic gates [28], [29]. Any boolean function can be computed by a single KWTA unit applied to weighted sums of input variables. Besides the applications in neural network model, the KWTA operation has important applications in machine learning, such as k-neighborhood classification, k-means clustering, etc. There have been many attempts to design VLSI circuits to do the KWTA operation [28]–[35].

A. Equivalent Reformulation

Mathematically, KWTA can be formulated as a function

$$x_i = f(v_i) = \begin{cases} 1, & \text{if } v_i \in \{k \text{ largest elements of } v\} \\ 0, & \text{otherwise.} \end{cases} \tag{31}$$

Fig. 6 shows the KWTA operation graphically.

To derive the equivalent formulation suitable for recurrent neural network design, two theorems are first given and proved.

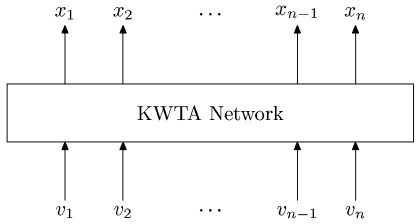


Fig. 6. Diagram of KWTA operation.

Theorem 3: The solution to the following discrete quadratic programming problem (32) is the same as the solution of (31):

$$\begin{aligned} &\text{minimize} \quad ax^Tx - v^Tx \\ &\text{subject to} \quad e^Tx = k, \qquad x_i \in \{0,1\}, \qquad i = 1,2,\ldots,n \end{aligned} \tag{32}$$

where a is a positive constant, $v := [v_1, v_2, \ldots, v_n]^T$, $x := [x_1, x_2, \ldots, x_n]^T$, $e := [1, 1, \ldots, 1]^T$.

Proof: $e^Tx = k$ can be written as

$$\sum_{i=1}^{n} x_i = k. \tag{33}$$

Because $x_i \in \{0,1\}$, we have

$$x_i^2 = x_i. \tag{34}$$

From (33) and (34), we get

$$\sum_{i=1}^{n} x_i^2 = k.$$

That is to say

$$ax^Tx = a\sum_{i=1}^{n} x_i^2 = ak$$

is a constant.

So the cost function can be rewritten as

$$\text{maximize} \quad v^Tx.$$

Suppose that the solution x^* of (32) is not the solution of (31). Without loss of generality, we assume that $x_i^* = 0$ with v_i in the top k largest inputs and $x_\ell^* = 1$, with v_ℓ not in the top k largest inputs. Because v_i is in the top kth largest inputs, whereas v_ℓ is not

$$v_i \geq v_\ell$$

where the equality holds if and only if the kth and $(k+1)$st largest inputs are equal, and $i = k, \ell = k+1$.) Define $\bar{x}_i := 1$, $\bar{x}_\ell := 0$, $\bar{x}_j := x_j^*$, $j \neq i, \ell$; then $\bar{x} = [\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n]^T$ also satisfies the constraints

$$\begin{aligned} v^T\bar{x} &= v_i\bar{x}_i + v_\ell\bar{x}_\ell + \sum_{j\neq i,\ell} v_j\bar{x}_j \\ &= v_i + \sum_{j\neq i,\ell} v_jx_j^* \\ &\geq v_\ell + \sum_{j\neq i,\ell} v_jx_j^* \\ &= v_ix_i^* + v_\ell x_\ell^* + \sum_{j\neq i,\ell} v_jx_j^* \\ &= v^Tx^*. \end{aligned}$$

This contradicts with the assumption that x^* is the solution of (32). □

Denote the kth largest element as $\tilde{v}_k$ and the $(k+1)$st largest element as $\tilde{v}_{k+1}$. Then we have the following theorem.

Theorem 4: If $\tilde{v}_k - \tilde{v}_{k+1} \geq 2a$, then the following continuous quadratic programming problem (35) and the discrete quadratic programming problem (32) have the same solution:

$$\begin{aligned} &\text{minimize} \quad ax^Tx - v^Tx \\ &\text{subject to} \quad e^Tx = k, \qquad x_i \in [0,1], \qquad i = 1,2,\ldots,n \end{aligned} \tag{35}$$

where a is a positive constant.

Proof: If we can show that the solution of the problem (35) is in the set $\{0,1\}^n$, then the theorem is proved.

From the equality constraint $e^Tx = k$, we can get

$$x_\ell = k - \sum_{j\neq\ell} x_j. \tag{36}$$

Substituting (36) into $ax^Tx - v^Tx$, we have

$$\begin{aligned} &ax^Tx - v^Tx \\ &\quad= a\left(\sum_{j=1}^{n} x_j^2\right) - \sum_{j=1}^{n} v_jx_j \\ &\quad= a\left(\sum_{j\neq\ell} x_j^2 + \left(k - \sum_{j\neq\ell} x_j\right)^2\right) \\ &\qquad - \sum_{j\neq\ell} v_jx_j - v_\ell\left(k - \sum_{j\neq\ell} x_j\right) \\ &\quad= a\left(x_i^2 + \cdots + x_i^2 - 2kx_i + 2x_i\sum_{j\neq\ell,i} x_j + \cdots\right) \\ &\qquad - v_ix_i + v_\ell x_i + \cdots \\ &\quad= 2ax_i^2 + \left(2a\sum_{j\neq\ell,i} x_j - 2ak + v_\ell - v_i\right)x_i + \cdots. \end{aligned}$$

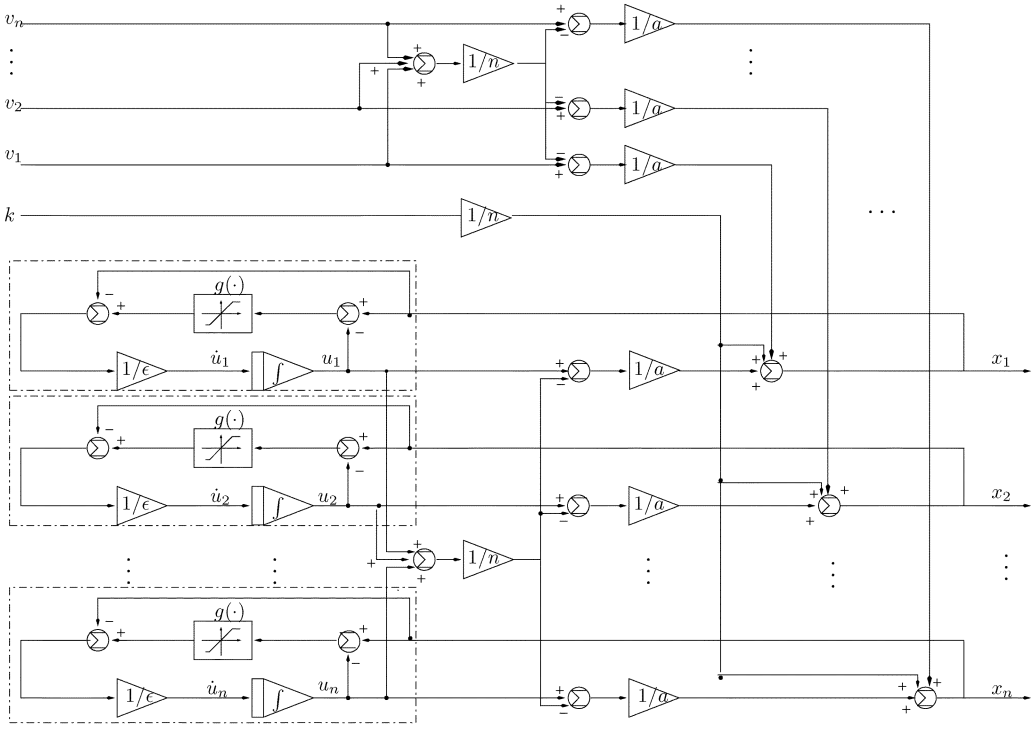


Fig. 7. Block diagram of the KWTA network.

From the previous derivation, we can take $ax^Tx - v^Tx$ as the function of variable x_i. It can be further written as

$$ax^Tx - v^Tx = 2ax_i^2 + \left(2a\sum_{j\neq\ell,i} x_j - 2ak + v_\ell - v_i\right)x_i + e(x) \quad (37)$$

where $e(x)$ has nothing to do with x_i and x_ℓ.

If the following condition is satisfied, $ax^Tx - v^Tx$ can only reach its minimum at its boundary, that is, $x_i = 0$ or $x_i = 1$:

$$\frac{v_i - v_\ell + 2a(k - \sum_{j\neq\ell,i} x_j)}{4a} \in (-\infty, 0] \cup [1, \infty).$$

Considering $k - \sum_{j\neq\ell,i} x_j = x_i + x_\ell$, the above condition is equivalent to

$$\frac{v_i - v_\ell + 2a(x_i + x_\ell)}{4a} \in (-\infty, 0] \cup [1, \infty). \quad (38)$$

If $v_i \in \{k \text{ largest elements of inputs}\}$, then we can choose $v_\ell \notin \{k \text{ largest elements of inputs}\}$. To let $x_i = 1$ and $x_\ell = 0$, the following conditions should be satisfied:

$$\frac{v_i - v_\ell + 2a(x_i + x_\ell)}{4a} \geq 1$$

and

$$\frac{v_\ell - v_i + 2a(x_\ell + x_i)}{4a} \leq 0.$$

We can get

$$v_i - v_\ell \geq 2a.$$

If $v_i \notin \{k \text{ largest elements of inputs}\}$, we can choose $v_\ell \in \{k \text{ largest elements of inputs}\}$. In the same way, we can get

$$v_\ell - v_i \geq 2a.$$

This means that, if $\tilde{v}_k - \tilde{v}_{k+1} \geq 2a$, the solution of (35) is in the set $\{0, 1\}^n$. □

From Theorems 3 and 4, we can easily see that if $\tilde{v}_k - \tilde{v}_{k+1} \geq 2a$, the solution to the continuous quadratic optimization problem (35) is the solution of (31).

Remark 2: Let us consider the solution of (35) when $\tilde{v}_k - \tilde{v}_{k+1} < 2a$. In this case

$$\frac{\tilde{v}_k - \tilde{v}_{k+1} + 2a(\tilde{x}_k + \tilde{x}_{k+1})}{4a} \in [0, 1].$$

Then, when

$$\tilde{x}_k = \frac{\tilde{v}_k - \tilde{v}_{k+1} + 2a(\tilde{x}_k + \tilde{x}_{k+1})}{4a}$$

the cost function reaches its minimum. That is

$$\tilde{x}_k - \tilde{x}_{k+1} = \frac{\tilde{v}_k - \tilde{v}_{k+1}}{2a}. \quad (39)$$

If other elements can be successfully separated, then

$$\tilde{x}_k + \tilde{x}_{k+1} = 1. \quad (40)$$

From (39) and (40), we can get

$$\tilde{x}_k = 0.5 + \frac{\tilde{v}_k - \tilde{v}_{k+1}}{4a} \quad \tilde{x}_{k+1} = 0.5 - \frac{\tilde{v}_k - \tilde{v}_{k+1}}{4a}.$$

In particular, if $\tilde{v}_k = \tilde{v}_{k+1}$, $\tilde{x}_k = \tilde{x}_{k+1} = 0.5$.

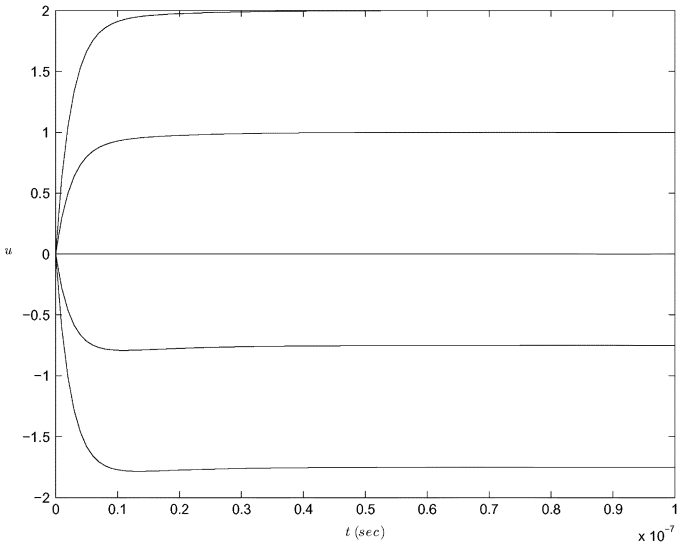


Fig. 8. Transient behavior of u of the KWTA network in Example 1.

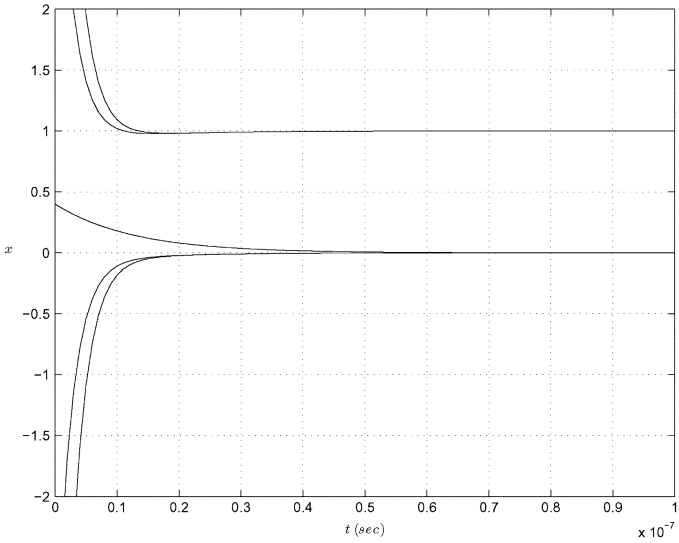


Fig. 9. Transient behavior of x of the KWTA network in Example 1.

B. KWTA Network Design

Based on the neural network (8) for solving quadratic programming (1) and the equivalent formulation (35) of KWTA operation, the KWTA network can be design as follows.

The KKT conditions of optimization problem (35) are

$$ax - v - ey - u = 0 \tag{41}$$

$$e^T x = k \tag{42}$$

$$x = g(x - u) \tag{43}$$

where

$$g(v_i) = \begin{cases} 0, & \text{if } v_i < 0 \\ v_i, & \text{if } 0 \leqslant v_i \leqslant 1 \\ 1, & \text{if } v_i > 1. \end{cases}$$

From (41)

$$x = \frac{1}{a}(ey + u + v). \tag{44}$$

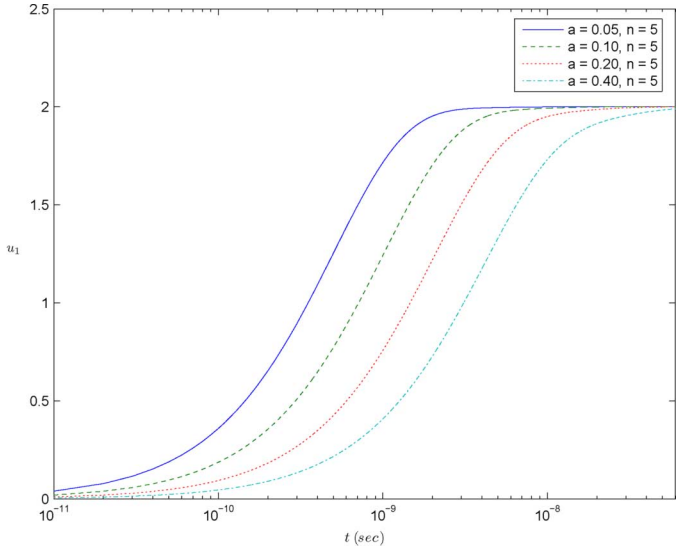


Fig. 10. Convergence behavior of the KWTA network with respect to different a in Example 1.

Substituting (44) into (42)

$$\frac{1}{a}e^T(ey + u + v) = k.$$

Then, y can be explicitly expressed by u

$$y = \frac{1}{n}(ak - e^T u - e^T v). \tag{45}$$

Substituting (45) into (44)

$$x = Mu + s \tag{46}$$

where $M := (I - ee^T/n)/a, s := Mv + (k/n)e$.

Then based on (43) and (46), the KWTA network can be designed as

$$\begin{aligned} \epsilon\frac{du}{dt} &= -Mu + g((M - I)u + s) - s \\ x &= Mu + s. \end{aligned} \tag{47}$$

Because M and $M - I$ are special square matrices with all the elements are identical except the diagonal ones, the above KWTA network can be implemented with $2n$ interconnections through utilizing a summator as shown in Fig. 7, which shows the architecture of the KWTA network. From the network model (47) and the architecture shown in Fig. 7, we can see that there are n neurons with $2n$ interconnections, where n is the number of inputs. This is in contract with most recurrent neural network, which is of full $(O(n^2))$ connectivity. From the scalability viewpoint, this means it can scale up easily to deal with large-scale applications in real implementation.

From the convergence study of the simplified dual neural network, we can get the result that the KWTA network is also globally Lyapunov stable and globally convergent to the solution of KWTA operation with resolution $2a$. Moreover, because the convergence speed of the KWTA network is dominated by the linear term of (47) (i.e., eigenvalues of M), which is independent of the number of inputs n, the convergence speed is independent of the problem scale. In addition, the eigenvalues of M

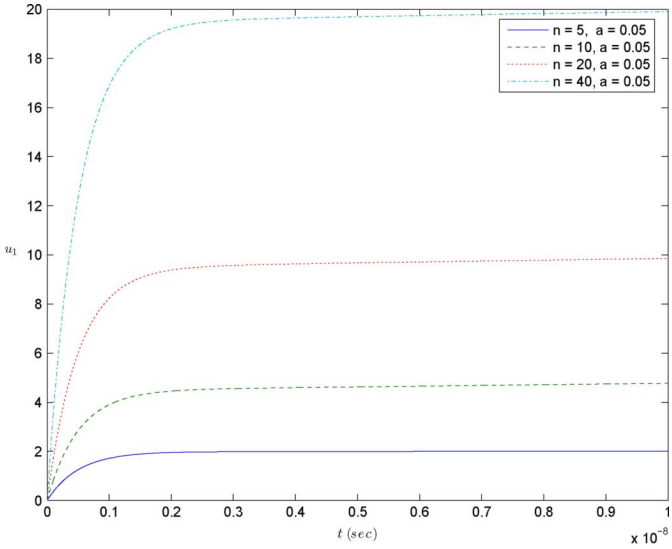


Fig. 11. Convergence behavior of the KWTA network with respect to different n in Example 1.

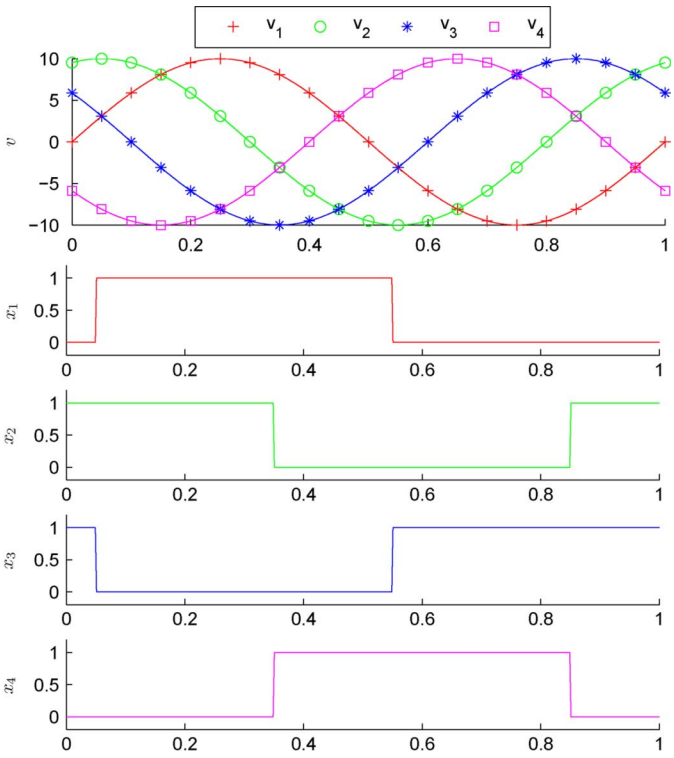


Fig. 12. Inputs and outputs of the KWTA network in Example 2.

are inversely proportional to a, so the convergence speed is also inversely proportional to a.

C. Simulation Results

Example 1: First, a static KWTA problem is tested. The inputs are $v_i = i\ (i = 1, 2, \ldots, n)$, $k = 2$, and $\epsilon = 10^{-8}$. When $n = 5$ and $a = 0.25$, the transient behaviors of u and v are shown in Figs. 8 and 9. In Fig. 9, the curves from bottom to top correspond, respectively, $x_1, x_2, \ldots, x_5$. It can be seen that the steady outputs are $[0\ 0\ 0\ 1\ 1]$. The two largest elements are successfully selected within 5×10^{-8} s. In Figs. 10 and 11, the relationship between the network's convergence rate and its parameters is shown with the first state variable u_1. In Fig. 10, it can be observed that when the parameter a increases exponentially, the convergence time also increases exponentially (note that the horizontal axis is in log scale). On the contrary, the convergence rate remains steady with respect to the changing of problem scale n, which can be observed in Fig. 11.

Example 2: Next, consider a set of four sinusoidal input signals with the following instantaneous values: $v_p(t) = 10\sin[2\pi(1000t + 0.2(p-1)]$ $(p = 1, 2, 3, 4)$ and $k = 2$. In this case, the inputs are time-varying and the corresponding quadratic optimization problem is also time-varying. Fig. 12 illustrates the five input signals and the transient outputs of the KWTA network with $\epsilon = 10^{-6}$ and $a = 10^{-3}$. The simulation results show that the KWTA network can effectively determine the two largest signals from the time-varying signals in real time.

V. Concluding Remarks

A simplified dual neural network is developed for solving strictly convex quadratic optimization problems with both equality and inequality constraints. The neural network is shown to be globally convergent to the optimal solutions. Compared with existing neural networks for optimization, the neural network has lower architecture complexity. An equivalent quadratic optimization formulation for the KWTA operation is given. Based on the simplified dual neural network, a KWTA network is designed and simulated. The designed KWTA network has good scalability performance. It has n neurons and $2n$ connections, and its convergence rate is independent of problem scale.

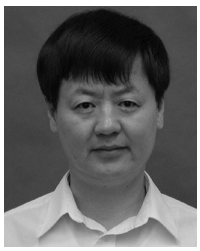
References

[1] S. J. Wright, *Primal-Dual Interior-Point Methods*. Philadelphia, PA: SIAM, 1997.
[2] S. Boyd and L. Vandenbeghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[3] P. Laarhoven and E. Aarts, *Simulated Annealing: Thoery and Applications*. Norwell, MA: Kluwer, 1987.
[4] D. Goldberg, *Genetic Algorithm in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[5] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*. New York: Wiley, 1993.
[6] C. Mead, *Analog VLSI and Neural Systems*. Boston, MA: Addison-Wesley, 1989.
[7] I. Parberry, *Circuit Complexity and Neural Networks*. Cambridge, MA: MIT Press, 1994.
[8] H. T. Siegelmann, *Neural Netowrks and Analog Computation: Beyond the Turing Limit*. Boston, MA: Birkhauser, 1999.
[9] S. Verdú, *Multiuser Detection*. New York: Cambridge Univ. Press, 1998.
[10] Y. Xia and J. Wang, "A one-layer recurrent neural network for support vector machine learning," *IEEE Trans. Syst., Man, Cybern.*, vol. 34, no. 2, pp. 1–10, Apr. 2004.
[11] R. Fantacci, M. Forti, M. Marini, D. Tarchi, and G. Vannuccini, "A neural network for constrained optimization with application to CDMA communication systems," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 50, no. 8, pp. 484–487, Aug. 2003.
[12] S. Liu and J. Wang, "Blind adaptive multiuser detection using a recurrent neural network," in *Proc. ICCCAS 2004*, Jun. 2004, pp. 1071–1075.
[13] J. Hertz, "Computing with attractors," in *The Handbook of Brain Theory and Neural Networks*, 2nd ed. Cambridge, MA: MIT Press, 2002, pp. 248–252.

[14] 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.
[15] J. J. Hopfield and D. W. Tank, "Computing with neural circuits: A model," *Science*, vol. 233, pp. 625–633, 1986.
[16] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Trans. Circuits Syst.*, vol. 35, no. 5, pp. 554–562, May 1988.
[17] S. Zhang and A. G. Constantinides, "Lagrange programming neural networks," *IEEE Trans. Circuits Syst. II*, vol. 39, no. 7, pp. 441–452, Jul. 1992.
[18] J. Wang, "A deterministic annealing neural network for convex programming," *Neural Netw.*, vol. 7, no. 4, pp. 629–641, 1994.
[19] 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.
[20] Y. Xia and J. Wang, "A dual neural network for kinematic control of redundant robot manipulators," *IEEE Trans. Syst., Man, Cybern.*, vol. 31, no. 1, pp. 147–154, Feb. 2001.
[21] Y. Zhang and J. Wang, "A dual neural network for convex quadratic programming subject to linear equality and inequality constraints," *Phys. Lett. A*, pp. 271–278, Jun. 2002.
[22] Y. Xia, G. Feng, and J. Wang, "A recurrent neural network with exponential convergence for solving convex quadratic program and linear piecewise equations," *Neural Netw.*, vol. 17, no. 7, pp. 1003–1015, 2004.
[23] D. Kinderlehrer and G. Stampacchia, *An Introduction to Variational Inequalities and Their Applications*. New York: Academic, 1980.
[24] C. Sun, K. Zhang, S. Fei, and C. Feng, "On exponential stability of delayed neural networks with a general class of activation functions," *Phys. Lett. A*, vol. 298, no. 2–3, pp. 122–132, Jun. 2002.
[25] A. K. J. Hertz and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Redwood City, CA: Addison-Wesley, 1991.
[26] D. Marr and T. Poggio, "Cooperative computation of stereo disparity," *Science*, vol. 195, pp. 283–328, 1977.
[27] A. L. Yuille and D. Geiger, "Winner-take-all networks," in *The Handbook of Brain Theory and Neural Networks*, 2nd ed. Cambridge, MA: MIT Press, 2002, pp. 1228–1231.
[28] W. Maass, "Neural computation with winner-take-all as the only nonlinear operation," *Adv. Neural Inf. Process. Syst.*, vol. 12, pp. 293–299, 2000.
[29] ——, "On the computational power of winner-take-all," *Neural Comput.*, vol. 12, pp. 2519–2535, 2000.
[30] W. J. Wolfe, "K-winner networks," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 310–315, Mar. 1991.
[31] 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.
[32] K. Urahama and T. Nagao, "K-winners-take-all circuit with O(N) complexity," *IEEE Trans. Neural Netw.*, vol. 6, no. 3, pp. 776–778, May 1995.
[33] B. Sekerkiran and U. Cilingiroglu, "A CMOS k-winners-take-all circuit with O(N) complexity," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 46, no. 1, pp. 1–5, Jan. 1999.
[34] Jayadeva and S. A. Rahman, "A neural network with O(N) neurons for ranking N numbers in O(1/N) times," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 10, pp. 2044–2051, Oct. 2004.
[35] 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.
[36] C. A. Marinov and B. Calvert, "Performance analysis for a k-winners-take-all analog neural network: Basic theory," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 766–780, Jul. 2003.
[37] 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.*, vol. 52, no. 5, pp. 949–959, May 2005.

Shubao Liu received the B.E. degree in information science and technology from the University of Science and Technology of China, Heifei, China, in 2003, the M.S. degree in automation and computer-aided engineering from the Chinese University of Hong Kong, Hong Kong, China, in 2005, and is currently working towards the Ph.D. degree in the Division of Engineering, Brown University, Providence, RI.

His research interests include machine learning, computer vision, and neural networks.



Jun Wang (S'89–M'90–SM'93) 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.

He is a Professor in the Department of Automation and Computer-Aided Engineering, Chinese University of Hong Kong, China. Before he joined this university in 1995, he was an Associate Professor at the University of North Dakota, Grand Forks. His current research interests include neural networks and their engineering applications.

Prof. Wang is an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS and IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, and was an Associate Editor of IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS (2001–2005). He is a past President of the Asia Pacific Neural Network Assembly (APNNA).