APLICAÇÃO DE UMA REDE NEURAL AUMENTADA AJUSTADA POR DELINEAMENTO DE EXPERIMENTOS PARA RESOLUÇÃO DE PROBLEMAS DE CORTE E EMPACOTAMENTO

Ricardo de Almeida

Pontifícia Universidade Católica do Paraná - PUCPR

r_almeida80@hotmail.com

Maria Teresinha Arns Steiner

Pontifícia Universidade Católica do Paraná - PUCPR

maria.steiner@pucpr.br

RESUMO

O objetivo do presente trabalho é mostrar a utilização da Rede Neural Artificial Aumentada (RNAA) como uma alternativa promissora para resolução de Problemas de Otimização Combinatória, especificamente, neste caso, em Problemas de Corte e Empacotamento (PCE). PCEs são vastamente encontrados em diversos ramos da indústria e o tratamento adequado deste tipo de problema pode gerar impactos diretos na economia de matérias-primas e/ou espaço físico das empresas. Para auxiliar a otimização dos parâmetros da RNAA, fez-se uso de um projeto de Delineamento de Experimento (DOE) do tipo Fatorial Completo. Os testes foram desenvolvidos em diversos problemas *benchmark* da literatura e os resultados se mostraram bastante satisfatórios, tanto para eficácia da RNAA na resolução de PCEs, como para a efetividade do DOE no ajuste de parâmetros da rede.

PALAVRAS CHAVE: Rede Neural, Corte e Empacotamento, Delineamento de Experimentos.

ABSTRACT

The objective of this work is to show the Augmented Neural Network (AugNN) as a promising alternative to solve Combinatorial Optimization Problems, specifically, in this case, Cutting and Packing Problems (CPP). CPPs are easily found among various industry sectors and its proper treatment can impact directly in savings of raw material and/or physical space of enterprises. In order to help the adjustment of parameters of AugNN a Full Factorial Design of Experiment (DOE) was applied. Tests were developed in many benchmark problems found in the literature and satisfactory results were achieved, both in effectiveness of AugNN in solving CPPs and in the effectiveness of DOE in adjusting parameters of network.

KEYWORDS: Neural Network, Cutting and Packing, Design of Experiments

1. INTRODUÇÃO

No contexto dos problemas de otimização combinatória, classificados como NP-difíceis, os procedimentos meta-heurísticos têm se consagrado como poderosa alternativa na busca por soluções quase ótimas, ou até mesmo ótimas. Apesar de ceticismo ocasionado pela falta de *insights* teóricos [16], a literatura é farta em evidências empíricas que demonstram sua efetividade.

Falkenauer [8] utiliza um algoritmo de grupamento genético híbrido nos problemas de *Bin Packing*. Loh *et al*. [11] apresentam o algoritmo *Weight Annealing* para resolução do problema de *Bin Packing*. Woodcock e Wilson [18] testam um método híbrido de Busca Tabu e *Branch & Bound* para problemas de designação. Araujo *et al.* [1] aplicam um algoritmo evolucionário na resolução de um problema de corte de estoque com objetos variados e limitados. Chen *et al.* [4] aplicam a meta-heurística *Simulated Annealing* para resolver a formulação de Programação Linear Inteira de um problema de corte de estoque unidimensional, utilizando análise estatística para verificar os efeitos dos diversos parâmetros na eficiência e precisão da solução.

Uma característica importante dos métodos meta-heurísticos é que, ao mesmo tempo em que se apresentam como ferramentas flexíveis e adaptáveis a vários problemas, geram críticas devido à grande possibilidade de manipulação de seus parâmetros. Dependendo do problema e do método, os parâmetros podem assumir infinitas combinações de ajustes. Uma ferramenta poderosa para auxiliar no ajuste de parâmetros é o Delineamento de Experimentos (DOE – *Design of Experiments*). Com o DOE é possível, além de estimar os efeitos que mudanças nos fatores causam na variável de resposta, verificar como os fatores interagem, subsidiando uma calibragem que busque a otimização do processo ou sistema [13]. Ruiz *et al.* [15] lançam mão de DOE para ajuste dos parâmetros de dois Algoritmos Genéticos. Demirel e Toksari [5] utilizam um experimento do tipo $2k$ para ajuste de um algoritmo de Colônia de Formigas. Pan e Ruiz [14] utilizam DOE para calibração de um algoritmo de busca local para resolver o problema de Programação Fluxo de Produção.

Este trabalho foi estruturado da seguinte forma: na próxima seção o contexto do Problema de Corte e Empacotamento (PCE) será apresentado, a seguir, na seção 3, será feita uma breve revisão literária sobre a Rede Neural Artificial Aumentada (RNAA), seguida pelo detalhamento do seu funcionamento. Na seção 4 será mostrada a metodologia utilizada nos testes. A seção 5 mostra os resultados e uma breve análise destes. Por fim, na seção 6 encontram-se as conclusões do trabalho.

2. PROBLEMA DE CORTE E EMPACOTAMENTO

Apesar de compartilharem a mesma estrutura lógica, tanto os problemas de corte como de empacotamento são encontrados de diversas formas na literatura, desde *Trim Problem* apresentado em [7], até alocação de memória, passando por *Bin Packing*, problema da mochila, carregamento de veículos, particionamento, entre outros. A Dyckhoff [6] é atribuído o primeiro trabalho com objetivo de integrar os vários problemas e tipologias encontrados no âmago dos PCEs.

De acordo com Dyckhoff [6], os dois principais grupos de dados do PCE são os **estoques** de **objetos** grandes e as **ordens** de **itens** pequenos. Itens são combinados formando **padrões de corte**, que serão designados aos objetos. As sobras decorrentes do padrão de corte são tratadas como **perdas de corte**. Os PCEs pertencem ao campo de geometria combinatória, sendo que os objetos e itens podem ser definidos por **1, 2 ou 3 dimensões** do espaço Euclidiano.

O Problema de corte/empacotamento consiste basicamente em “retirar/encaixar” itens de/em objetos, levando em consideração algum critério de desempenho, geralmente a

minimização das perdas de corte. É um problema de grande relevância na indústria, visto que diversos processos necessitam de transformação de objetos em itens menores, de forma a suprir uma determinada demanda. Exemplos práticos podem ser o corte de perfis na indústria metalúrgica ou a alocação otimizada de *pallets*.

3. REDE NEURAL AUMENTADA

Rede Neural Artificial (RNA) é uma técnica poderosa para solução de problemas de previsão, classificação e reconhecimento de padrões, entretanto não têm alcançado o mesmo sucesso na resolução de Problemas de Otimização Combinatória (POC) [17]. Os resultados obtidos através desta técnica, porém, em termos de qualidade de solução, têm sido comparáveis aos de outras meta-heurísticas. Ainda, segundo Smith [17], a primeira demonstração de resolução de um POC utilizando RNA foi feita por Hopfield e Tank [9], em um Problema de Caixeiro Viajante (PCV). Este trabalho serviu de inspiração para a Rede Neural Aumentada, proposta por Agarwal *et al.* [3].

A primeira Rede Neural Artificial Aumentada (RNAA) foi utilizada como alternativa para resolução de um problema de agendamento de tarefas [3]. A RNAA é uma meta-heurística híbrida que combina um método heurístico a uma RNA. Nas RNAAs, o método heurístico é construído em uma estrutura de rede neural, com elementos interligados, onde funções de entrada são transformadas em funções de saída através de funções de ativação. Estas funções são modeladas de forma a capturar as características e restrições dos problemas sob a ótica da heurística utilizada, permitindo grande flexibilidade de construção de arquiteturas de rede. A mudança controlada de pesos entre os elementos de processamento é responsável pela característica de busca local por soluções melhores. Hopfield e Tank [10] propõem uma RNAA para resolução de um PCE, que é estruturada como mostra a figura 1.

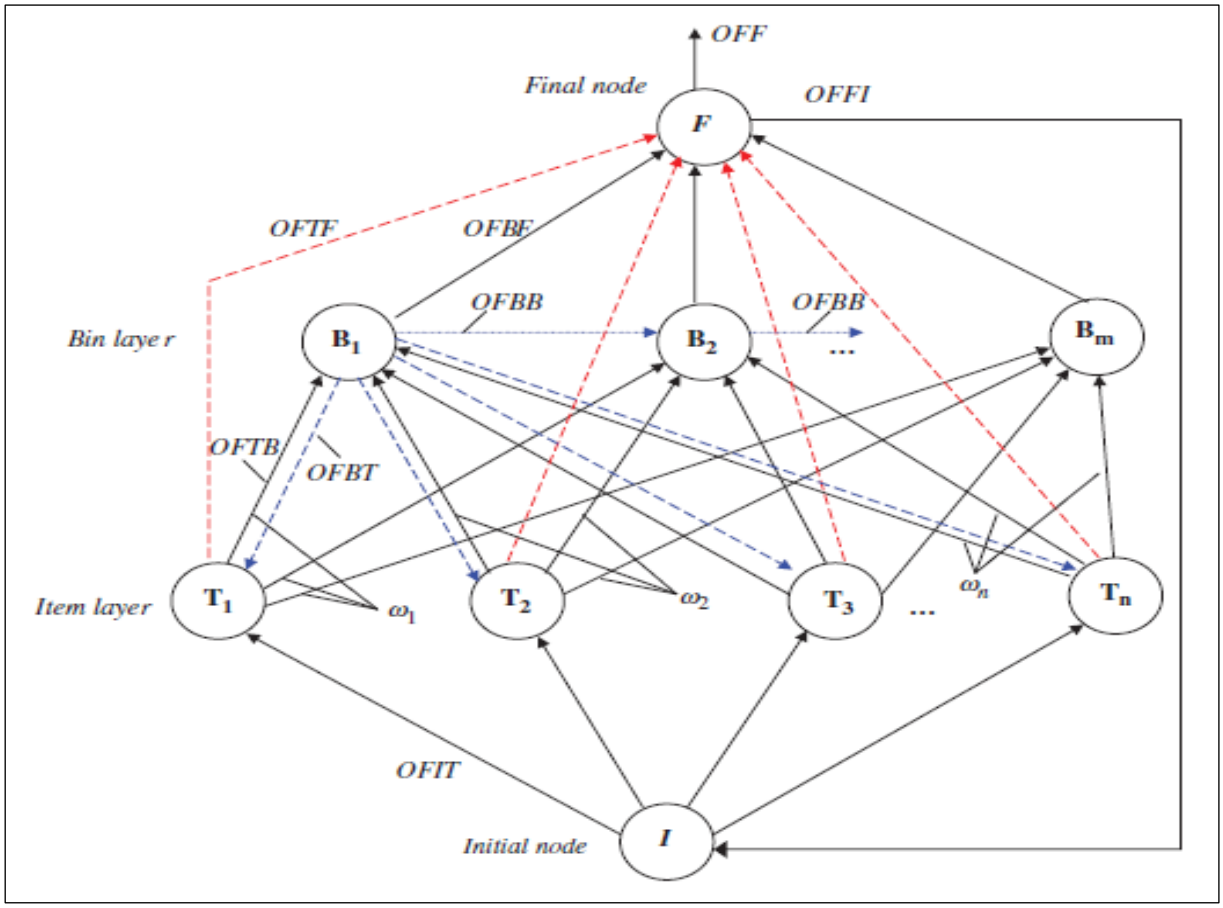

Figura 1: Estrutura da Rede Neural Aumentada para Problema de Corte e Empacotamento. Kasap e Agarwal [10].

A seguir, a notação utilizada para a RNAA:

n - número de itens.
m - número de objetos (UB).
T - conjunto de itens (1, 2,...,n) (Camada de itens/camada de entrada).
B - conjunto de objetos (1,2,...,m) (Camada de objetos/camada escondida).
C - capacidade do objeto.
k - número de iterações.
t - iteração de designação [0,n].
I - nó inicial.
F- nó final.
T_i - i-ésimo nó da camada de itens, $i \in T$.
B_j - j-ésimo nó da camada de objetos, $j \in B$.
S_i - tamanho do item $i, i \in T$.
SUI - conjunto de itens não designados.
LB - limite inferior da quantidade de objetos.
UB - limite superior da quantidade de objetos.
RF - fator de reforço.
BF - fator de retorno.
$\propto$ - taxa de aprendizagem/coeficiente de busca.
$IFI(t)$ - função de entrada do nó inicial.
$IFT_i(t)$ - função de entrada para os nós dos itens $T_i, i \in T$.
$IFB_{ij}(t)$ - função de entrada do nó T_i para os nós dos objetos $B_j, j \in B, i \in T$.
$IFFT(t)$ - função de entrada do nó final a partir dos nós dos itens.
$IFFB(t)$ - função de entrada do nó final a partir dos nós dos objetos.
$OFI(t)$ - função de saída do nó inicial.
$OFTB_i(t)$ - função de saída dos nós dos itens T_i para os nós dos objetos, $i \in T$.
$OFTF_i(t)$ - função de saída dos nós dos itens T_i para o nó final, $i \in T$.
$OFBF_j(t)$ - função de saída dos nós dos objetos B_j para o nó final, $j \in B$.
$OFBT_{ji}(t)$ - função de saída dos nós dos objetos B_j para os nós dos itens $T_i, i \in T, j \in B$.
$OFBB_j(t)$ - função de saída do nó do objeto B_j para o objeto $B_{j+1}, j \in B, j \neq m$.
$OFFI(t)$ - função de saída do nó final.
$\theta I(t)$ - função de ativação do nó inicial.
$\theta T_i(t)$ - função de ativação para os nós dos itens $T_i, i \in T$.
$\theta B_j(t)$ - função de ativação dos nós dos objetos $B_j, j \in B$.
$\theta F(t)$ - função de ativação do nó final.
$assign_{ij}(t)$ - designação do item i para o objeto $j, i \in T, j \in B$.
$RC_j(t)$ - capacidade residual do objeto $j, j \in B$.
$OFF(k)$ - função de saída do nó final na iteração k.
$\omega_i(k)$ - peso da ligação entre os nós dos itens T_i para os nós dos objetos na iteração $k, i \in T$.
$\varepsilon(k)$ - erro ou diferença entre a solução corrente e o limite inferior na iteração k.

A heurística utilizada como base para construção da RNAA analisada neste trabalho é conhecida com *First-Fit-Descending* (FFD). Este é um algoritmo bastante eficiente que consiste em, após organizar os itens do maior para o menor tamanho, designar cada item (RC_j), em ordem crescente de índice, no primeiro objeto aberto em que a capacidade residual seja suficiente. Caso o tamanho do item a ser designado seja maior que a capacidade residual dos objetos abertos, um novo objeto é aberto [12]. Outras heurísticas que poderiam ser utilizadas são: *Next-Fit* (NF), *First-Fit* (FF), *Best-Fit* (BF), *Next-Fit-Descending* (NFD) e *Best-Fit-Descending* (BFD).

3.1. FUNCIONAMENTO DA RNAA.

Primeiramente, os pesos $\omega_i\,(0)$ são inicializados em 1.00. Também são calculados os limites inferior (LB) e superior (UB), ou seja, a quantidade mínima e máxima de objetos necessários para colocação dos itens. O limite inferior pode ou não ser a solução ótima. O limite superior será usado para definir a quantidade m de nós na camada dos objetos, conforme as equações a seguir.

$$LB = \lceil(\sum_{i\in T} S_i)/C\rceil,$$

$$UB = \lceil n/(C/\max_i(S_i))\rceil, i \in T.$$

3.1.1. Nó inicial

Iniciam-se as iterações, sendo atribuído t = 0.

Função de entrada. No início do algoritmo (t = 0), a função de entrada recebe sinal "1" para inicializar a primeira designação da primeira iteração. Após, quando $t > 0$, a função de entrada recebe sinal do nó final.

$$IFI(0) = 1,$$

$$IFI(t) = OFFI(t), para\ t > 0.$$

Função de ativação. O estado do nó inicial é definido por t e k, que são inicializados em "1".

$$\theta I(0): \{t = 1, k = 1\}.$$

Para $t > 0$, t e k são atualizados de acordo com o sinal da função de entrada, $IFI(t)$, sendo:

$$\theta I(t) = \begin{Bmatrix} t = t + 1\ e\ k = k, & \{se\ IFI(t) = 1,\} \\ t = 1\ e\ k = k + 1, & \{se\ IFI(t) = 2,\} \\ t = 0\ e\ k = 0, & \{se\ IFI(t) = 3.\} \end{Bmatrix},$$

onde IFI = 1 indica nova designação (incrementa t) , IFI = 2 indica fim de uma iteração (incrementa k) e IFI = 3, fim do problema.

Função de saída. Sempre que $t > 0$, o problema precisa ser resolvido, assim, o nó inicial envia o sinal "1" para a camada de itens, indicando que se algum item ainda não está designado, deverá ser.

$$OFI(t) = \begin{Bmatrix} 1, & \{se\ t > 0,\} \\ 0, & \{caso\ contrário.\} \end{Bmatrix}.$$

3.1.2. Camada de itens

Função de entrada.

$$IFT_i(t) = OFI(t), i \in T.$$

Função de ativação. O estado "1" indica que o nó do item T_i ainda não foi designado; o estado "0" indica que T_i se encontra designado. Em $t = 0$, todos os nós são inicializados em "1".

$$\forall\, i\ \in T, j\ \in B,$$

$$\theta T_i(0) = \ 1,$$

$$\theta T_i(t) = \begin{Bmatrix} 0, & \{se\ \theta T_i(t-1) = 0\ \vee (\theta T_i(t-1) = 1\ \wedge\ OFBT_{ji}(t) = 1), \\ 1, & \{se\ \theta T_i(t-1) = 1\ \vee (\theta T_i(t-1) = 0\ \wedge\ IFI_i(t) = 2). \end{Bmatrix}, t > 0.$$

Função de saída. O sinal $OFTB$ envia um sinal, com o tamanho do item multiplicado pelo peso, para a camada de objetos, onde o maior valor será priorizado. Se o item já estiver designado ($\theta T_i(t) = 0$), $OFTB$ é "0". $OFTF$ envia um sinal para o nó final informando se o item está ("1") ou não ("0") designado.

$$\forall\, i\ \in T,$$

$$OFTB_i(t) = \ \theta T_i(t) * S_i * \omega_i(k),$$

$$OFTF_i(t) = \begin{Bmatrix} 1, & \{se\ \theta T_i(t) = 0, \\ 0, & \{caso\ contrário. \end{Bmatrix}.$$

3.1.3. Camada de objetos

Na camada de objetos a função de ativação precede a função de entrada, por fornecer informação para esta.

Função de ativação. No início, o primeiro objeto está aberto e os demais estão não abertos. Um novo objeto é aberto quando recebe sinal do objeto anterior ($OFBB_{j-1}(t)$). Este envia o sinal para o próximo objeto quando não possui capacidade residual suficiente para designar o item com máximo $OFTB_i(t)$. Quando a capacidade residual é inferior ao menor item não designado, o objeto é fechado.

$$RC_i(1) = \ C,$$

$$\theta B_1(1) = \ 1, \big(o\ estado\ do\ primeiro\ objeto\ na\ primeira\ iteração\ \text{é}\ 1\ (aberto)\big),$$

$$\text{Para } j > 1\ \wedge\ j \in B\ e\ t > 1,$$

$$\theta B_j(1) = \ 0,$$

$$\theta B_j(t) = \begin{Bmatrix} 0, & \{se\ \theta B_j(t-1) = 0\ \vee IFI(t) = 2: objeto\ não\ aberto, \\ 1, & \{\theta B_j(t-1) = 1\ \vee \big(\theta B_j(t) = 0\ \wedge\ OFBB_{j-1}(t) = 1\big): objeto\ aberto, \\ 2, & \{se\ \theta B_j(t-1) = 1\ \wedge\ RC_j(t) < \min[S_l], l\ \in SUI: objeto\ fechado. \end{Bmatrix}$$

$$RC_j(t) = \ RC_j(t) - \ S_i, onde\ i\ \text{é}\ o\ índice\ para\ max\ OFTB_i(t).$$

Função de entrada. Se o objeto está aberto, este aceita como entrada a saída máxima da camada dos itens.

$$\forall\, i\ \in T, j \in B,$$

$$IFB_j(t) = \begin{Bmatrix} \max_i \big(OFTB_j(t)\big), & \{se\ \theta B_j(t) = 1, \\ 0, & \{se\ \theta B_j(t) = 0\ \vee\ \theta B_j(t) = 2. \end{Bmatrix}.$$

Designação de item para objeto. Quando um item é designado para um objeto, os demais objetos não tentam designar este item, seguindo a lógica da heurística FFD.

$$assign_{ij}(t) = \begin{Bmatrix} 0, & \{se\ S_i > \ RC_j(t), \\ 1, & \{se\ S_i \leq \ RC_j(t). \end{Bmatrix}, onde\ i\ \text{é}\ o\ índice\ para\ max\ OFTB_i(t).$$

Função de saída.

$$\forall\, i\ \in T, j\ \in B,$$

$$OFBF_j(t) = \begin{Bmatrix} 1, & \{se\ \ \theta B_j(t) = 2, \\ 0, & \{caso\ contrário. \end{Bmatrix}.$$

Quando um objeto é fechado (estado "2"), o sinal $OFBF_j(t)$ é enviado para o nó final, que a partir deste sinal faz a contagem de objetos fechados.

$$OFBB_j(t) = \begin{Bmatrix} 1, & \{se\ \theta B_j(t-1) = 1 \wedge RC_j(t) < S_i(t), i\ é\ o\ índice\ para\ max\ OFTB_i(t), \\ 0, & \{caso\ contrário. \end{Bmatrix}.$$

O sinal $OFBB$ é enviado para abertura do próximo objeto quando o tamanho do item referente à $max\ OFTB_i(t)$ é maior que a capacidade residual do objeto atual.

$$OFBT_{ji}(t) = \begin{Bmatrix} 1, & se\ \ assign_{ij}(t) = 1, \\ 0, & caso\ contrário. \end{Bmatrix}.$$

Se o item é designado para o objeto, o sinal "1" é enviado para o nó do item.

3.1.4. Nó final

Função de entrada. O nó final recebe um sinal da camada de itens ($IFFT$), que representa a soma de todos os itens designados. Outro sinal é originado na camada de objetos ($IFFB$) e é utilizado para somar a quantidade de objetos utilizados para designação dos itens na iteração t.

$$IFFT(t) = \sum_{i=1}^{n} OFTF_i(t).$$

$$IFFB(t) = \sum_{j=1}^{m} OFBF_j(t),$$

Função de ativação. Existem três estados possíveis. O estado "0", que implica na existência de itens a serem designados; o estado "1", que indica que todos os itens foram designados, finalizando uma iteração; e o estado "2", que indica que o limite inferior foi atingido e a solução ótima foi encontrada.

$$\theta F(t) = \begin{Bmatrix} 0, & \{se\ IFFT(t) < n, \\ 1, & \{se\ IFFT(t) = n, \\ 2, & \{se\ IFFT(t) = n\ \wedge IFFB(t) = LB. \end{Bmatrix}.$$

Função de saída. Os três estados possíveis, 1, 2 e 3 indicam, respectivamente: existência de itens a serem designados; todos os itens foram designados, mas o LB não foi atingido; e fim do problema.

$$OFFI(t) = \begin{Bmatrix} 1, & \{se\ \theta F(t) = 0, \\ 2, & \{se\ \theta F(t) = 1\ \wedge k =\ k_{max}, \\ 3, & \{se\ (\theta F(t) = 1 \wedge k =\ k_{max})\ \vee \theta F(t) = 2. \end{Bmatrix},$$

$$OFF(k) = IFFB(t), se\ OFFI(t) = 2\ ou\ 3.$$

3.1.5. Estratégia de busca.

Os pesos são modificados a cada iteração, de modo a permitir que a rede faça uma busca por soluções melhores através da alteração na permutação dos itens. A estratégia apresentada por Agarwal [2] para ajuste de pesos tende a buscar soluções próximas à ótima local ou até mesmo a ótima local, que pode ou não ser a ótima global, partindo de uma solução inicial obtida por algum método heurístico, neste caso, *First-Fit-Descending*. Para um dado número aleatório *Rnd* ∈ [0, 1], os pesos são alterados como segue:

$$Se\ Rnd < 0.5, \omega_i(k+1) = \omega_i(k) + (\alpha * Rnd * \varepsilon * S_i),$$

$$Senão\ \ \omega_i(k+1) = \omega_i(k) - (\alpha * Rnd * \varepsilon * S_i)\ \forall\ i\ \in T,$$

onde o erro ε é dado por $OFF(k) - LB$.

Quando há uma melhora no resultado em relação à iteração anterior, um reforço no último conjunto de pesos é aplicado seguindo a regra a seguir:

$$\omega_i(k) = \omega_i(k) + RF* (\omega_i(k) - \omega_i(k-1))\ \forall\ i\ \in T.$$

Para prevenir que a rede siga um caminho que não gere soluções melhores por muitas iterações, é aplicado o mecanismo de *retorno*, que consiste em, após um número determinado de iterações sem melhoria da solução, reiniciar os pesos com o melhor vetor de pesos encontrado até o momento.

A seguir são apresentadas as condições em que a rede foi avaliada no presente artigo.

4. MEDOTOLOGIA

Neste artigo buscou-se encontrar um conjunto de parâmetros que aumentem a eficiência da RNAA, bem como avaliar os efeitos das interações entre os parâmetros. Para isso foi utilizado o método de Delineamento de Experimentos. O projeto escolhido foi do tipo **Fatorial Completo**, onde todas as combinações entre os fatores são avaliadas. Escolheu-se este tipo de projeto por ser mais preciso e devido ao baixo custo envolvido nos testes. Outros tipos de projeto de delineamento, onde são estimadas respostas para determinados níveis dos fatores, podem diminuir consideravelmente o número de testes, porém, podem ser menos precisos, exigindo mais atenção na validação e sendo mais indicados para situações onde estão envolvidos altos custos na realização dos ensaios.

Os parâmetros controláveis da RNAA são: *taxa de aprendizagem*; *fator de reforço*, *fator de retorno* e *número de iterações*. Os níveis dos fatores foram definidos a partir de alguns testes preliminares feitos a partir dos parâmetros indicados por Kasap e Agarwal [10]. Os níveis que serão combinados de cada fator são os seguintes:

- *taxa de aprendizagem (α)*: 10 níveis (0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1 e 0.5);

- *fator de reforço (RF)*: 2 níveis (2 e 3);

- *fator de retorno (BF)*: 2 níveis (500 e 1000);

- *número de iterações (k)*: 3 níveis (2000, 2500 e 3000);

A variável de resposta avaliada na análise estatística será a quantidade de problemas onde o resultado é melhorado em relação à heurística FFD. A quantidade de experimentos realizados no projeto Fatorial Completo é dada pelo produto dos níveis dos fatores e o número de repetições, ou seja, α * *RF* * *BF* * *k* * *n*, que resulta em 10*2*2*3*1 = 120 experimentos. Toda análise estatística foi feita com auxílio do *software* Minitab® 16.1.0

A RNAA, bem como a heurística FFD foram codificados em Visual Basic® 6.0. Para realização dos testes foi utilizado o conjunto de *benchmark* de dificuldade média disponível em *OR-Library* da *Technische Universitat Darmstadt* [1]. Este conjunto de dados é dividido em 48 subconjuntos com 10 instâncias em cada um deles. Este conjunto foi escolhido porque, além de ter uma quantidade significativa de problemas, menos da metade são resolvidos de forma ótima pela heurística FFD. No conjunto de dificuldade fácil, composto por 720 problemas, divididos em 36 subconjuntos com 20 problemas cada, 76% dos problemas são resolvidos de forma ótima através de FFD. O conjunto de problemas difíceis consiste em apenas 10 problemas.

5. RESULTADOS

Na tabela 1 são apresentados os resultados dos testes, onde é mostrada a quantidade de problemas onde houve melhora em relação à heurística FFD. O conjunto de problemas avaliados é composto por 480 problemas, destes, a heurística FFD atinge a resposta ótima em 236 casos. Dos 244 restantes, são quantificados os problemas onde o resultado da RNAA apresenta melhora em relação ao resultado da FFD. Na tabela 1 a quantidade de problemas melhorados está dividida de acordo com cada parâmetro. A média geral foi de 150,8 problemas melhorados, o que representa 61,8% problemas que o FFD não resolve de forma ótima. Vale destacar que, destes, em média 63 problemas foram resolvidos de forma ótima pela RNAA.

	Número iterações	**2000**				**2500**				**3000**			
	Fator de retorno	**500**		**1000**		**500**		**1000**		**500**		**1000**	
	Fator de reforço	**2**	**3**	**2**	**3**	**2**	**3**	**2**	**3**	**2**	**3**	**2**	**3**
Taxa de aprendizagem	**0,00001**	160	155	151	151	159	159	158	158	159	159	159	159
	0,00005	156	156	157	157	152	152	157	157	154	154	158	158
	0,0001	152	153	154	154	153	153	157	154	161	161	156	156
	0,0005	145	145	145	145	150	150	148	148	154	154	155	155
	0,001	149	147	146	146	150	148	150	150	152	152	153	149
	0,005	145	146	151	151	144	149	146	146	147	147	148	151
	0,01	150	145	145	145	148	150	152	152	149	149	153	151
	0,05	145	142	144	150	151	147	152	152	149	149	150	153
	0,1	148	146	145	146	147	146	151	151	152	152	148	150
	0,5	143	147	147	145	149	145	145	145	152	152	150	150

Tabela 1: Quantidade de problemas melhorados em relação à heurística FFD.

Observando os dados da tabela 1, pode-se notar que a variação no *fator de reforço* muito pouco afeta as respostas, o que pode ser comprovado pelas médias obtidas por cada *taxa de reforço*, apresentadas na tabela 2.

Fator de reforço	**2**	**3**
Problemas melhorados	150,9	150,8

Tabela 2: Quantidade média de problemas melhorados por *fator de reforço*.

1. http://www.wiwi.uni-jena.de/Entscheidung/binpp/bin2dat.htm, ultimo aceso em 04 de abril de 2013.

A maior quantidade de problemas melhorados foi de 161 (duas observações), para os parâmetros: *número de iterações*: 3000; *fator de retorno*: 500; *taxa de aprendizagem*: 0,0001; e *fator de retorno*: 2 e 3. A segunda maior quantidade, 160 problemas melhorados, foi obtida com os parâmetros: *número de iterações*: 2000; *fator de retorno*: 500; *fator de reforço*: 2; e *taxa de aprendizagem* 0,00001.

Analisando o *número de iterações*, é possível concluir que uma maior quantidade de iterações aumenta a média de problemas melhorados (tabela 3). Entretanto, um maior *número de iterações* impacta diretamente no tempo computacional, devendo manter-se em números razoáveis. Para 3000 iterações, o tempo para resolver um problema é de aproximadamente 13 segundos.

Número de iterações	**2000**	**2500**	**3000**
Problemas melhorados	148,8	150,3	153

Tabela 3: Quantidade média de problemas melhorados por *número de iterações*.

O *fator de retorno* apresenta leve tendência a funcionar melhor com valores maiores, como pode ser visto na tabela 4. Entretanto, observando dados isolados acredita-se que sua influência esteja relacionada a outros fatores.

Fator de retorno	**500**	**1000**
Problemas melhorados	150,6	151,1

Tabela 4: Quantidade média de problemas melhorados por *fator de retorno*.

A *taxa de aprendizagem* tende a melhorar os resultados à medida de seu valor diminui (tabela 5). Taxas menores tendem a causar menos perturbação na permutação dos itens, como observado em [2].

Taxa de aprendizagem	**0,00001**	**0,00005**	**0,0001**	**0,0005**	**0,001**	**0,005**	**0,01**	**0,05**	**0,1**	**0,5**
Problemas melhorados	157,3	155,7	155,3	149,5	149,3	147,6	149,1	148,7	148,5	147,5

Tabela 5: Quantidade média de problemas melhorados por *taxa de aprendizagem*.

Foram ainda analisadas as médias das seguintes interações entre os fatores:

- *número de iterações* X *fator de retorno*;
- *número de iterações* X *fator de reforço*;
- *número de iterações* X *taxa de aprendizagem*;
- *fator de retorno* X *fator de reforço*;
- *fator de retorno* X *taxa de aprendizagem*;
- *fator de reforço* X *taxa de aprendizagem*;
- *número de iterações* X *fator de retorno* X *fator de reforço*;
- *taxa de aprendizagem* X *número de iterações* X *fator de retorno*;
- *taxa de aprendizagem* X *número de iterações* X *fator de reforço*; e
- *taxa de aprendizagem* X *fator de retorno* X *fator de reforço*.

Em todas as interações analisadas, os resultados confirmaram as tendências observadas nas análises das médias isoladas, ou seja, combinações entre fatores não são capazes de produzir alterações na variável de resposta, diferentes daquelas obtidas pela alteração dos fatores isoladamente, em especial, da *taxa de aprendizagem* e do *número de iterações*.

6. CONCLUSÕES

Este trabalho descreve os passos da implementação de uma meta-heurística RNAA para resolução de Problemas de Corte e Empacotamento. Ainda, avalia a influência da variação dos parâmetros na efetividade da rede. Foi verificado que a rede se apresenta como uma alternativa viável para resolução dos PCEs. Adicionalmente, conforme relata Smith [17], os computadores digitais não são totalmente adequados para programação de RNAs, sendo que *hardwares* apropriados poderiam levar as RNAs, e consequentemente as RNAAs, a patamares bastante superiores em relação a outras meta-heurísticas.

A análise de médias feita a partir de DOE do tipo Fatorial Completo colaborou para verificar como os parâmetros podem influenciar no funcionamento da RNAA. Observou-se que *taxas de aprendizagem* menores e *número de iterações* maiores produzem melhores resultados. O *número de iterações* deve ser aumentado com cautela, pois impacta no tempo computacional necessário para resolução do problema. A média obtida para 3000 iterações e as quatro menores *taxas de aprendizagem* (0,00001, 0,00005, 0,0001 e 0,0005) foi de 157 problemas melhorados, contra a média geral de 150,8.

Verificou-se também que o *fator de reforço* pouco influencia nos resultados. Testes com outros valores de *fator de reforço* poderiam confirmar se sua utilização é realmente relevante. A variação do *fator de retorno* pouco influencia na média dos resultados, porém, observa-se que, em alguns problemas, o aumento do valor piora os resultados e vice-versa. Este fato pode indicar uma eficácia discutível do mecanismo de retorno, talvez pelo fato de reiniciar os pesos a partir da melhor solução, concentrando a busca em ótimos locais. Um mecanismo de retorno utilizando reinicialização a partir de outras heurísticas ou até mesmo de pesos aleatórios poderia incrementar o desempenho da RNAA.

Outras estratégias de aprendizagem poderiam ser exploradas, como a estratégia de *annealing schedule*, indicada por Agarwal [2], onde a *taxa de aprendizagem* inicia com valores altos e diminui ao decorrer das iterações. Novas estratégias de aprendizagem que incorporem outras informações capazes de capturar algumas particularidades de cada problema, como tamanho médio e desvio padrão dos itens, ou particularidades das soluções encontradas, por exemplo, a capacidade remanescente de cada objeto, poderiam ser testadas.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] ARAUJO, S. A., CONSTANTINO, A. A. e POLDI, K. C. (2010), An evolutionary algorithm for the one-dimensional cutting stock problem, *International Transactions in Operational Research*, 18, 115-127.

[2] AGARWAL, A. (2009), Theoretical insights into the augmented-neural-network approach for combinatorial optimization, *Annals of Operations Research*, 168, 101-117.

[3] AGARWAL, A., PIRKUL, H. e JACOB, V. S. (2003), Augmented neural networks for task scheduling, *European Journal of Operational Research*, 151, 481-502.

[4] CHEN, C. S., HART, S. M. e THAM, W. M. (1996), A simulated annealing heuristic for the one-dimensional cutting stock problem, *European Journal of Operations Research*, 93, 522-535.

[5] DEMIREL, N. Ç. e TOKSARI, M. D. (2006), Optimization of the quadratic assignment problem using an ant colony algorithm, *Applied Mathematics and Computation*, 183, 427-435.

[6] DYCKHOFF, H. (1990), A typology of cutting and packing problems, *European Journal of Operations Research*, 44, 145-159.

[7] EISEMANN, K. (1957), The trim problem, *Management Science*, 3, 279-284.

[8] FALKENAUER, E. (1996), A hybrid grouping genetic algorithm for bin packing, *Journal of Heuristics*, 2, 5-30.

[9] HOPFIELD, J. J e TANK, D. W. (1985), Neural computation of decisions in optimization problems, *Biological Cybernetics*, 52, 141-152.

[10] KASAP, N. e AGARWAL, A. (2012), Augmented neural networks and problem structure-based heuristics for the bin-packing problem, *International Journal of Systems Science*, 43, 1412-1430.

[11] LOH, K., GOLDEN, B. e WASIL, E. (2008), Solving the one-dimensional bin packing problem with a weight annealing heuristic, *Computers & Operations Research*, 35, 2283-2291.

[12] MARTELLO, S. e TOTH, P. (1990), Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, Cichester.

[13] MONTGOMERY, D. C. (2001), Design and Analysis of Experiments, 5th ed., John Wiley & Sons, New York.

[14] PAN, Q. e RUIZ, R. (2012), Local search methods for the flowshop scheduling problem with flowtime minimization, *European Journal of Operational Research*, 222, 31-43.

[15] RUIZ, R., MAROTO, C. e ALCARAZ, J. (2005), Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics, *European Journal of Operational Research*, 165, 34-54.

[16] RUIZ, R. e STUTZLE, T (2007), A simple and effective iterated greedy algorithm for the permutation flowshop problem with makespan and weighted tardiness objectives, *European Journal of Operational Research*, 187, 2033-2049.

[17] SMITH, K.A. (1999), Neural networks for combinatorial optimization: a review of more than a decade of research, *INFORMS Journal on Computing*, 11, 15-34.

[18] WOODCOCK, A. J. e WILSON, J. M. (2010), A hybrid tabu search/branch & bound approach to solving the generalized assignment problem, *European Journal of Operations Research*, 207, 566-578.