Proceedings of the V Brazilian Conference on Neural Networks – V Congresso Brasileiro de Redes Neurais
pp. 307-312, April 2-5, 2001 - Rio de Janeiro - RJ - Brazil

Neuro-Dea: A New Optimization Algorithm

Prof. M.Sc. Luiz Biondi Neto [1], Prof. D.Sc. Marcos Estellita Lins [2], Prof. M.Sc. Luis Chiganer [3], Prof. M.Sc. Fernando Hideo Fukuda [4], Prof. M.Sc. Vincenzo De Roberto Junior [5], Prof. M.Sc. Elias Restum Antonio [6], Prof. M.Sc. Fabiano S. Oliveira [7]
[1, 3, 4, 5, 6] Departamento de Ciências Exatas e Tecnologia, Universidade Veiga de Almeida, UVA
[2] Programa de Engenharia de Produção – COPPE – UFRJ
[7] Laboratório de Estatística Computacional – PUC-rio
E-mails: [1] lbiondi@embratel.net.br, [2] estellit@iis.com.br, [3] lchiganer@uol.com.br,
[4] fukuda@uninet.com.br, [5] droberto@uninet.com.br, [6] sosscan@br.homeshopping.com.br,
[7] gomes@ele.puc-rio.br

Abstract

This paper investigates the use of Data Envelopment Analysis (DEA) combined with Artificial Neural Networks (ANN) to produce a new hybrid optimization structure called Neuro-DEA. The basic idea is build an unconventional ANN where the optimizations modules used in DEA analysis are implemented in high-speed optimization structure called Neuro-LP. The Neuro-LP structure represent an ANN who implements a numerical solution based on gradient method. So the linear programming problem (LPP) is converted into an optimization problem without constraints by using a pseudo-cost function and penalty techniques. The Neuro-LP structure represents each Decision-making unit (DMU) and may be connected assembling the final proposed Neuro-DEA structure.

1. Introduction

Consider a decision-making units (DMU) acting as an enterprise, a department or an administrative unit whose efficiency is been investigated. The Data Envelopment Analysis (DEA) is a mathematical technique that has the objective of analyzing the DMUs performance.

It allows the evaluation of the relative operational efficiency of DMUs, contemplating each DMU relatively to all the others, in the same group.

The DEA technique compares the DMU efficiencies by their abilities in transforming inputs into outputs, measuring the reached output relation in terms of the provision supplied by the input [1]. In the end of the analysis, the DEA technique is able to assert which units are (relatively) efficient and which ones are (relatively) inefficient, [2] and [6].

The DEA technique involves the use of Linear Programming (LP) to solve a group of inter-related linear programming problems (LPPs), as many as the DMU numbers, finally objectifying, determining the relative efficiency of each DMU [13]. Optimization modules called Neuro-LP will be used in the neural model proposed (Neuro-DEA), inspired by the artificial neural network philosophy [5], [18], [20], [23] and [30].

The DEA models can be oriented to inputs or to outputs and this orientation must be previously chosen by the analyst [25]. The orientation to inputs indicates that we want to reduce the inputs, keeping the outputs unaffected. In the other hand, the orientation to outputs indicates that we want to increase the outputs without affecting the inputs [31]. The most important models are the following:

CCR – Model presented by CHARNES, COOPER AND RHODE [7] that builds a non parametrical surface, piece wise linear frontier, over the data and determines the investigated DMUs technical efficiency over this surface. It was conceived as an input oriented model and it works with constant return of scale (CRS), which means that each variation in the inputs produces a proportional variation in the outputs. The proposed model to a generic DMU, eq. (1), is the following:

$$\text{Max } h_0 = \frac{\sum_{j=1}^{s} u_j Y_{j0}}{\sum_{i=1}^{r} v_i X_{i0}}$$

$$\text{subject to : } \frac{\sum_{j=1}^{s} u_j Y_{jk}}{\sum_{i=1}^{r} v_i X_{ik}} \leq 1, \quad k = 1,\dots,n$$

$$u_j \text{ e } v_i \geq 0 \quad \forall \text{ j, i} \qquad (1)$$

where :

h_0 - DMU 0 efficiency
r - total amount of inputs
s - total amount of outputs
n - total amount of DMU
Y_{jk} – amount of output j to DMU k
X_{ik} – amount of input i to DMU k
u_j – weight to output j
v_i – weight to input i

The problem consists in determining the u_j and v_i weight values to maximize the linear combination of the outputs divided by the linear combination of the inputs [14] and [15]. The process must be repeated to each of the n DMUs and, by these processes, can be determined the relative value of each DMU efficiency.

If u and v are the optimum solution vectors, αu and αv will be optimum solution vectors too, and consequently the problem will present infinite solutions.

To solve this problem, CHARNES AND COOPER [6] introduced a linear transformation that allows transforming linear fractional problems into LPPs, creating the model, eq. (2), called Multipliers Model:

$$\begin{aligned} &\text{Max } h_0 = \sum_{j=1}^{s} u_j Y_{j0} \\ &\text{subject to :} \\ &\sum_{i=1}^{r} v_i X_{ik} = 1 \\ &\sum_{j=1}^{s} u_j Y_{jk} - \sum_{i=1}^{r} v_i X_{ik} \le 0, \quad k = 1,\dots,n \\ &u_j \text{ e } v_i \ge 0 \quad \forall \text{ j,i} \end{aligned} \tag{2}$$

It's possible derive the dual model to multiplier (primal). So, the dual will present a smaller amount of constraints (s+r < n+1), because the DEA model requires that the number of DMUs be greater than the number of variables. By the reasons above, the dual model, called Envelope Model, easily solved, is preferred compared to the Multipliers model. In the Envelope model, eq.(3), the objective is to determine the values of λ_k, minimizing θ:

$$\begin{aligned} &\text{Min } \text{è} \\ &\text{subject to :} \\ &-Y_{j0} + \sum_{k=1}^{n} Y_{jk} l_k \ge 0 \text{ j} = 1,\dots\text{s} \\ &\text{è}X_{j0} - \sum_{k=1}^{n} X_{ik} l_k \ge 0, \quad i = 1,\dots,r \\ &l_k \ge 0 \quad \forall \text{ k} \end{aligned} \tag{3}$$

To output oriented formulation, we just have to invert the quotient, calculating the relation between the weighted sum of the inputs divided by the weighted sum of the outputs, trying to minimize the inputs. In this case, the Envelope model, derived from the primal (multipliers), can be similarly obtained similarly to the case of the orientation to inputs.

BCC – Model developed by BANKER, CHARNES AND COOPER [3], allows variable return of scale (VRS) avoiding existing problems in imperfect competition situations, financial constraints etc. In this case, the VRS frontier considers increasing or decreasing returns in the efficient frontier. To do this job, it was introduced in the CRS model, a convexity constraint making the λ sum equal to 1. Trying to compare the CRS frontier to the VRS, the Figure 1, that represents 5 DMUs of one input and one output, shows the relation between these frontiers for the both cases.

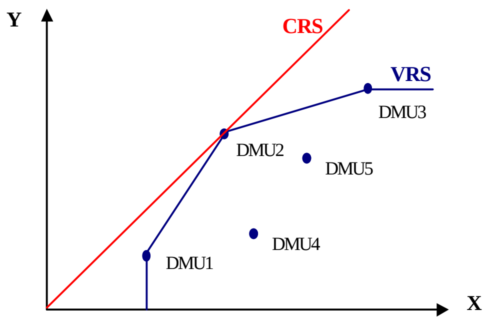

Figure 1 - CRS and VRS Frontier

The Envelope model, eq. (4), oriented to input, is represented by:

$$\begin{aligned} &\text{Min } \text{è} \\ &\text{subject to :} \\ &-Y_{j0} + \sum_{k=1}^{n} Y_{jk} l_k \ge 0 \text{ j} = 1,\dots\text{s} \\ &\text{è}X_{j0} - \sum_{k=1}^{n} X_{ik} l_k \ge 0, \quad i = 1,\dots,r \\ &\sum_{k=1}^{n} l_k = 1 \\ &l_k \ge 0 \quad \forall \text{ k} \end{aligned} \tag{4}$$

In Figure 1, it's possible to verify that the DMUs 4 and 5 are inefficient. In the BCC model that adopts VRS, each DMU is compared to the efficient DMUs that operate in the same scale. So, using the orientation to inputs, we verify that the optimum projection of the DMU 4 occurs in a point that is convex linear combination of DMUs 1 and 2.

Using the orientation to inputs, we verify that the optimum projection of the same DMU 4 happens in a point that reflects the convex linear combination of DMUs 2 and 3. However, for both orientation cases, the linear combination values are given by the λs.

The primal derived model (multipliers), eq. (5), is given by:

$$\begin{aligned} &\text{Max } h_0 = \sum_{j=1}^{s} u_j Y_{j0} + t_0 \\ &\text{subject to:} \\ &\sum_{i=1}^{r} v_i X_{ik} = 1 \\ &\sum_{j=1}^{s} u_j Y_{jk} - \sum_{i=1}^{r} v_i X_{ik} + t_0 \le 0, \quad k = 1,\dots,n \\ &u_j \text{ and } v_i \ge 0 \quad \forall \text{ j,i and } t_0 \text{ unrestricted} \end{aligned} \tag{5}$$

2. Mathematical Basics

In the traditional ANN [26], [27], [28] and [29], after the training phase, where the values of the synaptic weights are determined, the ANN will be ready to be executed. In this phase the ANN receives signals in the input, which, usually, did not take part in the training phase, and presents the result in the output, according to the knowledge acquired during the training phase and stored in the weight matrix [11], [32] and [33].

In the Neuro-LP case, the weights of unconventional ANN, are already known and represents the problem constraint coefficients [8], [9], [10] and [22]. The next step (execution phase) determines the output, which indicates the value of the LPP decision variables [5].

Considering an optimization problem without constraints where we wish to find the value $x \in \Re^n$ that minimizes a scalar function E(x), called Pseudo-cost, Energy or Objective function. According [9] and [10], the point x^* will be the global minimum of E(x), if $E(x^*) <= E(x)$ for all $x \in \Re^n$ and a local minimum if $E(x^*) <= E(x)$ is kept for a interval $\varepsilon > 0$.

If the first and the second E(x) derivatives exist, the point x^* will be a local minimum if the gradient $\nabla E(x^*)=0$ and the Hessian matrix $\nabla^2 E(x^*) > 0$. The necessary and sufficient conditions for the existence of a local minimum are:

For $\nabla^2 E(x)$ non singular for the point x*, so E(x*) will be <= E(x) for all $0 < \| x - x^* \| < \varepsilon$ e $\varepsilon > 0$ if the gradient $\nabla E(x^*) = 0$ and the Hessian matrix is symmetric and positive, $\nabla^2 E(x^*) > 0$.

The Dynamic Gradient Method is the mostly known method from the ones inspired in the "Steepest Descent" and the "Newton Method" [4]. It's based in the transformation of the optimization problem without constraints in a first order ordinary differential equation system, eq. (6), represented as follows:

$$\frac{dx_j}{dt} = -m_{ji} \sum_{i=1}^{n} \frac{\partial E(x)}{\partial x_j},$$
$$\text{where } x_j(0) = x_j^{(0)} \text{ are a initial conditions}$$
$$\text{and } m_{ji} \text{ is a learning matrix.} \qquad (6)$$

So, to find the value x* that takes E (x) to a minimum, its necessary to solve or simulate the solution of a differential equation system subjected to initial conditions. Its possible to conclude that x* can be determined by the, eq. (7), "solution path or trajectory curve" of the proposed system:

$$x^* = \lim_{t \to \infty} x(t) \qquad (7)$$

According [16] and [17], given a cost function E(x) which must be minimized, where $xj(0) = xj^{(0)}$ represents the point where the procedure starts and; being $\nabla E(x^*)$ the gradient of E(x) for the kth point in the path, then; the idea is to determine a particular path p where dE(x)/dp is minimized in each point of the path. The numerical solution can be obtained considering the following:

- If successive points x^k e x^{k+1} are determined obeying the solution path: $x^{k+1} = x^k - \eta^k \nabla E(x)^k$, where $0 <= \eta^k <= \eta^{max}$ is called integration step.
- The value of η^k is determined in a way that x^{k+1} always results in the improvement of the objective function, $E(x)^{k+1} < E(x)^k$.

The procedure ends when two consecutive points x^k and x^{k+1} are approximately the same, $\eta^k \nabla E(x)^k \cong 0$. As $\eta^k \neq 0$, so $\nabla E(x)^k = 0$.

3. Neuro-LP Modeling

To solve it using the ANN philosophy, its necessary to build a "new function" called pseudo-cost or energy function E (x) [35], which global minimum is the optimum solution of the LPP.

To build the new function E(x), it's incorporated a function or penalty term $P_i [R_i(x)]$ to the original objective function [8], [22] and [36]. The penalty term must cause a high cost (penalty) to the new function each time a constraint is violated ($R_i(x) < 0$) and zero cost if the constraint is satisfied ($R_i(x) > 0$).

In this way, the LPP is transformed into an optimization problem without constraint, where its desirable to find $x^* \in \Re^n$ that minimizes the new function E(x). The penalty term must penalize (big *p*) for the cases of no feasible solutions and inhibit for viable solutions of the LPP [12], [17] and [34]. The optimization problem without constraint with penalty term can be solved similarly to the ANN training phase, applying the gradient method, eq. (6) and eq. (7).

Its modeling without constraint, eq. (8), is the following:

$$\text{Min } E(x,p) = \sum_{j=1}^{n} C_j x_j + p \sum_{i=1}^{m} P_i[\mathrm{R_i}(x)]$$
$$\text{where } p >> 0, \text{represent a penalty parameter} \qquad (8)$$

$$\text{If } \mathrm{R_i}(x) \geq 0 \qquad P_i[\mathrm{R_i}(x)] = 0 \quad \text{(inhibit)}$$
$$\text{If } \mathrm{R_i}(x) < 0 \qquad P_i[\mathrm{R_i}(x)] > 0 \quad \text{(punishiment)}$$

The practice shows that *p* values extremely high are not convenient from the computing point of view.

According [5], [9] and [10] a great choice is to consider the pseudo-cost function, eq. (9), the following:

$$E(x,p) = \sum_{j=1}^{n} C_j x_j - p \sum_{i=1}^{m} \min\{0, \mathrm{R_i}(x)\}, \qquad (9)$$
$$\text{where } p > 0$$

In this case, higher values of *p* are not necessary for the correct convergence of the process. So, with reasonable *p* values, the minimum of the pseudo-cost function E(x, *p*) is equivalent to the final optimum solution of the original LPP, showed in eq. (10).

As $\Delta(x_j) = -m_j \nabla_{x_j} E(x_j, p)$ e

$R_i(x) \cong a_{ij} x_j$

$$\Delta(x_j) = -m_j \left[C_j + \sum_{i=1}^{m} D_i a_{ij} \right] \quad (j=1,2,...,n)$$

If $R_i(x) \geq 0$ $\quad D_i = 0$ $\quad$ (inhibit)

If $R_i(x) < 0$ $\quad D_i = -p$ $\quad$ (punishiment)

$p > 0$, $m_j > 0$ and D_i is a negative step

The solution curves are: $\qquad$ (10)

$$x_j^{(k+1)} = x_j^{(k)} + \Delta(x_j)$$

$$x_j^{(k+1)} = x_j^{(k)} - m_j \left[C_j + \sum_{i=1}^{m} D_i^{(k)} a_{ij} \right]$$

If $\sum_{j=1}^{n} a_{ij} x_j \geq b_i$ $\quad D_i = 0$

If $\sum_{j=1}^{n} a_{ij} x_j < b_i$ $\quad D_i = -p$

$p > 0, \quad m_j > 0, \quad i = 1,2,...,m, \; j = 1,2,...,n, \quad k = 1,2,....$

4. Neuro-DEA Architecture

Considering the data existence for **P** DMUs, with **R** inputs and **S** outputs and that the ith DMU is represented by a column vector X_i (inputs) and a Y_i (outputs).

The relationship between all inputs and outputs is obtained to each DMU: **u** Y_i / **v** X_i , where **u** and **v** are output weight vectors and input weight vectors, respectively. The optimum values for these weights are obtained solving a LPP for each DMU, [21] and [25].

So, for **P** DMUs, we will have **P** Neuro-LP modules. Each LPP of the Neuro-DEA model will represent a LPP in the Neuro-LP model and will be able to determine the relative efficiency of one DMU of **P** DMUs that compose the system. Figure 4.1 shows the proposed block diagram of the Neuro-DEA module.

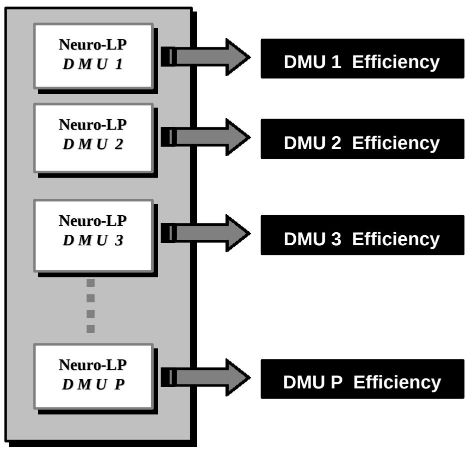

Figura 4.1 - Neuro-DEA Block Diagram

The implementation was done using the CRS Envelope model, input oriented. The reason for this choice is due to the fact that we can reduce the number of constraints, because in the envelope model we have a constraint for each input/output. This fact does not happen in the multipliers model, where we have as many constraints as the numbers of DMUs, complicating the implementation, increasing computational work. [14].

Figure 4.2 shows a generic architecture, used in cases until 5 variables, where it's possible identify step by step, the equations, eq. (10), that represent a Neuro-LP model used to determination of efficiency of each DMU. This simulation was done using MatLab Simulink Tool [19] and [24].

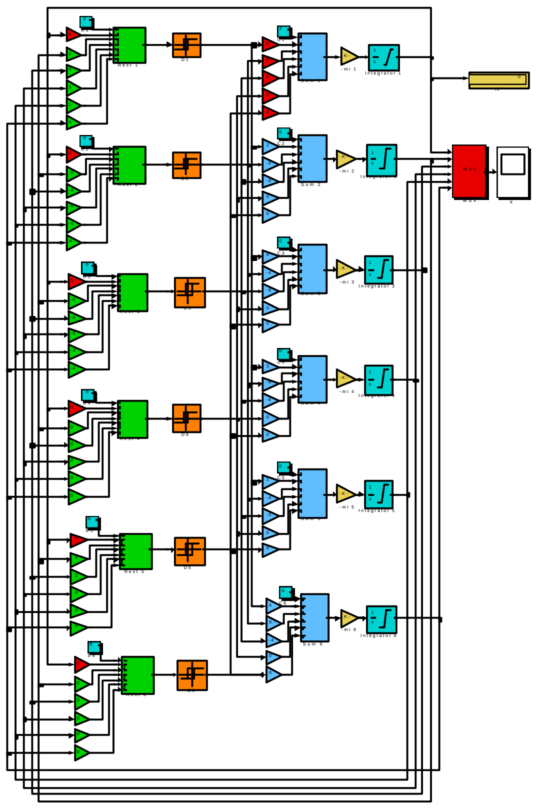

Figure 4.2 - Neuro-Dea Architecture until 5 variables

5. Neuro-DEA Model Implementation

Some data for a case involving 5 DMUs with two inputs and one output are shown in Table 1. Figure 5.1 represents 3D data and envelopment representation.

Figure 5.2 shows the final specification for the LPPs referring to each DMU. Figure 5.3 shows a CRS diagram that is referring to the problem, where its possible to verify that DMUs 1, 3 and 4 are not efficient and so, can have all of their inputs reduced of a certain percentage, without reducing the output. DMUs 2 and 5 are efficient.

Finally, in Table 2 there is a comparison for the obtained results, using two commercial and consecrated softwares (LINDO AND FRONTIER), with the results of the model proposed in this paper and calculating the error percentage.

Table 1 - DMUs Data

	Out	Input 1	Input 2	Input 1/Out	Input 2/Out
DMU 1	2	6	8	3	4
DMU 2	4	4	8	1	2
DMU 3	3	6	6	2	2
DMU 4	1	4	3	4	3
DMU 5	4	12	4	3	1

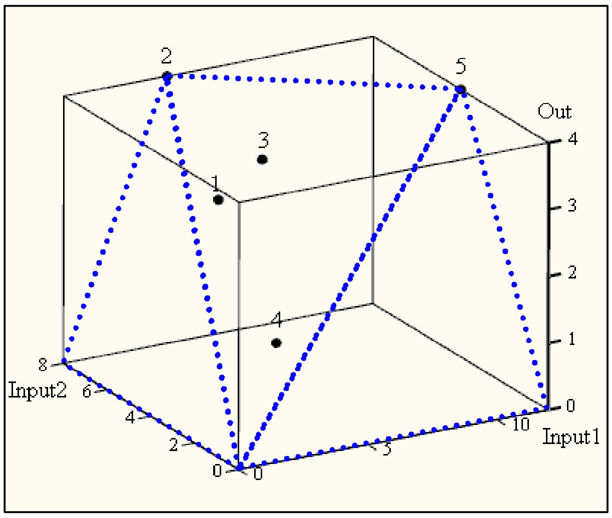

Figure 5.1 - 3D Data and Envelopment Representation

```
                         EFFIC 1 = Min t1
Min t1
Subject to:
        2l1+4l2+3l3+l4+4l5>=2
   6t1-6l1-4l2-6l3-4l4-12l5 >=0
    8t1-8l1-8l2-6l3-3l4-4l5 >=0
   I1, I2, I3, I4, I5 >=0

                         EFFIC 2 = Min t2
Min t2
Subject to:
        2l1+4l2+3l3+l4+4l5>=4
   4t2-6l1-4l2-6l3-4l4-12l5 >=0
    8t2-8l1-8l2-6l3-3l4-4l5 >=0
   I1, I2, I3, I4, I5 >=0

                         EFFIC 3 = Min t3
Min t3
Subject to:
         l1+4l2+3l3+l4+4l5>=3
   6t3-6l1-4l2-6l3-4l4-12l5 >=0
    6t3-8l1-8l2-6l3-3l4-4l5 >=0
   I1, I2, I3, I4, I5 >=0

                         EFFIC 4 = Min t4
Min t4
Subject to:
        2l1+4l2+3l3+l4+4l5>=1
   4t4-6l1-4l2-6l3-4l4-12l5 >=0
    3t4-8l1-8l2-6l3-3l4-4l5 >=0
   I1, I2, I3, I4, I5 >=0

                         EFFIC 5 = Min t5
Min t5
Subject to:
        2l1+4l2+3l3+l4+4l5>=4
  12t5-6l1-4l2-6l3-4l4-12l5 >=0
    4t5-8l1-8l2-6l3-3l4-4l5 >=0
    I1, I2, I3, I4, I5 >=0
```

Figure 5.2 - LPPs Specification to each DMU

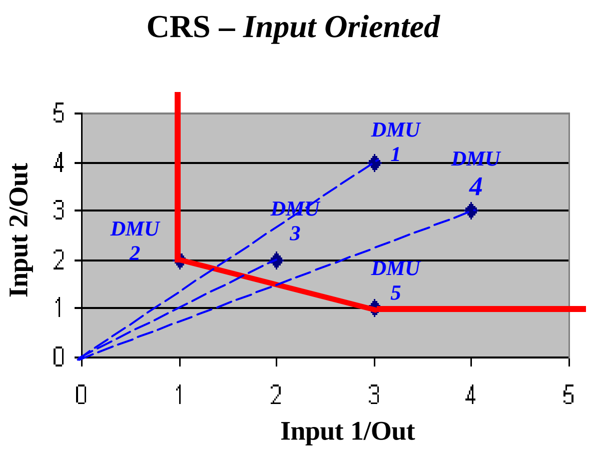

Figure 5.3 - CRS Diagram

Table 2 - Results Comparison

	Lindo	Frontier	Neuro DEA	% Error
DMU 1	0.453	0.454	0.452	0.44
DMU 2	1.000	1.000	1.000	0.00
DMU 3	0.832	0.833	0.831	0.24
DMU 4	0.500	0.500	0.500	0.00
DMU 5	1.000	1.000	1.000	0.00

6. Conclusions

This paper presented a new optimization algorithm base in a structure called Neuro-LP. This structure is an unconventional Neural Network model dedicated to execute high-speed optimization efficiency calculation.

This feature is very useful in real time applications and it was our main motivation. With these basic modules we are capable of building a Neuro-DEA architecture, a hybrid DEA-ANN optimization structure and algorithm.

In the Neuro-LP, the solution method for the ordinary differential equation system is similar to the technique used in ANN training, because both of them use the decreasing gradient method. The convergence of the algorithm showed is very fast in all data sets investigated.

The presented case was selected among hundreds of accomplished tests showing that the presented proposal is consistent. The values obtained with our prototype were validated comparing themselves with results obtained by renowned softwares as LINDO, to separately solve the LPPs referring to the DMUs and the FRONTIER to directly solve the DEA. In this case, the observed error did not surpass 0.5%.

The integration of ANN with other numerical methods to solve LPPs or DEA is a very promising research area to develop high-speed convergence solution.

Finally, it’s important to highlight that the proposed modules can be integrated in a chip and connected to a free slot in a computer.

References

[1] Abraham Charnes, Willian W. Cooper, Arie Y. Lewin And Lawrence M. Seiford, *Data Envelopment Analysis: Theory, Methodology, and Application*, Kluwer Academic Publishers, Boston, 1996.

[2] Angulo Meza L., *Data Envelopment Analysis (DEA) na Determinação da Eficiência dos Programas de Pós-Graduação do COPPE/UFRJ*, Dissertação de Mestrado, UFRJ – COPPE, Rio de Janeiro, RJ, Brasil, 1998.

[3] Banker R., Charnes A., Cooper W. W, Some Models for Estimating Technical an Scale Inefficiencies in Data Envelopment Analysis, *Management Science*, v.30, pp. 1078-1092., 1984.

[4] Bazaraa M. S., Sherali H. D., Shetty C. M, *Nonlinear Programming Theory and Algorithms*, John Wiley & Sons, Inc., New York, USA, 1993.

[5] Biondi L. N., Estellita Lins Et Al, Neuro-DEA: Novo Paradigma para Determinação da Eficiência Relativa de Unidades Tomadoras de Decisão, *9º Congresso da Associação Portuguesa de Investigação Operacional – APDIO*, n º 9, pp. 114, 2000.

[6] Charnes A., Cooper W. W., Programming with linear fractional functional, *Naval Research Logistics Quarterly*, v.9, n 3/4 , pp. 181-186, 1962.

[7] Charnes A., Cooper W. W., Rhodes E., Measuring the Efficiency of Decision-Making Units, *European Journal of Operational Research*, v.2, pp. 429-444, 1978.

[8] Chen J., Shanblatt M. And Maa C., Improved Neural Network for Linear and Nonlinear Programming, *International Journal of Neural Systems*, no. 2, pp. 331-339, 1992.

[9] Cichocki A. And Bargiela A., "Neural Networks for Solving Linear Inequality Systems", *Journal of Parallel Computing*, http:// www.bip.riken.go.jp/absl, 1996.

[10] Cichocki A. And Unbehauen R., *Neural Networks for Optimization and Signal Processing*, John Wiley & Sons, Inc, New York, USA, 1996.

[11] David, M. Skapura, *Building Neural Networks*, Addison-Wesley Publishing Company, New York, 1996.

[12] Dennis J. E. And Schnabel R.B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, USA, Prentice Hall, Inc, Englewood Cliffs, N.J., 1996.

[13] Estellita L. M., Lídia Angulo Et Al, *Análise de Envoltória de Dados e Perspectivas de Ambiente de Apoio à Decisão*, COPPE/UFRJ, Rio de Janeiro, Brasil,2000.

[14] Estellita L. M., *Análise de Fronteiras de Produtividade "Data Envelopment Analysis"* COPPE, notas de Aula, Rio de Janeiro, Brasil, 1997.

[15] Estellita L. M. E. Moreira, Modelo I O – Stepwise para seleção de variáveis em Modelos de Análise de Envoltória de Dados, *Revista Pesquisa Operacional*,v.19, no1, pp.39-49, 1998.

[16] Hamdy, A. Taha, *Operations Research: An Introduction*, Prentice Hall, New Jersey, 1992.

[17] Heitor Pina, *Métodos Numéricos*, McGraw-Hill, Lisboa, Portugal, 1995.

[18] Jacek M. Zurada, *Introduction to Artificial Neural Systems*, West Publishing Company, New York, 1992.

[19] Jerzy Moscinski, Zbigniew Ogonowski, *Advanced Control with MatLab Simulink*, Ellis Horwood Limited, London, 1995.

[20] Judith E. Dayhoff, *Neural Network Architectures an Introduction*, Van Nostrand Reinhold, London, 1990.

[21] Kallrath J. And Wilson J., *Business Optimization using Mathematical Programming*, Macmillian Press Ltd, London, 1997.

[22] Kennedy M. P. S Chua L., Neural Networks for Nonlinear Programming, *IEE Transactions on Circuits and Systems*, no. 35, pp. 554-562, 1994.

[23] M. Minsk And S. Papert, *Perceptrons*, MIT Press, Cambridge, 1969.

[24] Math Works, *Simulink Dynamic System Simulation Software, User's Guide*, Math Works, Inc., Massachusetts USA, 1995.

[25] Michael Norman And Barry Stoker, *Data Envelopment Analysis – The assessment of Performance*, John Wiley & Sons, New York, 1991.

[26] Philip D. Wasserman, *Advanced Methods in Neural Computing*, Van Nostrand Reinhold, London, 1993.

[27] Robert L. Harvey, *Neural Network Principles*, Prentice Hall International Editions, New York, 1994.

[28] Rosenblatt F., *Principles of Neurodynamics*, Spartan Editions, New York, 1962.

[29] Rumelhart, D. E., Hinton, G. E. And Willian, R. J., *Learning Internal Representation by Error Propagation. In Parallel Distributed Processing*, Vol1, M. A., D. E. Rumelhart and J. L. McClelland Eds., MIT Press, Cambridge, 1986.

[30] Simon Haykin, *Neural Networks a Comprehensive Foundation*, Macmillan College Publishing Co., London, 1994.

[31] Tim Coelli, D. S. Prasada Rao And George E. Battese, *An Introduction to Efficiency and Productivity Analysis*, Kluwer Academic Publishers, Boston, 1998.

[32] Vellasco M. E Pacheco M., *"Redes Neurais Artificiais"*, PUC, Notas de Aula, Rio de Janeiro, Brasil, 1994.

[33] Werbos, P., *Beyond Regression: New tools for Prediction and Analysis in the Behavioral Sciences*, PhD. thesis, Harvard University, USA, 1974.

[34] Werner C. Rheinboldt, *Methods for Solving Systems of Nonlinear Equations*, SIAM Editors, Philadelphia, 1998.

[35] Winstson, Wayne L., *Operations Research: Applications and Algorithms*, 3th edition, Duxbury Press, Belmont, California, 1994.

[36] Zhu X., Zhang S. And Constantinides A. G., Lagrange neural Networks to linear programming, *Journal of Parallel Distributed Computing*, no. 14, pp. 354-360, 1992.