MODELAGEM DEA-VRS COM ALTA VELOCIDADE DE CONVERGÊNCIA: UMA ABORDAGEM BASEADA NO PROBLEMA INVERSO DE REDES NEURAIS ARTIFICIAIS

Luiz Biondi Neto
UERJ - Universidade do Estado do Rio de Janeiro
Rua São Francisco Xavier 524/5019B, CEP 20550-013, Rio de Janeiro, RJ, Brasil
lbiondi@uerj.br, luizbiondi@terra.com.br

João Carlos Correia Baptista Soares de Mello
UFF – Universidade Federal Fluminense
Rua Passo da Pátria, 156, São Domingos, 24240-240, Niterói, RJ, Brasil
jcsmello@producao.uff.br

Eliane Gonçalves Gomes
Embrapa SGE
Parque Estação Biológica, W3 Norte Final, 70770-901 Brasília, DF, Brasil
eliane.gomes@embrapa.br

Lidia Angulo Meza
UFF – Universidade Federal Fluminense
Av. dos Trabalhadores 420, Vila Santa Cecília, 27255-125, Volta Redonda, RJ, Brasil
lidia_a_meza@yahoo.com

Pedro Henrique Gouvêa Coelho
UERJ - Universidade do Estado do Rio de Janeiro
Rua São Francisco Xavier 524/5025E, CEP 20550-013, Rio de Janeiro, RJ, Brasil
phcoelho@uerj.br

Resumo

Este artigo investiga a criação de estruturas de medida de eficiência relativa de unidades tomadoras de Decisão (DMUs), usando-se módulos de Programação Linear (PL) de alta velocidade de convergência, inspirados na solução do *problema inverso* de redes neurais artificiais (RNAs). A técnica proposta atende as exigências da modelagem DEA-VRS.

Os Problemas de Programação Linear (PPL) usados em Análise Envoltória de Dados (DEA) são transformados em problemas de otimização sem restrição criando-se uma função de pseudo custo na qual é adicionado um termo de penalidade, causando um alto custo toda vez que a restrição for violada. O problema é então convertido em um sistema de equações diferenciais e resolvido numericamente pelo método do gradiente decrescente, também usado no treinamento de RNAs.

Finalmente é apresentado um exemplo numérico com o objetivo de verificar, não só a exatidão do modelo proposto mas também a velocidade de convergência da resposta.

Palavras-Chaves: Redes Neurais Artificiais; Análise Envoltória de Dados; Métodos numéricos.

Abstract

This paper investigates the creation of efficiency measurements structures of

decision-making units (DMUs), by using high-speed optimization Linear Programming (LP) modules, inspired in the inverse problem solution of artificial neural network (ANN) and numerical methods. The high-speed proposal techniques fully execute the requests of DEA-VRS model.

In addition, the linear programming problem (LPP) used in Data Envelopment Models (DEA) is transformed into an optimization problem without constraints by using a pseudo-cost function, where's added a term of penalty, causing high cost all time that one of the constraints goes violated. The problem is converted into a differential equations system and solved by numerical solution for gradient method too used in learning of ANN.

Finally is showed a numerical example whose objective is to verify the accurate and convergence speed of results.

Keywords: Artificial Neural Networks; Data Envelopment Analysis; Numerical Methods.

1. INTRODUÇÃO

A análise DEA é uma técnica matemática com o objetivo de verificar o desempenho de Unidades Tomadoras de Decisão (DMUs). Essa técnica permite avaliar a eficiência operacional relativa de organizações (DMUs), contemplando cada DMU relativamente a todas as outras que compõem o grupo de DMUs investigadas. A técnica DEA envolve o uso maciço de Programação Linear (PL) [1,2] para resolver um conjunto de problemas de programação linear (PPLs) inter-relacionados, tantos quantos forem as quantidades de DMUs, objetivando finalmente determinar a eficiência relativa de cada DMU. Os modelos DEA podem ser orientados para *entrada* ou para *saída* e essa orientação deve ser escolhida, previamente, pelo analista como ponto de partida na análise DEA. A orientação para entrada indica que desejamos reduzir (minimizar) as entradas, mantendo as saídas inalteradas. Por outro lado, a orientação para saída significa que desejamos aumentar (maximizar) as saídas sem alterar as entradas [5]. Depois de escolhida a orientação define-se o modelo a ser adotado, cujos principais são os seguintes: Modelo CCR (**C**harnes, **C**ooper e **R**hodes); Modelo BCC (**B**anker, **C**harnes e **C**ooper). Dependendo do modelo, a fronteira de eficiência pode ser determinada por faces lineares, faces log-lineares ou Cobb-Douglas. Os modelos podem apresentar rendimentos de escala constante CRS ou variável (crescente ou decrescente) VRS, como o adotado nesse trabalho [5, 6].

Há situações na qual a análise DEA, em face da complexidade do problema, envolve intenso trabalho computacional, sendo necessários sistemas mais rápidos na tomada de decisão, como o proposto neste artigo [7, 17].

Por outro lado, o sistema nervoso humano é constituído, principalmente, de cerca de cem bilhões de neurônios que são células especializadas em receber impulsos elétricos de outras células semelhantes, processa-los gerando novas informações que serão transmitidas a outras células nervosas [3, 4].

Redes Neurais Artificiais (RNAs), também conhecidas como sistemas conexionistas, são estruturas inspiradas no funcionamento do cérebro humano. As RNAs são estruturas maciçamente paralelas, baseadas em neurônios artificiais simples, inspiradas no neurônio biológico e densamente interconectadas. Uma RNA é uma estrutura de processamento distribuído da informação, que consiste de neurônios artificiais interconectados por canais de comunicação unidirecionais denominados de conexões, por onde trafegam sinais.

Cada neurônio artificial emite um sinal que pode ser distribuído pelas conexões a tantos outros processadores quanto se deseje. Pode ser arbitrária a codificação da informação nos sinais que trafegam, bem como a natureza do processamento realizado pelos neurônios artificiais, conquanto seja preservado o local deste processamento.

Devido à topologia extremamente complexa do sistema nervoso, os processos biológicos que resultam em habilidades computacionais não estão, totalmente, ao alcance do

entendimento humano. O que na prática ocorre é a configuração de RNAs através do uso de modelos de elementos processadores ou neurônios artificiais às vezes incompletos.

A idéia principal do modelo de programação linear neural baseou-se na possibilidade de utilizarmos uma estrutura semelhante à de uma RNA que pudesse ser usada de *forma inversa*, isto é, como resultado da aplicação de um método iterativo a esta estrutura, se obtivesse os valores referentes às variáveis de decisão do PPL [7, 18, 19, 20].

A Figura 1 apresenta, ainda de forma pouco formal, essa hipótese na qual pode ser vista uma realimentação da saída para entrada. O primeiro bloco verifica se as restrições do problema foram atendidas ou violadas e um segundo atende as características das restrições e da função objetivo.

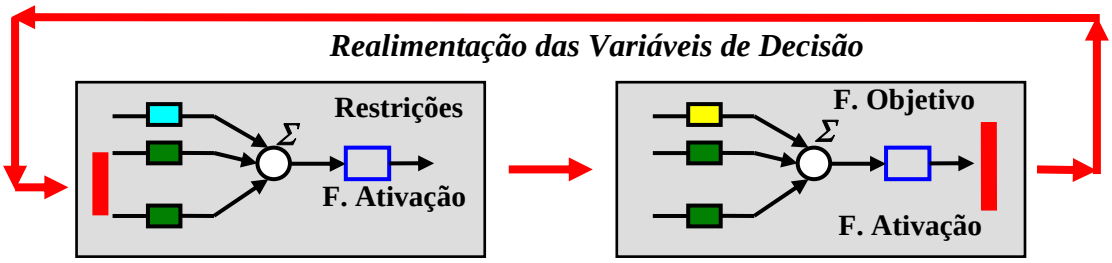

Figura 1 – Idéia do modelo de programação linear neural

Se a saída do último bloco fornecer o valor das variáveis de decisão do problema a cada iteração, a realimentação (saída/entrada) pode garantir, depois de certo número de iterações a convergência do processo para o valor ótimo, visto que, a entrada seria sempre realimentada com os novos valores dessas variáveis atualizadas, obtidas na saída. Dessa forma a solução seria melhorada a cada iteração.

Uma RNA apresenta em sua estrutura tradicional um conjunto de pesos. A soma ponderada desses pesos com os padrões de entrada feita no bloco (Σ) é aplicada na função de ativação, que pode ser um degrau, uma tangente hiperbólica ou qualquer outra que atenda melhor às necessidades da aplicação, como será feito no nosso caso, na *concepção inversa*.

No caso da programação linear neural esses pesos são representados pelos coeficientes das restrições, termos independentes e coeficientes da função objetivo, que são características conhecidas do PPL (dados do problema). Assim, esses parâmetros fixos não necessitam de nenhum treinamento, como ocorre normalmente nas RNAs.

Um método iterativo, "*semelhante*" ao treinamento das RNAs é usado, de f*orma inversa*, para atualizar nesse caso, os valores das variáveis de decisão, que são realimentadas da saída para entrada a cada iteração.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. ANÁLISE ENVOLTÓRIA DE DADOS

A análise DEA é uma técnica não paramétrica caracterizada pela busca de uma ou mais DMUs, dentre um grupo de "n" DMUs investigadas, que operem sobre a fronteira de eficiência. Cada DMU é representada por um conjunto de "s" saídas (s x n) e "m" entradas (m x n), conforme as matrizes (1 e 2) adequadamente escolhidas [5].

Os diversos modelos usados em análise DEA determinam uma fronteira ou um "*envelope de dados*", que representa o subconjunto das DMUs eficientes.

Por outro lado, se a DMU não pertencer à fronteira eficiente (for ineficiente) é atribuído um valor de eficiência relativa, tomando outras DMUs eficientes como referência.

A análise DEA permite através de mudanças nas diversas dimensões das entradas e saídas, projetar as DMUs ineficientes na fronteira eficiente [5].

$$y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \cdots & & & \\ y_{s1} & y_{s2} & \cdots & y_{sn} \end{bmatrix} \quad (1)$$

$$x = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots & & & \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \quad (2)$$

O Modelo CCR foi apresentado originalmente por CHARNES, COOPER e RHODES em 1978 e constrói uma superfície não paramétrica, linear por partes, sobre os dados. Foi concebido inicialmente como um modelo orientado a entrada (input) e trabalha com retorno constante de escala (CRS), isto é, qualquer variação nas entradas (inputs) produz variação proporcional nas saídas (outputs). Nesse modelo a eficiência é definida pela relação entre a soma ponderada das saídas (virtuais) e a soma ponderada das entradas (virtuais), visto que os pesos ou multiplicadores não são conhecidos.

Assim temos:

$$\text{EF} = \frac{\text{Soma ponderada das saídas virtuais}}{\text{Soma ponderada das entradas virtuais}} = \frac{u_1 Y_1 + u_2 Y_2 + \ldots + u_s Y_s}{v_1 X_1 + v_2 X_2 + \ldots + v_m X_m} = \frac{\sum_{j=1}^{s} u_j Y_j}{\sum_{i=1}^{m} v_i X_i} \quad (3)$$

onde : v_i e u_j representam os pesos das entradas (i) e das saídas (j)

A idéia foi admitir que cada DMU tenha autonomia de definir seu próprio conjunto de pesos. O conjunto dos pesos ótimos que maximiza a relação (3) é determinado pela solução do problema de programação linear fracionária (4) e varia de uma DMU para outra [5]. ???

$$\text{Max} \quad \text{E}_0 = \left(\frac{\sum_{j=1}^{s} u_j Y_{j0}}{\sum_{i=1}^{r} v_i X_{i0}} \right)$$

sujeito a :

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

u_j e $v_i \geq 0 \quad \forall$ j, i

onde :

E_0 - eficiência da DMU $_0$

r - quantidade total de entradas

s - quantidade total de saídas

n - quantidade total de DMUs

Y_{jk} – saída j para DMU k

X_{ik} – entrada i para DMU k

u_j – peso referente a saída j

v_i – peso referente a entrada i

(4)

É possível derivar o dual do modelo dos multiplicadores (primal). Assim, o dual apresentará uma menor quantidade de restrições (s+r < n+1), pois o modelo DEA exige que o número de DMUs seja maior que o número de variáveis. Pelas razões expostas e por ter solução computacional mais simples, o modelo dual, denominado *Envelope*, tem preferência sobre o dos Multiplicadores [5, 21, 22]. Tratando-se o modelo primal CCR, equações (5) com notação vetorial e matricial temos:

$$
\begin{aligned}
&Max \quad \mathbf{u}\mathbf{y}_0 \\
&Sujeito\ a: \\
&\mathbf{v}\mathbf{x}_0 = 1 \\
&-\mathbf{v}\mathbf{X} + \mathbf{u}\mathbf{Y} \le 0 \\
&\\
&\mathbf{v} \ge 0 \ e \ \mathbf{u} \ge 0
\end{aligned}
\tag{5}
$$

As variáveis duais são uma variável real θ e um vetor $\lambda = (\lambda_1, \lambda_2, \lambda_3, ..., \lambda_n)^T$. As equações (6) apresentam o problema dual (envelope) derivado dos multiplicadores.

$$
\begin{aligned}
&Min \quad \theta \\
&Sujeito \quad a: \\
&\theta \mathbf{x}_0 - \mathbf{X}\boldsymbol{\lambda} \ge 0 \\
&\mathbf{Y}\boldsymbol{\lambda} \ge y_0 \\
&\\
&\boldsymbol{\lambda} \ge 0
\end{aligned}
\tag{6}
$$

Escrevendo as equações (6) de outra forma chegamos às equações (7).

$$
\begin{aligned}
&\text{Min} \quad \theta \\
&\text{sujeito a:} \\
&\theta \mathbf{x}_{i0} - \sum_{k=1}^{n} \mathbf{X}_{ik} \boldsymbol{\lambda}_{\mathbf{k}} \ge 0, \quad i = 1,...,r \\
&-\mathbf{y}_{j0} + \sum_{k=1}^{n} \mathbf{Y}_{jk} \boldsymbol{\lambda}_k \ge 0 \quad \text{j} = 1,...\text{s} \\
&\\
&\boldsymbol{\lambda}_k \ge 0 \quad \forall \text{ k}
\end{aligned}
\tag{7}
$$

O modelo CCR apresentado anteriormente opera com retorno constante de escala (CRS) isto é, se existe uma DMU (x,y) com atividade viável de produção, então existirá outra DMU (λx, λy), cuja operação também é viável.

O modelo BCC, desenvolvido por BANKER CHARNES e COOPER em 1984, apresenta a fronteira de produção com forma côncava por partes, caracterizada pelo retorno variável de escala (VRS). O retorno variável de escala (VRS), foco desse trabalho, evita problemas existentes em situações de competição imperfeita, restrições nas finanças, etc [5].

A Figura 2 mostra na fronteira do modelo BCC, a região de retorno crescente de escala (**IRS**), definida pelo segmento que une os pontos onde operam as DMUs 1 e 2. A região correspondente ao segmento que une os pontos onde operam as DMU 2 e 3, definem a região de retorno de escala decrescente (**DRS**).

Finalmente, apenas no ponto onde opera a DMU 2 e por onde passa a reta radial do modelo CCR é que temos, na fronteira do modelo BCC, retorno constante de escala (**CRS**) [5].

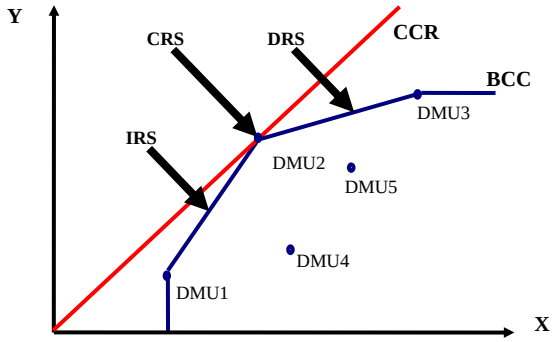

Figura 2 – Fronteira dos modelos CCR e BCC

/

De maneira simples, os autores do modelo original determinaram a eficiência da DMU_0, (modelo do envelope), resolvendo o PPL (8):

$$
\begin{aligned}
&Min \quad \theta \\
&\text{sujeito a:} \\
&\theta x_0 - \mathbf{X}\boldsymbol{\lambda} \geq 0 \\
&-Y_0 + \mathbf{Y}\boldsymbol{\lambda} \geq 0 \\
&\mathbf{I}\boldsymbol{\lambda} = 1 \\
&\boldsymbol{\lambda} \geq 0, \text{ onde I representa um vetor com todos os elementos iguais a 1}
\end{aligned}
\qquad (8)
$$

É importante salientar que o PPL (8) difere do PPL (7) apenas pela introdução da restrição de convexidade.

Semelhantemente ao que foi feito no modelo CCR é possível, nesse caso, derivar o modelo dual (dos multiplicadores) e expressa-lo segundo o PPL (9).

$$
\begin{aligned}
&Max \quad z = uy_0 - u_0 \\
&\text{sujeito a:} \\
&\mathbf{v}x_0 = 1 \\
&\mathbf{v}\mathbf{X} + \mathbf{u}\mathbf{Y} - \mathbf{I}u_0 \leq 0 \\
&u \geq 0, v \geq 0 \text{ e } u_0 \text{ irrestrito,} \\
&\text{onde z e } u_0 \text{ são escalares}
\end{aligned}
\qquad (9)
$$

2.2. REDES NEURAIS ARTIFICIAIS

As redes neurais artificiais (RNAs), são estruturas paralelas compostas de neurônios artificiais, densamente conectadas que, normalmente, apresentam duas fases distintas de operação: a aprendizagem (treinamento) e a execução [3, 4].

Na aprendizagem são apresentados estímulos à entrada, denominados padrões de treinamento e pelos alvos a serem atingidos, mapeamento esse definido pelos especialistas da área em estudo. Na fase de execução, as RNAs recebem como estímulo de entrada um conjunto de padrões de testes, e que não fizeram parte do treinamento.

A arquitetura utilizada foi baseada no perceptron de múltiplas camadas (MLP). A metodologia a ser usada no treinamento da rede MLP pode ser dividido em duas fases principais: "*forward*" e "*backward*". A primeira consiste na propagação dos estímulos apresentados à entrada para a saída. Esses estímulos fluem por toda a rede, recebendo a computação neural, camada por camada, até gerarem a saída .

A partir do resultado desejado (*Target*), calcula-se um erro na camada de saída. A

segunda etapa ocorre em sentido contrário. O erro calculado é então retro propagado pelas camadas antecessoras (*error-back-propagation*), atualizando os pesos das conexões.

A arquitetura usada na programação linear neural é semelhante, porém o problema é tratado de *forma inversa* pois o conhecimento sobre os coeficientes das restrições e da função objetivo (F) é conhecido. O que se deseja no processo iterativo é a determinação das variáveis de decisão que otimizam a função objetivo.

O algoritmo de retro propagação de erro (BP) é baseado no método do gradiente descendente, que computa as derivadas parciais de uma função de erro, com relação ao vetor peso W de certo vetor de entrada [3, 4].

A regra de Widrow-Hoff é a base fundamental dos diversos algoritmos e métodos de treinamento das RNAs. Ela avalia o erro médio quadrático a cada iteração através da derivada parcial do erro quadrático em relação ao peso W e ao limiar (bias) b, mostrados, de forma simplificada na Figura 3.

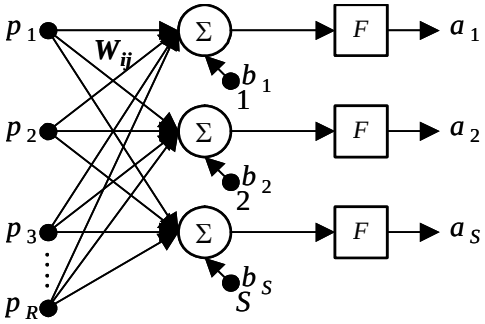

Figura 3 – Rede Neural Artificial

Sabendo-se que (t) representa o vetor alvo a ser atingido no treinamento, (p) representa o vetor de treinamento apresentado à rede, (S) representa o número de elementos do vetor de saída, R representa o número de elementos do vetor de treinamento e (*F*) a função de ativação, e analisando-se o neurônio um na n-esima iteração, para (j = 1, ..., R), tem-se (10):

$$\frac{\partial e^{2(n)}}{\partial W_{1j}} = 2e^{(n)} \frac{\partial e^{(n)}}{\partial W_{1j}}$$

$$\frac{\partial e^{2(n)}}{\partial b} = 2e^{(n)} \frac{\partial e^{(n)}}{\partial b}$$

$$\frac{\partial e^{(n)}}{\partial W_{1j}} = \frac{\partial [t^{(n)} - a^{(n)}]}{\partial W_{1j}} = \frac{\partial}{\partial W_{1j}} [t^{(n)} - Wp^{(n)}] \qquad (10)$$

onde :

$$Wp^{(n)} = \sum_{j=1}^{R} W_{1j} P_j^{(n)} + b$$

$$\frac{\partial e^{(n)}}{\partial W_{1j}} = -P_j^{(n)}$$

$$\frac{\partial e^{(n)}}{\partial b} = -1$$

Em (11) é mostrado a variação ΔW do peso e Δb do limiar, bem como o método numérico usado no processo adaptativo de atualização dos parâmetros internos da rede.

$$\Delta w^{(n)} = -2\eta e^{(n)} \frac{\partial e^{(n)}}{\partial W}$$
$$\Delta b^{(n)} = -2\eta e^{(n)} \frac{\partial e^{(n)}}{\partial b} \qquad (11)$$

$$W^{(n+1)} = W^{(n)} + \Delta w^{(n)} = W^{(n)} + 2\eta e^{(n)} P^{(n)}$$
$$b^{(n+1)} = b^{(n)} + \Delta b^{(n)} = b^{(n)} + 2\eta e^{(n)}$$

3. METODOLOGIA

Para tratar o problema de otimização de forma similar a uma RNA (um problema típico de otimização sem restrição), mesmo que na *forma inversa*, é preciso converter o problema de programação linear (PPL), composto de uma função objetivo e de um conjunto de restrições, em um problema de otimização sem restrições. Para isso é preciso adotar uma nova função de custo, onde é adicionado um termo de penalidade causando alto custo (penalizando) toda vez que, pelo menos, uma das restrições for violada [8, 9].

Como a solução do problema propõe o uso de uma arquitetura semelhante à de uma RNA não convencional, apresentando inclusive realimentação da saída para a entrada, é preciso, antes de resolver o problema numericamente, verificar a questão da estabilidade, visto que o modelo é caracterizado por um sistema de equações diferenciais ordinárias de primeira ordem. Depois de garantida a estabilidade, que não será tratada nessa pesquisa, é possível resolve-lo numericamente pelo método do gradiente.

Não há atualização dos pesos, pois são conhecidos e representam os coeficientes das restrições, da função objetivo e do termo independente. Como o caso investigado é uma arquitetura neural não tradicional, nosso objetivo é recuperar a informação, determinando o valor ótimo das variáveis de decisão do PPL [10, 11].

É importante ressaltar que, em princípio, o modelo proposto nessa pesquisa não pode ser confundido com uma RNA com “retro alimentação” nem com uma rede “recorrente” convencional. Na realidade é um produto híbrido que apresenta algumas peculiaridades comuns desses modelos.

Assim, dado um problema de otimização sem restrição deseja-se encontrar o valor de $x \in \Re n$ que minimize uma função escalar E(x), denominada de Função Pseudo custo, de Energia ou Objetivo [12]. O ponto x* será o mínimo global de E(x) se E (x*) <= E(x) para todo $x \in \Re n$ e um mínimo local se a relação E (x*) <= E(x) for mantida para certo intervalo $\varepsilon > 0$. Se a primeira e a segunda derivadas de E(x) existirem, o ponto x* será um mínimo local se o gradiente ∇ E (x*) = 0 e a matriz de Hessian ∇^2 E (x*) > 0.

Assim, as condições necessárias e suficientes para a existência de um mínimo local são as seguintes: Seja ∇^2 E(x) não singular para o ponto x*, então E (x*) será <= E(x) para todo $0 < \| x - x^* \| < \varepsilon$ e $\varepsilon > 0$ se o gradiente ∇ E (x*) = 0 e a matriz Hessian for simétrica e positiva, isto é ∇^2 E (x*) > 0 [16].

O problema de otimização sem restrição é caracterizado por um sistema de equações diferenciais ordinárias de primeira ordem representada pela equação (12).

$$\frac{dx_j}{dt} = -\eta_j \sum_{j=1}^{n} \frac{\partial E(x)}{\partial x_j}, \qquad (12)$$

onde $x_j(0) = x_j^{(0)}$ são as condições iniciais e η_j o fator de aprendizado.

Logo para se encontrar o valor de x* que leva E(x) a um mínimo é preciso resolver ou simular a solução de um sistema de equações diferenciais submetidas a condições iniciais.

Através da regra de Euler [16] é possível transformar o sistema de equações diferenciais em um sistema de equações diferença, concluindo finalmente que x* possa ser determinado, no limite, através da "curva da trajetória de solução" (13) do sistema proposto:

$$x^* = \lim_{t\to\infty} x(t) \qquad (13)$$

É interessante salientar que no caso de uma RNA convencional, teríamos um sistema de equações diferenciais semelhante, apenas as variáveis determinadas no processo iterativo seriam os pesos sinápticos da RNA, que no caso investigado são conhecidos.

Dos métodos inspirados no "Steepest descent" e no "Método de Newton" [2, 16], o mais difundido é o Método do Gradiente e foi a solução numérica adotada.

Dessa forma, uma função de custo E(x) a qual se deseja minimizar, onde $x_j\ (0) = x_j^{(0)}$ representa o ponto onde o procedimento se inicia e, seja ∇ E (x*) o gradiente de E(x) para o k-ésimo ponto da trajetória então, a idéia é determinar um caminho particular p ao longo do qual dE(x) /dp é minimizada a cada ponto da trajetória. A solução numérica [12, 16] pode ser obtida atendendo o seguinte:

- Se sucessivos pontos x^k e x^{k+1} forem determinados obedecendo a seguinte "*trajetória de solução*": $x^{k+1} = x^k - \eta^k \nabla E(x)^k$, onde $0 <= \eta^k <= \eta^{max}$ é denominado de taxa de aprendizado que pode ser também otimizada.
- O valor de η^k é determinado de modo que x^{k+1} resulte sempre num melhoramento da função objetivo, isto é, $E(x)^{k+1} < E(x)^k$.
- O procedimento termina quando dois pontos sucessivos x^{k+1} e x^k são iguais, isto é, $\eta^k \nabla E(x)^k \cong 0$. Como η^k é $\neq 0$, então $\nabla E(x)^k = 0$.

Ao contrário da RNA, as variáveis determinadas no método numérico iterativo são as variáveis de decisão que, no limite, apresentam os valores ótimos desejados para determinação da eficiência [13, 14, 15].

Nosso problema apresenta restrições. Para resolvê-lo, usando-se a filosofia semelhante à usada nas RNAs, é preciso construir uma "*nova função*" denominada de função pseudo custo ou de energia E(**x**) para qual o mínimo global seja a solução ótima do PPL (14).

$$\text{Min} \sum_{j=1}^{n} c_j x_j$$

$$\text{Sujeito a:} \qquad (14)$$

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i$$

$$b_i \geq 0 \quad \{i = 1,2,3 \ldots m\}$$

$$x_j \geq 0 \quad \{j = 1,2,3 \ldots n\}$$

Para se construir a nova função E(x) incorpora-se uma função (15) ou termo de penalidade r_i+(x) à função objetivo original.

$$r_i^+(\mathbf{x}) = \begin{cases} 0 & \text{se} \quad r_i(\mathbf{x}) \geq 0 \ \text{(inibe)} \\ r_i(\mathbf{x}) & \text{se} \quad r_i(\mathbf{x}) < 0 \ \text{(penaliza)} \end{cases} \qquad (15)$$

$$\text{e} \quad r_i(\mathbf{x}) = \sum_{j=1}^{n} a_{ij}\, x_j - b_i$$

O termo de penalidade deve causar alto custo (penalizar) à nova função toda vez que uma restrição for violada ($r_i(x) < 0$) e custo zero se a restrição for satisfeita ($r_i(x) >= 0$).

O termo de penalidade deve, normalmente, penalizar para os casos de soluções inviáveis e inibir para soluções viáveis do PPL.

Dessa forma o PPL é transformado em um problema de otimização sem restrição,

onde se deseja encontrar x* $\in \Re$n que minimize a nova função E(x).

Agora o problema de otimização sem restrição e com termo de penalidade pode ser resolvido de maneira "*inversa*", mas semelhante ao usado no treinamento das RNAs, ou seja, pela aplicação do método do gradiente decrescente ou suas variações.

Nesse caso, a equação da solução x_j (k+1) = x_j (k) + Δ (x_j) (onde j=1... n, representa o número de variáveis e k a k-ésima iteração do método numérico) indica na convergência o valor das variáveis de decisão do problema e Δ (x_j) o produto do negativo do gradiente da função de custo em relação à x_j por um fator constante [13, 14, 15, 17] ligado à velocidade de convergência e à estabilidade do método.

Como ponto de partida, a função de pseudo custo pode ser seguinte:

$$E(\mathbf{x}) = \sum_{j=1}^{n} C_j x_j - p \sum_{i=1}^{m} r_i^{+}(\mathbf{x}), \tag{16}$$

onde, $p > 0$ representa o parâmetro de penalidade

(17)

Se $r_i(x) \geq 0$ $\quad D_i = 0$ (inibe)

Se $r_i(x) < 0$ $\quad D_i = -p$ (penaliza)

$p > 0$ e $\eta_j > 0$ e D_i é um degrau negativo (18)

Como, $r_i(x) \cong a_{ij} x_j$, então :

$\nabla_{x_j} E(x_j) = C_j + \sum_{i=1}^{m} D_i a_{ij}$ onde D_i é um degrau negativo

Do método do gradiente decrescente temos que : (19)

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

$\Delta(x_j) = -\eta_j \nabla_{x_j} E(x_j)$, logo : (20)

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

Podemos então obter a equação da solução : (21)

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

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

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

$p > 0$, $\eta_j > 0$, i = 1,2,..., m, j = 1,2,..., n, $k = 1,2,....$

4. MODELAGEM

A Figura 4 mostra um trecho da arquitetura final do modelo de programação linear neural, no qual é verificado o atendimento das restrições. A restrição 1 é dada por: $r_1(x) = (a_{11}x_1+a_{12}x_2+ ... +a_{1n}x_n) – b_1$ e a saída do bloco (*0 ou –p*) é dada pela função degrau negativo.

Os blocos referentes aos coeficientes a_{ij} serão implementados usando-se amplificadores, o Σ por módulos somadores/subtratores e a função degrau por um gerador de função específico, simulando circuitos eletrônicos. Assim, se $r_1(x) = (a_{11}x_1+a_{12}x_2+ ... +a_{1n}x_n) – b_1>=0$, a saída $D_1=0$, caso contrário será – p.

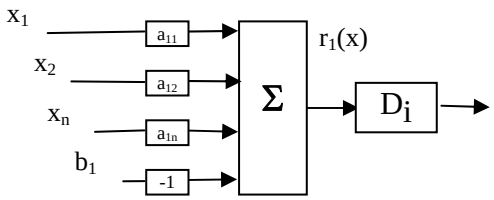

Figura 4 – Módulo referente à verificação da restrição

A equação (21) mostra que além do degrau necessitamos dos coeficientes das restrições e da função objetivo que, juntamente com o passo de aprendizado (ou passo de integração), completam o segundo membro da equação representado por $-\eta[c_j + \sum_{i=1}^{m} D_i a_{ij}]$.

A Figura 5 recebe o sinal de saída dos degraus referentes a cada restrição, processa esse sinal juntamente com os coeficientes das restrições $\sum_{i=1}^{m} D_i a_{ij}$ e soma os coeficientes da função objetivo $c_j + \sum_{i=1}^{m} D_i a_{ij}$. O sinal resultante é multiplicado pelo passo de aprendizado $-\eta[c_j + \sum_{i=1}^{m} D_i a_{ij}]$.

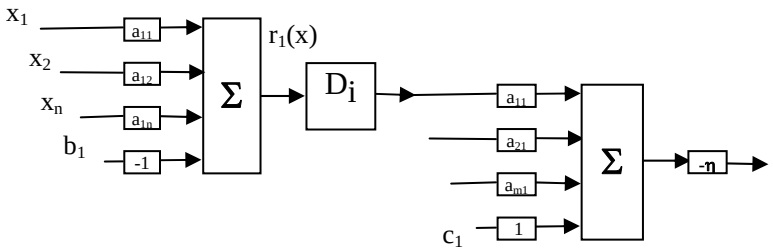

Figura 5 – Módulo referente à primeira restrição e à função objetivo

A saída nesse ponto ainda não é igual ao valor da variável de decisão. Como estamos resolvendo numericamente um sistema de equações diferenciais ordinárias de primeira ordem nesse ponto temos a derivada da função dx_1/dt.

Assim, é necessária a inclusão de um último módulo responsável pela a operação de integração $x_1=\int dx_1/dt$ e pela fixação das condições iniciais do problema $x_1^{(0)}$.

A Figura 6 apresenta a arquitetura completa do modelo de programação linear neural.

Nos tópicos anteriores foram mostrados os principais modelos consagrados em análise DEA e fundamentada a metodologia e modelagem da programação neural que será usada para construir a arquitetura DEA-VRS Neural.

Adotamos como base de nossa arquitetura o modelo BCC – envelope, embora inicialmente os autores tenham estudado profundamente o modelo CCR [5]. O modelo do envelope é caracterizado resolvendo-se o PPL mostrado no conjunto de equações (8) que determina a eficiência da DMU_0. A única modificação relativa ao modelo (7) é a introdução da restrição de convexidade.

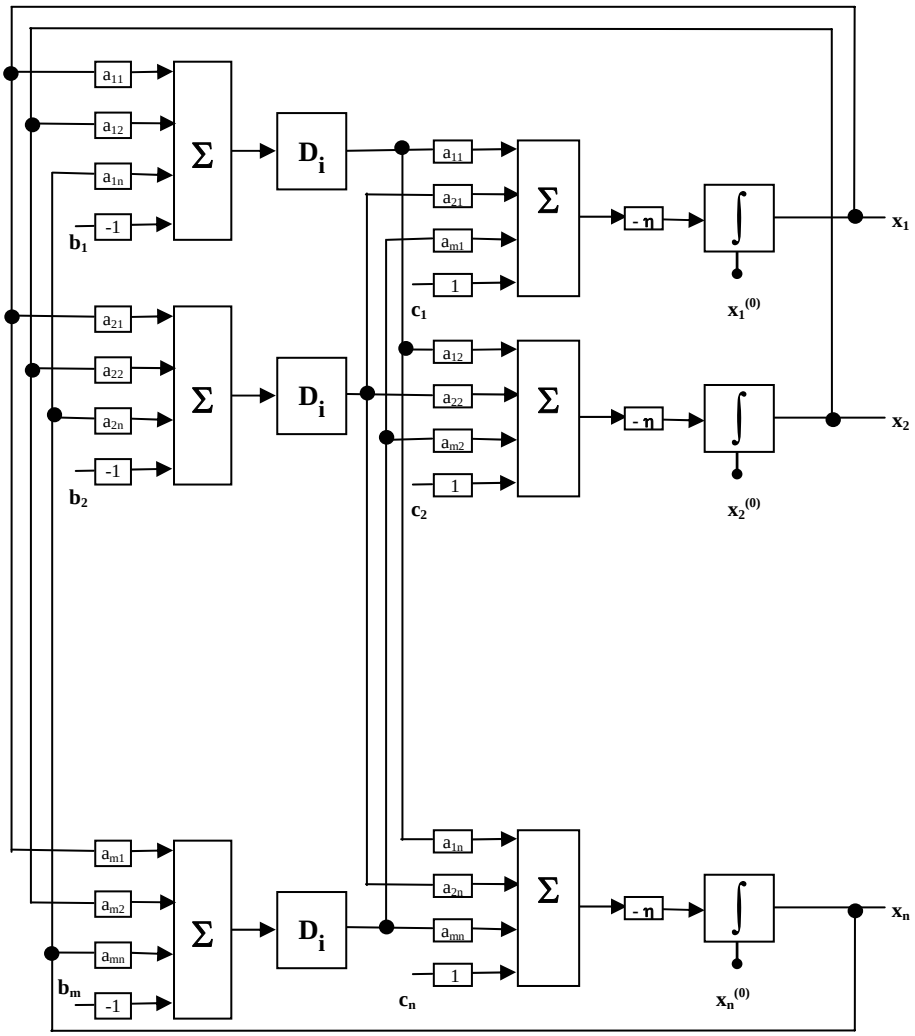

Figura 6 – Arquitetura do modelo de programação linear neural

Considerando-se uma situação envolvendo cinco DMUs com duas entradas e uma saída, mostrada de forma literal na Tabela 1, vamos aplicar o modelo proposto (8).

Tabela 1 - Dados referentes às DMUs

	Y	**X_1**	**X_2**
DMU 1	y_1	x_{11}	x_{21}
DMU 2	y_2	x_{12}	x_{22}
DMU 3	y_3	x_{13}	x_{23}
DMU 4	y_4	x_{14}	x_{24}
DMU 5	y_5	x_{15}	x_{25}

A equação (22) mostra o modelo referente à Tabela 1 e determina a eficiência da DMU 1. A parte ressaltada não sofre modificação no cálculo da eficiência das demais DMU.

$$
\begin{array}{l}
\textbf{Min } \theta \\
\text{s. a.} \\
\begin{array}{l}
0 \\
+\mathbf{x_{11}}\theta \\
+\mathbf{x_{21}}\theta \\
0 \\
0 \\
0
\end{array}
\begin{array}{|ccccc|l|}
\hline
+y_1\lambda_1 & +y_2\lambda_2 & +y_3\lambda_3 & +y_4\lambda_4 & +y_5\lambda_5 & \geq +\mathbf{y_1} \\
-x_{11}\lambda_1 & -x_{12}\lambda_2 & -x_{13}\lambda_3 & -x_{14}\lambda_4 & -x_{15}\lambda_5 & \geq 0 \\
-x_{21}\lambda_1 & -x_{22}\lambda_2 & -x_{23}\lambda_3 & -x_{24}\lambda_4 & -x_{25}\lambda_5 & \geq 0 \\
\lambda_1 & \lambda_2 & \lambda_3 & \lambda_4 & \lambda_5 & \geq 1 \\
-\lambda_1 & -\lambda_2 & -\lambda_3 & -\lambda_4 & -\lambda_5 & \geq -1 \\
0 & 0 & 0 & 0 & 0 & \geq 0 \\
\hline
\end{array} \\
\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5 \geq 0
\end{array}
\qquad (22)
$$

Para determinar a eficiência das outras DMUs basta substituir os termos em negrito pelos correspondentes aos da DMU que se deseja obter a eficiência.

5. RESULTADOS

A Tabela 2 mostra um exemplo hipotético envolvendo 5 DMUs com duas entradas e uma saída. O objetivo principal é verificar a exatidão do método proposto nesse artigo e também confirma na Figura 8 a elevada rapidez na convergência atingida na simulação para determinação da eficiência relativa da DMU-5. Os resultados numéricos são compatíveis com os obtido pelo SIAD (Sistema Integrado de Apoio à Decisão) [23] apresentando erros inferiores a 1%.

Tabela 2 - Dados e resultados de um exemplo hipotético

	DADOS			**RESULTADOS**		
	Y	X1	X2	DEA-VRS ***SIAD***	**DEA-VRS NEURAL**	ERRO %
DMA - 1	2	6	8	0,666	**0,671**	0,745
DMA - 2	4	4	8	1,000	**1,000**	0,000
DMA - 3	3	6	6	0,925	**0,920**	0,597
DMA - 4	1	4	3	1,000	**0,999**	0,100
DMA - 5	4	12	4	1,000	**0,995**	0,472

A Figura 7 mostra os resultados obtidos através do SIAD [23] e serve de base de comparação no cálculo do erro percentual dos resultados obtidos pelo modelo proposto.

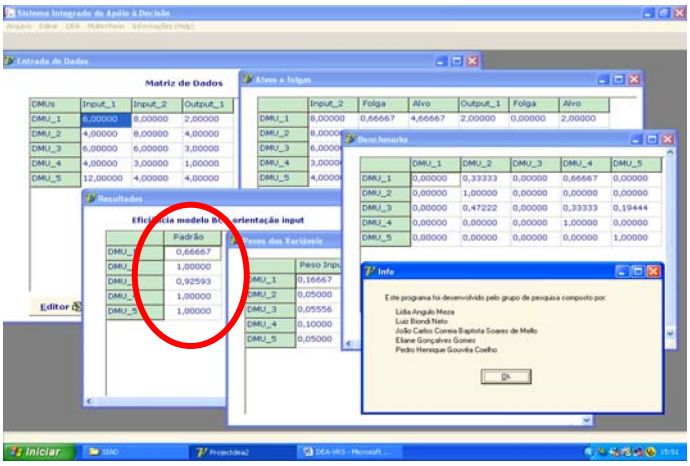

Figura 7 - Resultados obtidos pelo SIAD

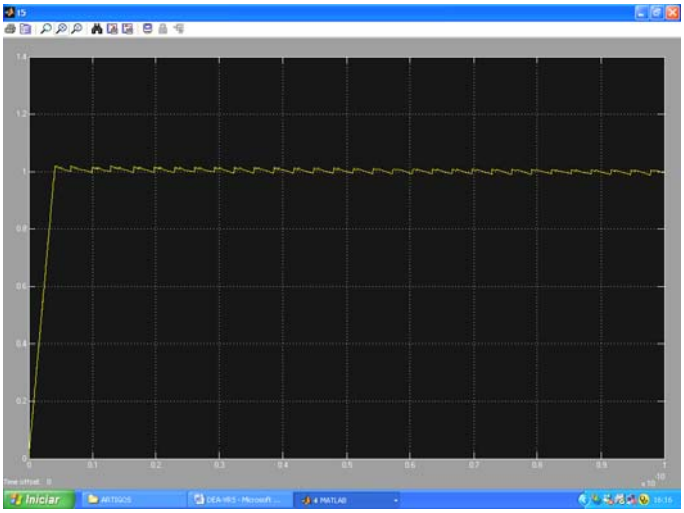

Figura 8 - Resultado da Eficiência da DMU-5 obtido pelo DEA-VRS Neural

6. CONCLUSÕES

A contribuição do procedimento desenvolvido nessa pesquisa reside na rapidez da convergência dos módulos de programação linear neural. Aplicando-se o método proposto em modelos de até 60 variáveis, o tempo de resposta ficou na casa dos 10^{-9} segundos, Figura 9, contra a cerca 10^{-3} segundos conseguidos com softwares comerciais.

Com relação ao DEA–VRS Neural as vantagens são as mesmas quanto a convergência visto que o mesmo herda as características dos módulos de programação linear neural. A exatidão dos resultados obtidos pelo DEA-VRS Neural no exemplo hipotético e nos demais testes com um maior número de variáveis indica um erro percentual de cerca de 1 %.

Finalmente é importante salientar que um grande número de casos distribuídos nas áreas ligadas à DEA, não exigem rapidez na resposta ao usuário. Assim, os problemas podem ser resolvidos usando-se pacotes específicos de softwares disponíveis no mercado que se beneficiam da freqüência de operação, atualmente muito alta e da memória de acesso randômico (RAM) dos modernos computadores, cada vez maiores. Quando se exige resposta rápida é necessário um sistema com alta velocidade de convergência, como o proposto.

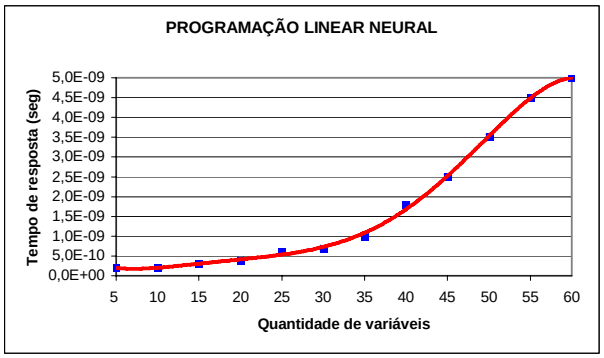

Figura 9 - Velocidade de Convergência X Quantidade de Variáveis

Se aliado às características de alta velocidade de convergência o sistema simulado nessa pesquisa, puder ser integrado em um “chip” dedicado poderá ser construída uma placa contendo esses “chips” e conectada a um “slot” livre de um computador permitindo a execução, praticamente em tempo real, dos principais modelos de otimização.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] BAZARAA, M.S., JARVIS, J. J., 1977, Linear Programming and Network Flows, John Wiley & Sons Inc, New York, USA.

[2] BERTSEKAS, D.P., 1999, Nonlinear Programming, Athena Scientific Publisher’s, Belmont, Mass. USA.

[3] ZURADA, J.M., 1992, Introduction to Artificial Neural Systems, London, West Publishing Company.

[4] HAYKIN, S., 1994, Neural Networks a Comprehensive Foundation, London, Macmillan College Publishing Co..

[5] BIONDI NETO, Luiz; LINS, Marcos P Estellita. Neuro-DEA. In: LINS, Marcos P. Estellita; ANGULO, Lidia.(Org.). 2000, Análise de Envoltória de Dados e Perspectiva de Integração no Ambiente de Apoio à Decisão. Rio de Janeiro, p. 177-197.

[6] BIONDI, L.N., LINS, M.P.E., CHIGANER, L., FUKUDA, F.H., VINCENZO, R.J., RESTUM, A.E., 2000, “Neuro-DEA: Novo paradigma para determinação da eficiência relativa de unidades tomadoras de decisão”, 9º APDIO, Setúbal, Portugal, 19/4-19/4.

[7] SEIFORD, L.M., THARALL, R.M., 1990, “Recent Developments in DEA – The Mathematical Programming Approach to Frontier Analysis”, J. of Econ., n. 46, pp.7-38.

[8] UNBEHAUEN, R., HE, S., REI, K., 2000, "Multilayer Neural Networks for solving a Class of Partial Differential Equations", Neural Networks, n.13, pp.385-396.

[9] AGUILAR, J., 1998, "Definition of an Energy Function for the Random Neural to Solve Optimization Problems", Neural Networks, n. 11, pp. 731-737.

[10] CHEN, J., SHANBLATT, M., MAA, C.Y., 1992, "Improved Neural Network for Linear and Nonlinear Programming", I. Journal of Neural Systems, n. 2, pp. 331-339.

[11] KENNEDY, M.P.S., CHUA, L., 1994, "Neural Networks for Nonlinear Programming", IEE Transactions on Circuits and Systems, n. 35, pp. 554-562.

[12] CICHOCKI, A., UNBEHAUEN, R., WEINZIERL, K., HÖLZEL, R., 1996, "A New Neural Network for Solving Linear Programming Problems", European Journal of Operational Research, n. 93, pp.244-256.

[13] HONG-XING, L., XU-LI, D., 2000, "A Neural Representation of Linear Programming", European Journal of Operational Research, n. 124, pp.224-234.

[14] ZHU, X., ZHANG, S., CONSTANTINIDES, A.G., 1992, "Lagrange Neural Networks to Linear Programming", Journal of Parallel Distributed Computing, n. 14, pp. 354-360.

[15] BARGIELA, A., CICHOCKI, A., 1996, "Neural Networks for Solving Linear Inequality Systems", Journal of Parallel Computing. Disponível em http://www.bip.riken.go.jp/absl.

[16] DENNIS, J.E., SCHNABEL, R.B., 1996, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Englewood Cliffs, N.J., USA, Prentice Hall.

[17] BIONDI NETO, L.; DUNCAM, L. A.; CHIGANER, L.; LINS, M. P. E., 2000, "Data Envelopment Analisys: The High Speed Convergence Approach", AIRO 2000, Milano, Itália.

[18] BIONDI NETO, L.; LINS, M. P. E.; CHIGANER, L.; GOMES, E. G.; MELLO, J. C. S., 2001, "Data Envelopment Analysis: A Neural Approach", ICORD 2001, Pretoria, África do Sul.

[19] BIONDI NETO, L.; LINS, M. P. E.; CHIGANER, L.; FUKUDA, F. H.; ROBERTO, V.; RESTUM, E.; SALDANHA, F. O., 2001, "Neuro-DEA: A New optimization Algorithm", V CBRN, Rio de Janeiro, Brasil.

[20] BIONDI NETO, L.; CHIGANER, L.; FUKUDA, F. H.; LINS, M. E.; RESTUM, E.; ROBERTO, V. , 2000, "Programação Linear Neural - Neuro – LP", XXV CEIO, Vigo XXV Congresso Nacional de Estadístita e Investigación Operativa, Espanha.

[21] BIONDI NETO, Luiz; LINS, Marcos P Estellita; GOMES, Eliane Gonçalves; MELLO, João Carlos Correia Baptista Soares; SALDANHA, Fabiano Oliveira. , 2004, "Neural Data Envelopment Analysis: A Simulation". International Journal Of Industrial Engineering, Estados Unidos da América, v. 1, n. 1, p. 14-24.

[22] BIONDI NETO, Luiz; MELLO, João Carlos Correia Baptista Soares; MEZA, Lidia Angulo; COELHO, Pedro Henrique Gouvêa; SALDANHA, Fabiano Oliveira; GOMES, Eliane Gonçalves., 2004, "NEURO-DEA: Uma versão VRS para Medida de Eficiência Relativa". In: 11º APDIO, Porto - Portugal. IO - 2004 v. 1, p. 85-85.

[23] MEZA, Lidia Angulo; BIONDI NETO, Luiz; MELLO, João Carlos Correia Baptista Soares; GOMES, Eliane Gonçalves; COELHO, Pedro Henrique Gouvêa., 2005, "Free Software for Decision Analysis: A Software Package for Data Envelopment Models", ICEIS 2005, v. 02, pp 207-212.