Using Hopfield Neural Networks to Solve DEA Problems

Shing-Cheng Hu, Yun-Kung Chung, Yun-Shiow Chen
Department of Industrial Engineering and Management
Yuan Ze University
Chung Li, Taiwan
ieychung@saturn.yzu.edu.tw

***Abstract*—Theory and application of both artificial neural networks (ANNs) and data envelopment analysis (DEA) have gone through major growth over the past three decades; nevertheless, using ANN as an optimal algorithm for finding the DEA solution has been limited. In this paper, a Hopfield neural network is applied as a solution tool to DEA models. An illustrative example from a known DEA problem helps to gain insight into the proposed alternative DEA solution method, including its capability and limitations.**

***Keywords*—data envelopment analysis, Hopfield neural networks, optimization, Lagrange function.**

I. INTRODUCTION

DEA was first developed by Charnes, Cooper and Rhodes in 1978 [1], and since then it has mostly been used to benchmark or evaluate the relative operating efficiencies of DMUs (decision-making units) composed of a unified system being assessed. The major advantages of DEA are its allowing the relative efficiency to change over time and requiring no prior assumption on the best solution frontier; therefore, lots of businesses or organizations have applied DEA to find their operating performances for further making decisions on the efficiency improvement. Those inefficient DMUs can be identified and proposed to make up their input resources and/or generated benefits. The identification has been performed widely by linear programming (LP) technique [2, 3, 4]; yet, the serious dependence on the number of DMUs causes the LP technique to a longer DEA computation time. To overcome this limitation, an alternative approach seems to be needed.

It has been well known that ANN is a better optimization technique than a classical optimal search method because of its virtues of inherent parallelism, fault tolerance, robust solution, learning ability and fast real-time solution. Using Hopfield neural networks (HNNs) to perform an LP procedure exactly is one of such examples of taking advantage of the ANN's optimal training procedure to solve classical optimization problems effectively [5, 6, 7]. This ANN-based LP method inspired this paper to conceive the conception of solving a DEA problem by means of the HNN optimal learning algorithm.

The paper is organized as follows. A brief review of previous works about using ANN to DEA is given in the following section. Sections 3 and 4 respectively provide a summary of DEA and HNN approaches. Development of HNN-based DEA using the Lagrange function is described in Section 5. A numerical DEA example is the focus of Section 6. Conclusions and thoughts for future research are addressed in Section 7.

II. PREVIOUS STUDIES ON DEA WITH ANN

There have been several articles presenting the combination of DEA and ANN to solve efficiency evaluation problems. Wu et al. [8] employed back-propagation neural networks (BPNs) to learn the data generated by DEA formulated, and then the trained BPN predicted the bank efficiency. Their method was impractical because the efficiency solutions had been obtained by DEA, there was unnecessary for sending them to BPN to do the efficiency prediction again. Like Wu et al. [8], Wang [9] also took BPN to learning the data of frontier obtained from CCR model, and then the trained BPN was used to predict DMU efficiency. Both papers were done with similar idea and similar discussions and conclusions on the BPN-based DEA solution. Still, they are impractical for performing the DEA evaluation because the BPN training data acquisition was tedious and the optimal DMU solution had been firstly found by CCR for a certain DEA problem. Using the trained BPN to predict the same DEA problem for estimating its approximated efficiency seems to be unnecessary. Also, different DEA problems may not be tackled by the same trained BPN, and the generalized DEA solution may not be an exact value of one for an efficient DMU. Pendharkar and Rodger [10] did the similar comparison of BPN to DEA as both Wu et al. [8] and Wang [9], but their DEA functioned as the data pre-processing for BPN training. The data first classified efficiency and inefficiency ones via DEA, then the two classified data were presented into BPN for its learning and efficiency classification. The solution quality of the both models was paralleled.

Costa and Markellos [11] depicted that both BPN and DEA were used to predict the public transport productive efficiency in which BPN was able to provide more information on the production function evaluation than DEA since BPN gets fault-tolerance structure and seldom statistical data assumptions. This paper, however, did not seem to analyze the DMU efficiency and inefficiency problems which is DEA essential, but to model a production function for the solution quality comparisons. Apart from the unclear DEA formulation, BPN training data definition was also not

presented. Athanassopoulos and Curram [12] compared BPN with DEA and found that BPN predicted the relative DMU ranks very well, whilst DEA was good at estimating empirical production functions. The comparisons were made with two experiments: one was based on simulated data from a known production function with different levels of inefficiencies and random noise, another on a multiple-input multiple-output data set of 250 commercial bank branches. Liao [13] did the similar comparison work as Athanassopoulos and Curram [12]. Three frontier methods, DEA, ANN and stochastic frontier analysis, were compared with the performance of DMUs. The result shown that BPN was a promising alternative to traditional approaches since it approximated production functions more accurately and evaluated efficiency and productivity under non-linear space with minimum assumptions.

To sum up, there are two sorts of combining ANN with DEA literatures, one focuses on the comparisons of the solution quality of production function of different frontier methods to BPN, which means that BPN is regarded as an individualized approach; another aims to use the laborious systematic DEA generating solutions to train BPN for generalizing an approximated DEA efficiency. The criticism on the both sorted methodologies was briefly presented above. A pragmatic idea of using ANN as an alternative to find the DEA solution may be to take advantage of ANN training algorithm to find DEA optimal solution, not to use the trained ANN to generate approximated DEA solutions. This paper will present such an idea of using HNN optimal algorithm to get DEA solution.

III. Introduction To DEA

As aforementioned, DEA is an LP-based technique to measure the relative efficiency of DMUs constituting a system being evaluated. Since the initiated DEA model, called CCR, named with the first letter of its developers' last names, Charnes, Cooper and Rhodes [1], was formulated in 1978, DEA has been widely applied to assess performance in diverse businesses and industries such as educational learning, hospital management, service process, manufacturing and production operations, personnel productivity, and so on. The reasons why DEA has been employed as a so popular tool include its ability of dealing with multiple (resources) inputs and multiple (production) outputs allowed to be expressed in different DMUs of measurement, and another is its capability of comparing a virtual DMU which is formed by linearly combining all original DMUs to a single original DMU by means of finding an optimal efficiency score of each original DMU. These efficiency scores (the points at the coordination of input and output) define the production possibility set of which a subset of its boundary points forms the efficient frontier and the remaindering inefficient DMUs are enveloped by this frontier where possesses substantial information on their inefficiency improvement potential. The degree of the improvement potential is determined by comparing a single original DMU to the linear combination of all DMUs located on the efficient frontier that utilize the same degree of inputs and produce the same or a higher degree of outputs [14].

There are various kinds of DEA model. In principle, they can be classified by two conceptions. One is based on whether the consideration of all DMU efficiencies is a type of input or output. An input-oriented DEA model thinks of efficiency as the least resources input for the same amount of production output, whereas an output-oriented DEA model regards efficiency as the most production output for the same amount of resources input. If the relative DMU is better than an assessing DMU by either making more output with the same input or making the same output with less input, then the assessing DMU is inefficient.

Another conception is according to the scale of optimal DMU score. This optimal score scale, called returns to scale (RTS) referring to increasing or decreasing efficiency based on size in terms of output/input ratio, discriminates DEA models into four types. If a DMU's production increases, its efficiency could be increased, remained constantly, decreased or varied, thus, creating Increasing Returns to Scale (IRS), Constant Returns to Scale (CRS), Decreasing Returns to Scale (DRS) or Variable Returns to Scale (VRS), respectively. VRS refers to a situation where both an increase and a decrease in RTS are observed at different degrees of production output, which means that not all DMUs operate on an optimal scale. CRS means that DMU is able to scale its inputs and outputs without increasing or decreasing efficiency; in other words, if the scale is up (increased) or down (decreased) with the DMU inputs and outputs, the DEA model respectively is IRS or DRS. In this paper, an input-oriented CCR model is demonstrated as a LP-based DEA of which formulation is solved by Hopfield neural network (HNN). This alternative solution method to CCR is presented later. As for more discussions about DEA, two textbooks are worthy of reading, Cooper, et al. [15] and Charnes, et al. [16], and a good survey paper described by Avkiran [17]. Certain computational numeric comparisons of the DEA solutions based on various LP algorithms were investigated in [18] and [19]. In what follows, CCR is depicted for the preparation of its HNN optimal training process.

CCR is an efficiency assessment tool that assumes non-negative of all inputs and outputs. The CCR model given the following (1) enables the search of efficient points on the frontier of DMU^o being assessed.

$$\begin{aligned} \text{minimize} \quad & E^o = \sum_{i=1}^{n} v_i x_{io} \\ \text{subject to} \quad & \sum_{r=1}^{s} u_r y_{ro} = 1 \\ & \sum_{i=1}^{n} v_i x_{ij} - \sum_{r=1}^{s} u_r y_{rj} \geq 0, \quad j = 1, 2, \ldots, m \\ & v_i \geq 0, \quad i = 1, 2, \ldots, n, \quad \forall i \\ & u_r \geq 0, \quad r = 1, 2, \ldots, s, \quad \forall r \end{aligned} \tag{1}$$

where E^o represents CCR efficiency of objective DMU^o, x_{ij} and y_{rj} represent the observed inputs ($i = 1, 2, \ldots, n$) and outputs ($r = 1, 2, \ldots, s$) of DMUs, $j = 1, \ldots, m$. $(\boldsymbol{v}^*, \boldsymbol{u}^*)$ obtained as an LP-based optimal solution for (1) results in a set of optimal weights for the distributed proportions of resource inputs and production outputs of DMU^o, respectively..

IV. INTRODUCTION TO HNN

There have been a great number of published papers taking advantage of ANN to formulate and solve optimization problems over the past 15 years. The earliest one could be presented in 1986 when a pioneer work was performed by Tank and Hopfield [20], who created the theoretical foundation of his neural networks with formulating and solving LP problems; a couple of years later, Kennedy [21, 22] extended Hopfield's LP model to nonlinear programming, and Maa [23] substantiated the theoretical foundation of the pioneered Hopfield neural network. Since then, a lot of HNN theory and applications have been developed to now. Wang et al. [24] extended HNN to a stochastic high-order HNN that possesses stronger approximation property, faster convergence rate, greater storage capacity, and higher fault tolerance than lower-order HNN. Linear matrix inequalities were solved with the high-order HNN to demonstrate its global asymptotic convergence of the equilibrium point.

Guo et al., [25] applied HNN to solve an optimization problem of power partitioning of real-time computer operating system which is a critical component in the system-on-a-chip. The power partitioning optimization is to minimize the energy consumption of the chip. HNN was formulated with a new energy function, operating equation and coefficients based on the power consumption factors and performed by its optimal learning algorithm. The HNN simulation showed the better optimal solution than other optimization methods. Readers who are interested in ANN can refer to [26, 27, 28]. The following HNN introduction was copied from Cochicki and Unbehauen [29] for the preparation of HNN-based DEA development later. They detailed HNN approaches for optimization problems in their particular and professional textbook addressing and developing the theory and applications of HNN-based optimization.

The architecture of HNN differs from BPN. Its structure is a class of recurrent one in which the dynamics are non-trivial and plays an important role. The dynamics of HNN is described by a system of nonlinear ordinary differential equations and by their corresponding computation energy, named Lyapunov function, which is minimized during the computation process. Figure 1 shows that HNN can be implemented by interconnecting an array of resistors, nonlinear amplifiers with symmetrical outputs, external bias current sources and n fully interconnected artificial neurons. The black dots at the current intersections stand for the interconnection weights between inputs and outputs. These weights are summed by superimposition of the resistor currents at the input onto each nonlinear amplifier. Mathematically, the HNN model can be derived from Kirchhoff's Current Law applied to each artificial neuron of the amplifier as:

$$C_j \frac{\mathrm{d}u_j}{\mathrm{d}t} = -\frac{u_j}{R_j} + \sum_{i=1}^{n} G_{ji} x_i + I_j \tag{2}$$

$$x_j = \psi_j(u_j), \quad (j = 1,2,\ldots,n) \tag{3}$$

where

$C_j > 0$ is the capacitance,

u_j and x_j are the internal and output voltages of the neuron j, respectively,

G_{ji} is the conductance representing the synaptic weight from neuron i to neuron j defined as

$$G_{ji} = \begin{cases} \dfrac{1}{R_{ji}^{+}} & \text{if} \quad G_{ji} > 0 \quad \text{with} \quad R_{ji}^{-} = \infty, \\ -\dfrac{1}{R_{ji}^{-}} & \text{if} \quad G_{ji} < 0 \quad \text{with} \quad R_{ji}^{+} = \infty, \end{cases} \tag{4}$$

$$\frac{1}{R_j} = \frac{1}{R_{j0}} + \sum_{i=1}^{n} \left[\frac{1}{R_{ji}^{+}} + \frac{1}{R_{ji}^{-}} \right] \tag{5}$$

and $\psi_j(u_j)$ is the nonlinear, differentiable, monotonically increasing activation function; typically it is defined as

$$\psi_j(u_j) = \left[1 + e^{-\gamma_j u_j}\right]^{-1} \quad \text{or} \quad \psi_j(u_j) = \tanh(\gamma_j u_j). \tag{6}$$

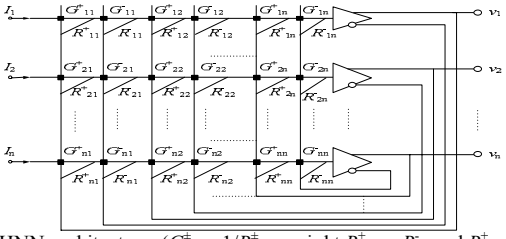

Figure 1. HNN architecture ($G_{ij}^{\pm} = 1/R_{ij}^{\pm}$, weight R_{ij}^{+} or R_{ij}^{-} and $R_{ii}^{+} = R_{ii}^{-} = \infty$)

The above system of differential equations can be written in the more convenient normalized form:

$$\tau_j \frac{\mathrm{d}u_j}{\mathrm{d}t} = -\alpha_j u_j + \sum_{i=1}^{n} w_{ji} \psi_i(u_i) + \Theta_j \quad (j = 1,2,\ldots,n) \tag{7}$$

where

$$\tau_j = r_j C_j, \quad \alpha_j = \frac{r_j}{R_j},$$

$$w_{ji} = r_j G_{ji}, \quad \Theta_j = r_j I_j,$$

r_j is the scaling resistance ($\alpha_j = 1$ in the special case $r_j = R_j$).

In HNN, the interconnection weights w_{ji} is symmetrical, i.e., $w_{ji} = w_{ij}$, with diagonal elements which equal zero, $w_{ii} = 0$. The set of equilibrium points of HNN is determined form the system of differential equations (7) by taking $\mathrm{d}u_j/\mathrm{d}t = 0$, i.e. from the set of nonlinear equations:

$$-\alpha_j u_j + \sum_{i=1}^{n} w_{ji} \psi_i(u_i) + \Theta_j = 0 \tag{8}$$

HNN has shown that a sufficient condition for its optimization stability, namely its convergence to a stable state. This stable state is the local minimum of HNN computational energy function

$$E(\mathbf{x}) = -\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{n} w_{ji} x_i x_j - \sum_{j=1}^{n} x_j \Theta_j + \sum_{j=1}^{n} \frac{\alpha_j}{\gamma_j} \int_0^{x_j} \Psi_j^{-1}(x)\,\mathrm{d}x \tag{9}$$

Simplifying this computational energy function can obtain the form

$$E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^{T}\mathbf{W}\mathbf{x} - \mathbf{x}^{T}\boldsymbol{\Theta} \tag{10}$$

The optimal (equilibrium or stable) points correspond exactly to the local minimum of the HNN's computational energy function. The operation of HNN can be regarded as a minimization computation process of the energy function *E(***x***)*. Lyapunov stability requires the *E(***x***)* to be monotonically decreasing in time. Consider the time derivative of *E(***x***)* given in (9)

$$\begin{aligned} -\frac{dE}{dt} &= \sum_{j=1}^{n}\frac{\partial E}{\partial x_j}\frac{dx_j}{dt} = -\sum_{j=1}^{n}\frac{dx_j}{dt}\left[\sum_{i=1}^{n} w_{ji}x_i + \Theta_j - \alpha_j u_j\right] \\ &= -\sum_{j=1}^{n}\tau_j\frac{dx_j}{dt}\frac{du_j}{dt} - \sum_{j=1}^{n}\tau_j\frac{du_j}{dx_j}\left[\frac{dx_j}{dt}\right]^2 \\ &= -\sum_{j=1}^{n}\tau_j\left[\Psi_j^{-1}(x_j)\right]'\left[\frac{dx_j}{dt}\right]^2 \end{aligned} \tag{11}$$

Since the time constant τ_j is positive for all *j* and the nonlinear inverse function $\Psi_j^{-1}(x_j)$ is monotonically increasing (i.e., greater than zero), (11) can be rewritten as

$$\frac{dE}{dt} \le 0 . \tag{12}$$

$dE/dt = 0$ implies $dx_j/dt = 0$ for all *j*. Combining this with the fact that the energy *E(***x***)* is bounded, HNN can converge to a stable state at the minimum of its *E(***x***)* It should be noted that the computational energy function is constructed as the first integral of the dynamic differential equations of (7); in other words, HNN is a gradient-like system described by the matrix differential equation

$$\frac{d\mathbf{u}}{dt} = -\boldsymbol{\mu}\nabla_{\mathbf{x}}E(\mathbf{x}), \tag{13}$$

where

$$\mathbf{u} = [u_1, u_2, \ldots, u_n]^{T},$$

$$\boldsymbol{\mu} = \mathrm{diag}\left(\tau_1^{-1}, \tau_2^{-1}, \ldots, \tau_n^{-1}\right),$$

$$\nabla_{\mathbf{x}}E(\mathbf{x}) = \left[\frac{\partial \mathbf{E}}{\partial x_1}, \frac{\partial \mathbf{E}}{\partial x_2}, \ldots \frac{\partial \mathbf{E}}{\partial x_n}\right]^{T}.$$

These energy (Lyapunov) functions are invaluable tools since they can be used to express complex ANN dynamics in the form of optimization process.

V. Development Of Lagrange Hnn-based Dea Approach

There a number of published papers have been addressed about HNN-based LP techniques, such as Grippo [30], Huang [31] and Wang [32]; however, there is a major weakness of using HNN to an optimization problem. A penalty function used in their HNNs possibly causes not to ensure the convergence of an optimal point and not to tackle a large number of variables for its optimality. Fortunately, Cichocki and Bargiela [33] proposed a Lagrange-based HNN optimization approach for solving the LP problem. In the earlier time, Zhang and Constantinides [34] also showed a Lagrange multiplier method-based HNN approach, but for a non-linear programming problem. These Lagrange-based HNN methods do not encounter the problems of infeasibility and physical implementation occurred in penalty-based HNN optimizations, and can always give a feasible solution. In this section, the development of Lagrange HNN-based for DEA is discussed.

The Lagrange function $L(\boldsymbol{v},\boldsymbol{u},\boldsymbol{s},\alpha,\boldsymbol{\beta})$ of CCR formulated in (1) can be established using the Lagrange multiplier method as follows.

Let

$$\sum_{r=1}^{s} u_r y_{ro} - 1 = r_0(\boldsymbol{u})$$

$$\sum_{i=1}^{n} v_i x_{ij} - \sum_{r=1}^{s} u_r y_{rj} - s_j^{+} = r_j\left[\boldsymbol{v}, \boldsymbol{u}, \boldsymbol{s}^{+}\right],\ j = 1, 2, \ldots, m$$

The Lagrange (HNN energy) function of (1) can be

$$\begin{aligned} \hat{L}(\mathbf{v},\mathbf{u},\mathbf{s}^{+},\boldsymbol{\beta}) &= \sum_{i=1}^{n} v_i x_{io} + \beta_0 \cdot G\left(\sum_{r=1}^{s} u_r y_{ro} - 1\right) \\ &\quad + \sum_{j=1}^{m}\beta_j \cdot G\left(\sum_{i=1}^{n} v_i x_{ij} - \sum_{r=1}^{s} u_r y_{rj} - s_j^{+}\right) \\ &= \sum_{i=1}^{n} v_i x_{io} + \beta_0 \cdot G\left(r_0(\mathbf{u})\right) \\ &\quad + \sum_{j=1}^{m}\beta_j \cdot G\left(r_j(\mathbf{v},\mathbf{u},\mathbf{s}^{+})\right) \end{aligned} \tag{14}$$

where α is an individual parameter and $\boldsymbol{\beta} = (\beta_1 \ldots \beta_m)^{T}$ is a parameter matrix. Taking partial derivate of equation (14) with respect to time, the following dynamic differential equations of (1), i.e., (13), can be derived:

$$\frac{dv_i}{dt} = -\mu_v\frac{\partial \hat{L}(\boldsymbol{v},\boldsymbol{u},\boldsymbol{s}^{+},\beta)}{\partial v_i}$$

$$\begin{aligned} \frac{\partial v_i(\mathbf{v},\mathbf{u},\mathbf{s}^{+},\beta)}{\partial v_i} &= x_{io} + \sum_{j=1}^{m}\beta_j \cdot \frac{\partial G_j\left(r_i(\mathbf{v},\mathbf{u},\mathbf{s}^{+})\right)}{\partial r_i(\mathbf{v},\mathbf{u},\mathbf{s}^{+})} \cdot \frac{\partial r_i(\mathbf{v},\mathbf{u},\mathbf{s}^{+})}{\partial v_i} \\ &= x_{io} + \sum_{j=1}^{m}\beta_j \cdot \psi\left(r_i(\mathbf{v},\mathbf{u},\mathbf{s}^{+})\right) \cdot x_{ij} \end{aligned}$$

$$\text{where}\quad \psi\left(r_i(\mathbf{v},\mathbf{u},\mathbf{s}^{+})\right) \approx \frac{\partial G_j\left(r_i(\mathbf{v},\mathbf{u},\mathbf{s}^{+})\right)}{\partial r_i(\mathbf{v},\mathbf{u},\mathbf{s}^{+})}$$

$$\frac{dv_i}{dt} = -\mu_i\left(x_{io} + \sum_{j=1}^{m}\beta_j \cdot x_{ij} \cdot \psi\left(r_i(\boldsymbol{v},\boldsymbol{u},\boldsymbol{s}^{+})\right)\right)$$

$$\frac{du_r}{dt} = -\mu_r\left(\beta_0 \cdot y_{ro} \cdot \psi(r_0(\boldsymbol{u})) - \sum_{j=1}^{m}\beta_j \cdot y_{rj} \cdot \psi\left(r_i(\boldsymbol{v},\boldsymbol{u},\boldsymbol{s}^{+})\right)\right)$$

$$\frac{ds_j^{+}}{dt} = -\mu_j\left(-\sum_{j=1}^{m}\beta_j \cdot \psi\left(r_i(\boldsymbol{v},\boldsymbol{u},\boldsymbol{s}^{+})\right)\right)$$

$$\frac{d\beta_j}{dt} = -\mu_{\beta_j} \cdot G\left(r_j(\boldsymbol{v},\boldsymbol{u},\boldsymbol{s}^{+})\right),\ j = 1, 2, \ldots, m$$

$$\frac{d\beta_0}{dt} = -\mu_{\beta_0} \cdot G\left(r_0(\boldsymbol{u})\right)$$

These dynamic differential equations can be solved by the fourth-order Runge-Kutta numeric method [35].

VI. A Numerical Lagrange Hnn-based Dea Example

To illustrate the solution performance of the proposed HNN-based DEA method, a commercial DEA package, DEA-Solver, was used to get a traditional LP-based DEA solution; meanwhile, the fourth-order Runge-Kutta implemented in MATLAB utility functions was utilized to solve the same DEA

example. Table 1 shows the example. Figure 2 shows a simplified HNN-based DEA network of the example.

TABLE 1. INPUT/OUTPUT DATA FOR EXAMPLE

DMU	x_1	x_2	y
A	1	1	1
B	2	1	1
C	3	2	1

The derivatives of HNN-based DEA for each DMU of the example are as follows.

1) case 1 for DMU^A:

a) the CCR original formulation for DMU^A:

$$\begin{aligned} \text{minimize} \quad & \phi_A = v_1 + v_2 \\ \text{subject to} \quad & u = 1 \\ & v_1 + v_2 - u \geq 0 \\ & 2v_1 + v_2 - u \geq 0 \\ & 3v_1 + 2v_2 - u \geq 0 \\ & v_1, v_2, u \geq 0 \end{aligned}$$

b) the standard LP formulation for DMU^A:

$$\begin{aligned} \text{minimize} \quad & \phi_A = v_1 + v_2 \\ \text{subject to} \quad & u = 1 \\ & v_1 + v_2 - u - s_1 = 0 \\ & 2v_1 + v_2 - u - s_2 = 0 \\ & 3v_1 + 2v_2 - u - s_3 = 0 \\ & v_1, v_2, u, s_1, s_2, s_3 \geq 0 \end{aligned}$$

c) energy function for DMU^A:

$$L_A = (v_1 + v_2) - \alpha(u-1) - \frac{1}{2}\beta_1(v_1 + v_2 - u - s_1)^2 - \frac{1}{2}\beta_2(2v_1 + v_2 - u - s_2)^2 - \frac{1}{2}\beta_3(3v_1 + 2v_2 - u - s_3)^2$$

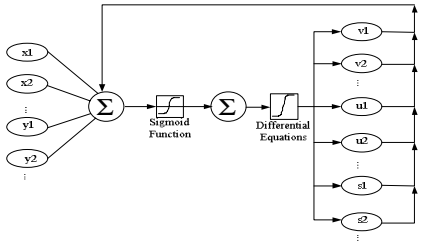

Figure 2. A simplified Lagrange HNN-based DEA network

A system of dynamic differential equations of HNN for DMU^A can be shown as:

$$\frac{\partial v_1}{\partial t} = 1 - \beta_1(v_1 + v_2 - u - s_1) - 2\beta_2(2v_1 + v_2 - u - s_2) - 3\beta_3(3v_1 + 2v_2 - u - s_3)$$

$$\frac{\partial v_2}{\partial t} = 1 - \beta_1(v_1 + v_2 - u - s_1) - \beta_2(2v_1 + v_2 - u - s_2) - 2\beta_3(3v_1 + 2v_2 - u - s_3)$$

$$\frac{\partial u}{\partial t} = -\alpha + \beta_1(v_1 + v_2 - u - s_1) + \beta_2(2v_1 + v_2 - u - s_2) + \beta_3(3v_1 + 2v_2 - u - s_3)$$

$$\frac{\partial s_1}{\partial t} = \beta_1(v_1 + v_2 - u - s_1)$$

$$\frac{\partial s_2}{\partial t} = \beta_2(2v_1 + v_2 - u - s_2)$$

$$\frac{\partial s_3}{\partial t} = \beta_3(3v_1 + 2v_2 - u - s_3)$$

$$\frac{\partial \alpha}{\partial t} = (u - 1)$$

$$\frac{\partial \beta_1}{\partial t} = \frac{1}{2}(v_1 + v_2 - u - s_1)^2$$

$$\frac{\partial \beta_2}{\partial t} = \frac{1}{2}(2v_1 + v_2 - u - s_2)^2$$

$$\frac{\partial \beta_3}{\partial t} = \frac{1}{2}(3v_1 + 2v_2 - u - s_3)^2$$

The derivatives for both DMU^B and DMU^C cases are similar with the above case 1 (DMU^A) and are not depicted here since the limited pages of the paper length regulation. The convergence trajectory of the HNN for DMU^A is shown in Figures 3, similar trajectory also happened to both DMU^B and DMU^C cases. Table 2 summarizes the optimal DEA solution obtained from the both methods and shows that the same solution was for the two methods.

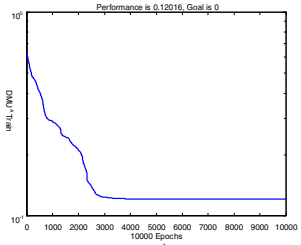

Figure 3. Trajectory line of DMU^A convergence to its stable state after 5500 iterations

TABLE 2. COMPARISON OF LAGRANGE HNN-BASED DEA TO LP-BASED DEA

Method	LP-based CCR solution				Lagrange HNN-based solution			
Weight	v_1	v_2	u	ϕ	$v_1^{(1)}$	$v_2^{(1)}$	$u_1^{(1)}$	ϕ
DMU^A	0.5	0.5	1	1	0.5	0.5	1	1
DMU^B	0	1	1	1	0	1	1	1
DMU^C	0	0.5	0.5	0.5	0	0.5	0.5	0.5

VII. CONCLUSIONS

Seldom is a discussion given on the trials and issues of HNN optimal training for DEA optimization. This work aimed to demonstrate a feasible method of utilizing HNN to find the optimal relative efficiency among multiple DMUs of a DEA problem. The approach was exemplified for substantiating the solution quality of Lagrange HNN-based DEA; while, it compared with the one of the DEA-Solver package, the two solutions were the same. The compared results presented primarily discussed on the application and the end results obtained when applying HNN to DEA. In the future, HNN will be extended to deal with a DEA model that exhibits complex uncertain characteristics and cannot be treated satisfactorily by the traditional LP technique. The ability to learn the sophisticated uncertain relationships and to tolerate the data fault makes HNN provide an ideal way to overcome the complicated stochastic DEA data.

REFERENCES

[1] Charnes, A., W.W. Cooper, and E. Rhodes, "Measuring the efficiency of decision making units," European Journal of Operations Research, vol.2, no.6, 1978, pp.429-44.

[2] Seiford, L.M. and R.M. Thrall, "Recent developments in DEA: the mathematical programming approach to frontier analysis." Journal of Econometrics, vol.46, 1990, pp.7-38.

[3] Bowlin, W. F., "Measuring Performance: An Introduction to Data Envelopment Analysis (DEA)", Journal of Cost Analysis, Fall, 1998, pp.3-27.

[4] Bouchard, G.., S. Girard and A. Iouditski, A. Nazin, "Some linear programming methods for frontier estimation," Applied Stochastic Models in Business and Industry, vol.21, no.2, 2005, pp.175-185.

[5] Zhang, S. W. and Constantinides, A. G., "Lagrange programming neural networks," IEEE Transactions on CAS, 39, 1992, pp.441–451.

[6] Zhu, X., Zhang, S. W. and Constantinides, A. G., "Lagrange neural networks for linear programming," Journal of Parallel and Distributed Computing, 14, 1992, pp.354–360.

[7] S.H. Zak, V. Upatising and S. Hui, "Solving linear programming problems with neural networks: a comparative study", IEEE Transactions on Neural Networks, vol.6, no.1, 1995, pp.94-104.

[8] Wu, Deshdng, Yang, Zijiang, and Liang, Liang (2006), "Using DEA-neural network approach to evaluate branch efficiency of a large Canadian bank," Expert System with Applications, Vol. 31, No.1, pp. 108-115.

[9] Wang, Shouhong, "Adaptive non-parametric efficiency frontier analysis: a neural-network-based model," Computers & Operations Research, vol.30, no.2, 2003, pp279-295.

[10] Pendharkar, P. C. and Rodger, J. A.., "Technical efficiency-based selection of learning cases to improve forecasting accuracy of neural networks under monotonicity assumption," Decision Support Systems, vol.6, no.1, 2003, pp.117-136.

[11] Costa, A. and Markellos, R.N., "Evaluating public transport efficiency with neural network models," Transportation Research Part C: Emerging Technologies, vol.5, no.5, 1997, pp.301-312.

[12] Athanassopoulos, A. D. and S. P. Curram, "A comparison of data envelopment analysis and artificial neural networks as tools for assessing the efficiency of decision making units," The Journal of the Operational Research Society, vol.47, no.8., 1996, pp.1000-1016.

[13] Liao, Hailin, Bin Wang and Tom Weyman-Jones, "Neural Network Based Models for Efficiency Frontier Analysis: An Application to East Asian Economies' Growth Decomposition," Global Economic Review, vol.36 - v3636, no.4, 2007, pp.361-384.

[14] Boussofiane, A., Dyson, R.G., Thanassoulis, E. "Applied data envelopment analysis," European Journal of Operational Research, vol.52, no.1, 1991, pp. 1-15.

[15] Cooper, W. W., L. M. Seiford and K. Tone, Data Envelopment Analysis: A Comprehensive Text with Models, Applications, References and DEA-Solver Software, Springer Science, 2006.

[16] Charnes, A., Cooper, W. W. and Seiford, L. M., Data envelopment analysis: theory, methodology, and application (3rd ed), Kluwer Academic Publishers, 1997.

[17] Avkiran, N. K. and T. Rowlands, "How to better identify the true managerial performance: State of the art using DEA," Omega, vol.36, no.2, 2008, pp.317-324.

[18] Dula, J. H., "A computational study of DEA with massive data sets," Computers and Operations Research, vol.35, no.4, 2008, pp.1191-1203.

[19] Zhou, P., B.W. Ang and K.L. Poh, "Measuring environmental performance under different environmental DEA technologies," Energy Economics, vol.30, no.1, 2008, pp.1-14.

[20] Tank, D. W. and J. J. Hopfield., "Simple neural optimization networks: an A/D converter, signal decision network, and a linear programming circuit," IEEE Transactions on Circuits Systems, vol.CAS-33, 1986, pp.533-541.

[21] Kennedy, M. P. and L. O. Chua, "Neural networks for nonlinear programming," IEEE Transactions on Circuits Systems, vol.CAS-35, no.5, 1988, pp.554-562.

[22] Kennedy, M. P. and L. O. Chua, "Unifying the Tank and Hopfield linear programming circuit and the canonical nonlinear programming circuit of Chua and Lin" IEEE Transactions on Circuits Systems, vol.34, no.3, 1988, pp.210-214.

[23] Maa, C-Y. and M. A. Shanblatt, "Stability of linear programming by neural networks," Proc. Int. Joint Conf. Neural Networks, vol.3, pp.941-946,1990.

[24] Wang, Z., Fang, J., Liu, X., "Global stability of stochastic high-order neural networks with discrete and distributed delays," *Chaos, Solitons and Fractals*, vol.36, no.2, 2008, pp. 388-396.

[25] Guo, B., Wang, D.H., Shen, Y. and Li, Z.S. "A Hopfield neural network approach for power optimization of real-time operating systems," Neural Computing and Applications, vol.17, no.1, 2008, pp.11-17.

[26] Masson, E. and Wang, Y.-J. "Introduction to computation and learning in artificial neural networks," vol.189, no.1, European Journal of Operational Research, 2008, pp.1-28.

[27] Ripley, Brian D., Pattern Recognition and Neural Networks, Cambridge University Press, 2008.

[28] Samarasinghe, S., Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition, Taylor and Francie Group, 2007.

[29] Cichocki, A. and R. Unbehauen, Neural Networks for Optimization and Singal Processing, John Wiley, 1993.

[30] Grippo, L., "A Class of Unconstrained Minimization Methods for Neural Network Training," Optimization Methods and Software, vol. 4, no.1, 1994, pp. 135-150.

[31] Huang, Y. C., "A Novel Method to Handle Inequality Constraints for Convex Programming Neural Network," Neural Processing Letters, 16, 2002, pp.17–27.

[32] Wang, J. and C. Vira. "Recurrent neural networks for linear programming: analysis and design principles," Computers and Operations Research, vol.19, no.3, 1992, pp.297-311.

[33] Cichocki, A. and A. Bargiela, "Neural networks for solving linear inequality systems," Parallel Computing, vol.22, no.11, 1997, pp.1455-1475.

[34] Zhang, S. and A. G. Constantinides, "Lagrange programming neural networks," IEEE Transactions on Circuits Systems - 2: Analog and Digital Signal Processing, vol.39, no.2, 1992, pp.441-454.

[35] Chapra, Steven C., "Applied Numerical Methods with MATLAB for Engineers and Scientists," MacGraw Hill, 2006.