A New Ranking Model in DEA Based on Euclidean Distance and Solving It with Neural Network Approach

A. Ghomashi, F. Hosseinzadeh Lotfi, G.R. Jahanshahloo

Department of Mathematics, Science and Research Branch, Islamic Azad University, Tehran, Iran

ABSTRACT

One of the important concepts in Data Envelopment Analysis (DEA) is ranking. A drawback in almost of existing ranking models is inability to rank non-extreme units. In this paper, we proposed a model to rank extreme and non-extreme DMUs based on Euclidean distance from convex combination of full inefficient DMUs which first presented by Jahanshahlo and Afzalinejad[20]. A ranking method based on a full-inefficient frontier,” Applied Mathematical Modelling, vol.30, no. 3, pp. 248-260, 2006.]. Then, we use a novel neural network to solve proposed model for ranking. The validity of the proposed model for ranking and proposed neural network to solve it are considered by proof of lemma, theorems and solving numerical examples.

INTRODUCTION

DEA is a nonparametric approach in operations research first proposed by Charnes et al in 1978 to estimate the performance evaluation and relative efficiency of a set of homogeneous DMUs such as business units, government agencies, police departments, educational institutions and etc[17]. The seminal model of Charnes et al was called CCR model. Banker et al in 1984 developed a variable returns to scale version of the CCR model that was called BCC model[18]. DEA successfully divides DMUs into two categories: efficient DMUs and inefficient DMUs. DEA does this by assigning a relative efficiency score to each DMU such that the DMUs in efficient category have identical relative efficiencies equal to one and the rest have the relative efficiencies between zero and one. The researchers proposed some methods to difference DMUs in efficient category which this concept has named Ranking efficient units in the DEA. There are lots of ranking methods and each of them has special quality and property to rank efficient[21]. A drawback of existing models in ranking is inability to rank non-extreme units. A ranking method to rank non-extreme units is Jahanshahlo and Afzalinejad method[20]. They defined a full-inefficient frontier and ranked DMUs based on their distance from this frontier. In this paper, based on their work, we proposed a model to rank extreme and non-extreme DMU based on Euclidean distance from convex combination of full-inefficient DMUs. Proposed model is a special case of quadratic programming with linear constraints. We solve this model using a novel neural network approach. The neural networks are computing systems composed of a number of highly interconnected simple information processing units, when the dimension and denseness of the structure of optimization problem increases thus neural network can usually solve optimization problems in execution times at the orders of magnitude much faster than most popular optimization algorithms for general-purpose digital[5]. From a mathematical point of view, to formulate an optimization problem in terms of a neural network, there exist three types of methods[2], one approach commonly used in developing an optimization neural network is to first convert the constrained optimization problem into an associated unconstrained optimization problem, and then design a neural network that solves the unconstrained problem with gradient methods. Second approach is to construct a set of differential equations such that their equilibrium points correspond to the desired solutions and then find an appropriate Lyapunov function such that all trajectory of the systems converges to some equilibrium points. Last approach which proposed by Xia and Wang, combination of the above two type methods[2]. In this paper, to solve our proposed ranking model we are concerned with solving QP problem with lower model complexity than existing ones. Kennedy and chua[3] presented a neural network for solving the strict convex quadratic programming. Lately, many researchers...
successively proposed a number of models. In [4], Rodriguez-Vazquez et al. proposed a neural network for exactly solving linear and quadratic programming problems. In 1992, Maa and Shanblatt summarized and proved the computational capability of neural networks to LP and QP [5]. Xia [6], Xia and Wang [7-13], presented several neural networks for solving linear and quadratic programming problems, which have proved to be globally convergent to the exact solutions. In this paper, we propose a neural network model with two layer structure for solving our proposed ranking model. This paper is divided into six sections. In next section preliminary information is introduced to facilitate later discussions. In section III, we propose a quadratic model to rank efficient DMUs in BCC model based on definition of full-efficient DMUs. In section V, we proposed a neural network model to solve proposed ranking model and we analyze stability condition and global convergence proposed neural network model. In section V, some simulation examples are discussed. Section VI gives the conclusions of this paper.

2 Preliminaries:
2.1 DEA Preliminaries:
Suppose there are n DMUs with m inputs and s outputs. The input and output vectors of DMU(j = 1, ..., n) are
\[ X_j = (x_{j1}, ..., x_{jm})', Y_j = (y_{j1}, ..., y_{js})' \]
respectively, where \( X_j \geq 0, X_j \neq 0, Y_j \geq 0, Y_j \neq 0 \).

The PPS is represented as
\[ T = \{(X, Y) \in R^{m+s} | Y \text{ can be produced from } X\} \]
Banker et al. [18] inferred the following PPS for BCC model:
\[ T_{BCC} = \{(X, Y) \in R^{m+s} | X \geq \sum_{j=1}^{n} \lambda_j X_j, Y \leq \sum_{j=1}^{n} \lambda_j Y_j, \sum_{j=1}^{n} \lambda_j = 1, \lambda_j \geq 0 \} \]

The BCC model is used for an efficiency measure under the condition of variant returns to scale (VRS) [18].

Additive model in \( T_{BCC} \) is called the which is as follows [19]:
\[ z^*_{ADD} = \max \left( \sum_{i=1}^{m} S_i^- + \sum_{i=1}^{s} S_i^+, \sum_{j=1}^{n} \lambda_j X_j + S^- = X_p, \sum_{j=1}^{n} \lambda_j Y_j - S^+ = Y_p, \sum_{j=1}^{n} \lambda_j = 1 \right) \]
\[ \lambda_j \geq 0 \quad j = 1, ..., n \]

Where \( S^- \in R^m \) and \( S^+ \in R^s \).

**Definition 2.1** DMU \( p \) is ADD-efficient if and only if \( z^*_{ADD} = 0 \) [19].

**Definition 2.2** if DMU \( p \) is ADD-efficient then DMU \( p \) is called BCC-efficient otherwise DMU \( p \) is called BCC-inefficient [19].

**Definition 2.3** Reference set of DMU \( p \) is defined the following form [19]:
\[ \text{Ref}_p = \{ j | \lambda^*_j \neq 0 \text{ in one of optimal solutions of (1)} \} \]
Ref \( p \) helps us to construct a benchmark for DMU \( p \).

**Lemma 2.1** Let \( k \in \text{Ref}_p \) then DMU \( k \) is efficient DMU.

**Proof:** see [19]

**Definition 2.4** Suppose \( S \subseteq R^{m+s} \) as the convex hull of all the DMUs. The following set is called full-inefficient frontier [20].
\[ F(S) = \{(X, Y) \in S | \forall (X', Y') \in R^{m+s} ((X', Y') \cup (-X', Y') = S) \subseteq S \} \]
Each DMU belonged to \( F(S) \) is called a full-inefficient DMU.

2.2 Functions:

**Definition 2.5** A function \( F : R^m \rightarrow R^s \) is said to be Lipschitz continuous with constant \( L > 0 \) [14] if for each pair of points \( x, y \in R^m \)
\[ \| F(x) - F(y) \| \leq L \| x - y \| \]

**Definition 2.6** Let \( X = \{ x \in R^m | x_i \leq u_i, \forall i \in N = \{1, ..., m \} \} \). \( P_X : R^m \rightarrow X \) is called a projection operator and defined the following form [14]:
\( P_X(x) = \arg \min \| x - y \|, y \in X \)

Indeed \( P_X(x) \) is a projection of vector \( x \) to set \( X \). Since \( X \) is a box set, the projection operator \( P_X \) can be stated by
\[ P_X(x) = [P_X(x_1), ..., P_X(x_m)] \]
where $P_x(x_i)$ is a piecewise function. For $i \in \{1, \ldots, m\} \cap N$, $P_x(x_i) = x_i$ and for $i \in N$, $P_x(x_i)$ is defined the following form[14]:

$$
P_x(x_i) = \begin{cases}
  l_i, & x_i < l_i \\
  x_i, & l_i \leq x_i \leq u_i \\
  u_i, & x_i > u_i
\end{cases}
$$

**Lemma 2.2** Let $X \in \mathbb{R}^n$ be a closed convex set. Then

$\|P_x(x) - P_y(y)\| \leq \|x - y\|, \quad \forall x, y \in \mathbb{R}^n$

where $P_x(x)$ is a projection operator on $X$.

**Proof.** See[14].

**Definition 2.7** The finite-dimensional variational inequality problem $VI(F, K)$ is to determine a vector $x^* \in K \subseteq \mathbb{R}^n$ such that[14]

$$
F(x^*)^T(x - x^*) \geq 0, \quad \forall x \in K
$$

where $F$ is a given continuous function from $K$ to $\mathbb{R}^n$. $K$ is a given closed convex set.

**Theorem 2.1** Assume that $K$ is closed and convex. Then $x^* \in K$ is a solution of the variational inequality problem $VI(F, K)$ if and only if for any $\gamma > 0$, $x^*$ is a fixed point of the map $P_x(I - \gamma F): K \rightarrow K$ that is

$$
P_x(x^* - \gamma F(x^*)) = x^*
$$

**Proof:** see[14].

**Lemma 2.3** Let $U : D \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^n$ be differentiable on an open convex set $D$. Let $g: \mathbb{R}^n \rightarrow \mathbb{R}$ given by

$$
g(x) = \int_0^1 (x - x_0)^T U(x_0 + t(x - x_0))dt
$$

If the matrix of $U(x)$ is symmetric for all $x \in D$, then $\nabla g(x)^T = U(x)$ where $\nabla g(x)$ is the gradient of $g$.

**Proof:** see[1].

**Theorem 2.2** For all $y \in \mathbb{R}^n$ and all $x \in X \subseteq \mathbb{R}^n$

$$(y - P_x(y))^T(P_x(y) - x) \geq 0$$

**Proof.** See[14].

### 2.3 Differential Equation Preliminaries:

**Definition 2.8** Take the following dynamical system

$$
\dot{x} = f(x(t)), \quad x(t_0) = x_0 \in \mathbb{R}^n
$$

where $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$. $x^*$ is called an equilibrium point of (4) if [15]

$$
f(x^*) = 0.
$$

**Theorem 2.3** Assume that $f$ in (4) is a continuous mapping, then for arbitrary $t_0 \geq 0$ and $x_0 \in \mathbb{R}^n$ there exists a local solution $x(t)$ to (4) where $t \in [t_0, \tau]$ for some $\tau > t_0$. Furthermore if $f$ is locally Lipschitzian continuous at $x_0$ then the solution is unique, and if $f$ is Lipschitzian continuous in $\mathbb{R}^n$ then $\tau$ can be extended to $+ \infty$ [15].

**Theorem 2.4** Let $x^*$ be an equilibrium point of (4) and $X \subseteq \mathbb{R}^n$ be an open neighborhood of $x^*$, if $V: \mathbb{R}^n \rightarrow \mathbb{R}$ is a continuously differentiable function over $X$ and $V$ satisfies in the following conditions:

- $V(x^*) = 0, \quad \frac{dV(x^*)}{dt} = 0$
- $\frac{dV(x(t))}{dt} \leq 0, \quad V(x) > 0, \quad \forall x \in X - \{x^*\}$
- $\|x\| \rightarrow \infty \Rightarrow \|V(x)\| \rightarrow \infty$

then $x^*$ is a lyapunov stable equilibrium and the solution always exist globally, if
\[
dV(x(t)) \frac{dt}{dt} < 0 \quad \forall x \in X - \{x^*\}
\]
then \(x^*\) is a globally asymptotically stable equilibrium [15].

3  Ranking Model:
Based on full-inefficient DMU definition, we use the following model to identify full-inefficient DMU:

\[
z^*_p = \text{Max}\{\sum_{i=1}^{m} S_i^+ + \sum_{i=1}^{r} S_i^- | \sum_{j=1}^{n} \lambda_j X_j - S^- = X_p, \sum_{j=1}^{n} \lambda_j Y_j + S^+ = Y_p, \sum_{j=1}^{n} \lambda_j = 1\}, \quad \lambda_j \geq 0 \quad j = 1, \ldots n
\]  
(5)

Where \(S^- \in R^n\) and \(S^+ \in R^r\).

**Lemma 3.1** if \(z^*_p = 0\) then DMU \(p\) is full-inefficient DMU.

**Proof:** Based on definition 2.4 proof is complete.

**Lemma 3.2** Each convex combination of full-inefficient DMU is inefficient.

**Proof:** Based on definition 2.3 and lemma 2.1 proof is complete.

Let \(RI = \{ j \in \{1, \ldots, n\} | z_j = 0\}\) and \(RE = \{ j \in \{1, \ldots, n\} | w^*_{j,DO} = 0\}\), we proposed the following model to rank of efficient DMUs:

\[
z^*_p = \text{Max}\left\|\begin{bmatrix} X \\ Y \end{bmatrix} - \begin{bmatrix} X_p \\ Y_p \end{bmatrix}\right\|_2 \quad p \in RE
\]

s.t. \(\sum_{j \in RI} \lambda_j X_j = X\),
\[
\sum_{j \in RI} \lambda_j Y_j = Y,
\]
\[
\sum_{j \in RI} \lambda_j = 1,
\]
\[
\lambda_j \geq 0 \quad j = 1, \ldots n
\]

Regarding to (6), if \(z^*_p > z^*_q (p, q \in RI)\) then DMU \(p\) has better performance respect to DMU \(q\) since it has more distance from convex hull of full-inefficient DMUs. From Fig. 1 it can be seen that A, B and C are BCC-efficient and F, G, H are full-inefficient and \(Z^*\) presents Euclidean distance of DMUB from convex combination of full-inefficient DMUs.

**Fig. 1:** Geometry illustrative of proposed model for ranking

proposed model (6) is a special case of general form of quadratic programming with linear constraints which matrix in objective function is identity matrix. There exists models to solve it, in next section we proposed a novel neural network model to solve (6) with low complexity.

4  Neural Network Model:
Consider the following QP problem:
Min \( c^T x + \frac{1}{2} x^T x \)

S.t.
\[
\begin{align*}
    b_1 & \leq Ax \leq b_2 \\
    l & \leq x \leq u
\end{align*}
\]

where \( c, x \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, b_1 \) and \( b_2 \in \mathbb{R}^n \).

(7) is a special case of general form of quadratic programming which \( Q = I_{x} \) so \( Q \) in (7) is a symmetric and positive definite matrix. The Lagrangian function of (7) is the following form[16]:

\[
L(x, y, u) = c^T x + \frac{1}{2} x^T x - u^T (Ax - y)
\]

where \( x \in X = \{x | l \leq x \leq u\}, y \in Y = \{y | b_1 \leq y \leq b_2\} \) and \( u \in \mathbb{R}^n \).

**Definition 4.1** \((\bar{x}, \bar{y}, \bar{u})\) is called a saddle point of (8) if \( \bar{x} \in X \), \( \bar{y} \in Y \) and

\[
L(x, y, u) \leq L(\bar{x}, \bar{y}, \bar{u}) \leq L(x, y, u) \quad \text{for all } x \in X \text{ and } y \in Y \quad [16].
\]

**Lemma 4.1** \((\bar{x}, \bar{y}, \bar{u})\) is a saddle point of (8) if and only if \( \bar{x} \) are optimal solutions of (7).

**Proof:** see[16].

Substituting (8) into (9), we have

\[
\begin{align*}
    c^T \bar{x} + \frac{1}{2} (\bar{x}^T \bar{x} - u^T (A\bar{x} - \bar{y})) & \leq c^T \bar{x} + \frac{1}{2} ((\bar{x})^T \bar{x} - (\bar{u})^T (A\bar{x} - \bar{y})) \\
    & \leq c^T x + \frac{1}{2} x^T x - (\bar{u})^T (Ax - y)
\end{align*}
\]

for all \( x \in X \), \( y \in Y \) and \( u \in \mathbb{R}^n \). From the first inequality in (10) we have

\[
(u - \bar{u})^T (A\bar{x} - \bar{y}) \geq 0, \quad \forall u \in \mathbb{R}^n
\]

so we have

\[
A\bar{x} = \bar{y}
\]

Define

\[
f(x) = c^T x + \frac{1}{2} x^T x - (\bar{u})^T Ax
\]

From second inequality in (10) and definition of \( f(x) \) we have

\[
f(\bar{x}) - f(x) \leq (\bar{u})^T (y - \bar{y}) \quad \forall x \in X, \quad \forall y \in Y
\]

from the above inequality we have

\[
f(\bar{x}) - f(x) \leq 0, \quad \forall x \in X
\]

since if there exist a \( \bar{x} \in X \) such that \( f(\bar{x}) - f(\bar{x}) > 0 \) then we have \((\bar{u})^T (y - \bar{y}) > 0, \forall y \in Y\), which is contradictory when \( y = \bar{y} \), so we have

\[
(\bar{u})^T (y - \bar{y}) \leq 0, \quad \forall y \in Y
\]

Using (16), Definition 2.7 and theorem 2.1, we have \((\bar{y}, \bar{u})\) is the solution the following equation:

\[
y = P_x(y - u)
\]

On the other hand (15) implies that \( \bar{x} \in X \) minimizer of \( f(x) \) thus

\[
\forall f(\bar{x}) = \bar{x} + c - A^T \bar{u} = 0
\]

(12),(17) and (18) imply that \( \bar{x} \in X \) is optimal solution of (7) if and only if there exist \( \bar{y} \) and \( \bar{u} \) such that satisfying the following equations:

\[
\begin{align*}
    Ax - y & = 0 \\
P_x(y - u) - y & = 0 \\
x + c - A^T u & = 0
\end{align*}
\]

By substituting first equation into second equation in (19) we have

\[
\begin{align*}
P_x(Ax - u) - Ax & = 0 \\
x + c - A^T u & = 0
\end{align*}
\]
Lemma 4.2 \( \bar{x} \in X \) is optimal solution of (7) if and only if there exist \( \bar{u} \) such that \( (\bar{x}, \bar{u}) \) satisfies in (20).

Proof: Using lemma 4.1 and (21), proof is complete.

In this, Using lemma 4.2, we proposed the following recurrent neural network for solving (7):

- State equation
  \[
  \frac{du}{dt} = \lambda(P_t(Ax-u) - Ax) \tag{21}
  \]

- Output equation
  \[
  x = P_x(A'u - c) \tag{22}
  \]

where \( \gamma > 0 \) is scalar and . The system described by Eq. (21) can be applied for solving (7) and can be easily realized by a recurrent neural network. The proposed neural network can be implemented by using a simple hardware only without analog multipliers for the variables or the penalty parameter. The operator \( P_t \) and \( P_x \) may be implemented by using a piecewise activation function.

\[
\text{Fig. 2: Architecture of the recurrent neural network (21).}
\]

Corollary 4.1:

Right hand side of (21a) is Lipschitz continuous function.

Proof. Let the right hand side of (21a) be denoted by \( L(u) \) and \( h(u) = x = P_x(A'u - c) \) by lemma 2.2, for any \( \bar{u} \) and \( \bar{u} \), we have:

\[
\|L(\bar{u}) - L(\bar{u})\| = \|\lambda(P_t(\bar{A}'(\bar{u}) - \bar{A}h(\bar{u})) - \lambda(P_t(\bar{A}h(\bar{u}) - \bar{A}h(\bar{u})))\|
\]

\[
= \|\lambda(P_t(\bar{A}h(\bar{u}) - \bar{A}h(\bar{u}) - \lambda(P_t(\bar{A}h(\bar{u}) - \bar{A}h(\bar{u}))))\|
\]

\[
\leq \|\lambda(\bar{A}h(\bar{u}) - \bar{A}h(\bar{u}) + \bar{A}h(\bar{u}) - \bar{A}h(\bar{u})))\|
\]

Which gives the desired results.

Lemma 4.3 For each initial point \( u(t_0) \in R^n \), there exist a unique continuous solution \( u(t) \) \((t \in [t_0, \tau])\) for (21a) and the equilibria of neural network in (21a) correspond to unique optimal solution of (7).

Proof: Theorem 2.3 and corollary 4.1 yield a unique continuous solution \( u(t) \) over \( [t_0, \tau] \) exist for (21a). Since the feasible set of problem(7) is nonempty and objective function in (7) is strictly convex function so there exists unique optimal solution for (7), then using lemma 4.2 and (21a), proof is complete.

5 Convergence Analysis:

In this section, we prove globally asymptotically convergent of (21a).

Definition 5.1 The neural network in (21a) is said to be stable in the sense of Lyapunov and globally convergent, globally asymptotically stable, if the corresponding dynamic system is so.[1]

Theorem 5.1 The proposed neural network in (21a) is stable in the sense of Lyapunov and is globally asymptotically convergent to the unique solution of (7). Moreover, the convergence rate of the neural network in (21a) increase as \( \lambda \) increases.
Proof. By Lemma 4.3, we know that for each initial point \( u(t_0) \in R^n \), there exist a unique continuous solution \( u(t) \ (t \in [t_0, \tau]) \) for (21a). Let \( u' \) be equilibria point of (21a), define a lyapunov function below:

\[
V(u(t)) = \int_{t_0}^{t}(u(t') - u)'I(u' + s(u(t') - u'))dt', \quad \forall t \geq t_0,
\]

where \( I(.) \) is identity function. Let \( u = u(t) \), then time derivative of \( V \) along the trajectory of (21a) as follows:

\[
\frac{d}{dt} V(u) = \frac{dV}{du} \frac{du}{dt}
\]

on the other hand by theorem 2.2 we have:

\[
(y - P_{x}(y))(P_{x}(y) - u) \geq 0, \quad \forall y \in R^n, \forall u \in Y
\]

Let \( y = Ax - u \) and \( u = Ax \) then

\[
(Ax - u - P_{x}(Ax - u))'(P_{x}(Ax - u) - Ax) \geq 0
\]

where \( x = P_{x}(A^{'})u - c \) the above inequality rewritten as follows:

\[
I(u)'(P_{x}(Ax - u) - Ax) \leq -\|P_{x}(Ax - u) - Ax\|^{2}
\]

since the jacobian matrix of \( I(u) \) in (22) is symmetric, form lemma2.3 we have

\[
\frac{dV}{du} = I(u)
\]

Then

\[
\frac{dV}{du} = \lambda I(u)'(P_{x}(Ax - u) - Ax) \leq -\|P_{x}(Ax - u) - Ax\|^{2} \leq 0
\]

so we have

\[
\{u(t)|t_0 \leq t < \tau\} \subset P_{o} = \{u \in R^n \mid V(u) \leq V(u(t_0))\}
\]

since \( P_{o} \) is bounded and \( \{u(t)|t_0 \leq t < \tau\} \subset P_{o} \) \( u(t) \) is bounded and thus \( \tau = \infty \). Moreover

\[
\frac{dV(u)}{dt} = 0 \iff P_{x}(Ax - u) - Ax = 0 \iff \frac{du}{dt} = 0 \ (25)
\]

so by applying the theorem 2.4 and lemma 4.2, we get result that the proposed neural network is globally asymptotically convergent to the unique solution of (7).

Since \( \frac{dV}{dt} < 0 \) then we can result that as \( \lambda \) increases, the convergence rate of the neural network in (21a) increases. This proof is completed.

6 Illustrative Examples:

In this section, we demonstrate the effectiveness and performance of the proposed model for ranking and proposed neural network to solve it with two illustrative examples. The ordinary differential equation solver engaged in ode23 in matlab 2012.

Example 1 Fifteen DMUs were evaluated in terms of three inputs \((x_1, x_2, x_3)\) and three output \((y_1, y_2, y_3)\) that defined in table 1.

Table 1: Data set in Example 1

<table>
<thead>
<tr>
<th>DMUs</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( y_1 )</th>
<th>( y_2 )</th>
<th>( y_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>225935.0</td>
<td>405.0</td>
<td>1000.0</td>
<td>178295.0</td>
<td>4967.0</td>
<td>3588.0</td>
</tr>
<tr>
<td>2</td>
<td>102200.0</td>
<td>179.0</td>
<td>575.0</td>
<td>75526.0</td>
<td>3508.0</td>
<td>2093.0</td>
</tr>
<tr>
<td>3</td>
<td>94390.0</td>
<td>135.0</td>
<td>388.0</td>
<td>63968.0</td>
<td>5455.0</td>
<td>1246.0</td>
</tr>
<tr>
<td>4</td>
<td>51100.0</td>
<td>95.0</td>
<td>263.0</td>
<td>31835.0</td>
<td>1222.0</td>
<td>559.0</td>
</tr>
<tr>
<td>5</td>
<td>43800.0</td>
<td>43.0</td>
<td>186.0</td>
<td>19360.0</td>
<td>1112.0</td>
<td>373.0</td>
</tr>
<tr>
<td>6</td>
<td>31755.0</td>
<td>36.0</td>
<td>116.0</td>
<td>23372.0</td>
<td>2095.0</td>
<td>275.0</td>
</tr>
<tr>
<td>7</td>
<td>31390.0</td>
<td>28.0</td>
<td>99.0</td>
<td>17798.0</td>
<td>462.0</td>
<td>277.0</td>
</tr>
<tr>
<td>8</td>
<td>29930.0</td>
<td>31.0</td>
<td>159.0</td>
<td>14067.0</td>
<td>794.0</td>
<td>164.0</td>
</tr>
<tr>
<td>9</td>
<td>27714.0</td>
<td>41.0</td>
<td>226.0</td>
<td>15895.0</td>
<td>1130.0</td>
<td>366.0</td>
</tr>
<tr>
<td>10</td>
<td>23360.0</td>
<td>39.0</td>
<td>153.0</td>
<td>18127.0</td>
<td>1143.0</td>
<td>269.0</td>
</tr>
<tr>
<td>11</td>
<td>20440.0</td>
<td>31.0</td>
<td>94.0</td>
<td>148610.0</td>
<td>949.0</td>
<td>157.0</td>
</tr>
<tr>
<td>12</td>
<td>10585.0</td>
<td>15.0</td>
<td>67.0</td>
<td>4498.0</td>
<td>148.0</td>
<td>29.0</td>
</tr>
<tr>
<td>13</td>
<td>8030.0</td>
<td>15.0</td>
<td>62.0</td>
<td>8511.0</td>
<td>335.0</td>
<td>70.0</td>
</tr>
<tr>
<td>14</td>
<td>11680.0</td>
<td>26.0</td>
<td>110.0</td>
<td>9449.0</td>
<td>435.0</td>
<td>161.0</td>
</tr>
<tr>
<td>15</td>
<td>11315.0</td>
<td>20.0</td>
<td>92.0</td>
<td>4797.0</td>
<td>71.0</td>
<td>53.0</td>
</tr>
</tbody>
</table>
The results of proposed ranking model, non radial AP model[21](nr-AP) and jahanshahloo-afzalinejad ranking model[20](JA) is provided in table 2. Fig.3 shows that transient behavior of the neural network of (21a) in terms of $w(t)$. As can be seen from Fig. 3, the proposed neural network model is globally convergent to the optimal solution.

**Fig. 3:** Transient behavior of the neural network of (21a) in terms of $w(t)$ in example 1

**Table 2:** Comparison of three models in Example 1

<table>
<thead>
<tr>
<th>DMUs</th>
<th>Rank based on our model</th>
<th>nr-AP</th>
<th>JA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

**Example 2** The inputs and outputs of seven DMUs which each DMU consumes two inputs $(x_1, x_2)$ to produce four outputs $(y_1, y_2, y_3, y_4)$ is presented in table 3.

**Table 3:** Data set in Example 2

<table>
<thead>
<tr>
<th>DMUs</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y_1$</th>
<th>$y_2$</th>
<th>$y_3$</th>
<th>$y_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>2</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>3.0</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>3</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>3.0</td>
<td>2.0</td>
</tr>
<tr>
<td>4</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>5</td>
<td>1.0</td>
<td>1.0</td>
<td>2.0</td>
<td>2.5</td>
<td>3.5</td>
<td>2.0</td>
</tr>
<tr>
<td>6</td>
<td>1.0</td>
<td>3.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>4.0</td>
</tr>
<tr>
<td>7</td>
<td>2.0</td>
<td>1.0</td>
<td>2.0</td>
<td>2.5</td>
<td>2.25</td>
<td>3.0</td>
</tr>
</tbody>
</table>

The results of running the proposed ranking model, (nr-AP) and (JA) are summarized in table 4. Fig. 4 shows that transient behavior of the neural network of (9) in terms of $w(t)$ which is globally convergent to the optimal solution.
Fig. 4: Transient behavior of the neural network of (21a) in terms of $w(t)$ in example 2

<table>
<thead>
<tr>
<th>DMUs</th>
<th>Rank based on our model</th>
<th>nr-AP</th>
<th>JA</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Conclusion:

In this paper, a new quadratic model introduced to rank non-extreme and extreme efficient units in DEA. Proposed model is based on full-inefficient unit in DEA. Then, we proposed a novel neural network model to solve quadratic programming with linear constraints which in objective function, $Q=I$, and apply to solve proposed quadratic model. The neural network model in this paper has low complexity respect to similar existing model. It is shown here that the proposed neural network is stable in the sense of Lyapunov and globally convergent to the optimal solutions. Finally, examples are provided to show the effectiveness of the proposed model for ranking and neural network model to solve proposed quadratic model.

REFERENCES


