APLICAÇÃO DEUMA REDE NEURAL AUMENTADA PARA RESOLUÇÃO DE PROBLEMAS DE CORTE E EMPACOTAMENTOUTILIZANDO NOVAS ESTRATÉGIAS DE APRENDIZAGEM

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. São apresentadas quatro novas estratégias de aprendizagem alternativas àquela descrita por Agarwal (2009). 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 eficiência das funções de aprendizagem propostas.

PALAVRAS CHAVE: Rede Neural, Corte e Empacotamento, Meta-heurísticas.

ÁREA PRINCIPAL: IND, MH, OC.

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 and/or physical space of enterprises. Four new learning strategies are presented alternatively to that described by Agarwal (2009). 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 efficiency of the learning functions proposed.

KEYWORDS: Neural Network, Cutting and Packing, Meta-heuristics.

MAIN AREA:IND, MH, OC.

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 mesmoótimas. Apesar de ceticismo ocasionado pela falta de *insights* teóricos (Ruiz e Stutzle, 2007), a literatura é farta em evidências empíricas que demonstram sua efetividade.

Falkenauer (1996) utiliza um algoritmo de grupamento genético híbrido nos problemas de *Bin Packing*.Loh*et al.*(2008), apresentam o algoritmo *WeightAnnealing* para resolução do problema de*Bin Packing*. Woodcock e Wilson (2010) testam um método híbrido de Busca Tabu e *Branch&Bound* para problemas de designação.Araujo *et al.*(2010), aplicam um algoritmo evolucionário na resolução de um problema de corte de estoque com objetos variados e limitados.Chen *et al.*(1996) aplicam a meta-heurística *SimulatedAnnealing* 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.

Este trabalho foi estruturado da seguinte forma: no próximo capítulo o contexto do Problema de Corte e Empacotamento (PCE) será apresentado, a seguir, no capítulo 3, será feita uma breve revisão literária sobre a Rede Neural Artificial Aumentada (RNAA), seguida pelo detalhamento do seu funcionamento. No capítulo 4 será mostrada a metodologia utilizada nos testes. O capítulo 5 mostra os resultados e uma breve análise destes. Por fim, no capítulo 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 *TrimProblem* apresentado por Eisemann (1957), até alocação de memória, passando por *Bin Packing*, problema da mochila, carregamento de veículos, particionamento, entre outros. ADyckhoff (1990) é atribuído o primeiro trabalho com objetivo de integrar os vários problemas e tipologias encontradas no âmago dos PCEs.

De acordo com Dyckhoff (1990), os dois principais grupos de dados do Problema de Corte e Empacotamento são os **estoques** de **objetos** grandes e as **ordens** de **itens** pequenos. Itens são combinados formando **padrões de corte**, que serão designados aos objetos. 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 Artificial Aumentada

Rede NeuralArtificial (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 OtimizaçãoCombinatória (Smith, 1999). Os resultados

obtidos através desta técnica, porém, em termos de qualidade de solução, têm sido comparáveis aos de outras meta-heurísticas. Ainda, segundo Smith (1999), a primeira demonstração de resolução de um Problema de OtimizaçãoCombinatória utilizando RNA foi feita por Hopfield e Tank (1985), em um Problema de Caixeiro Viajante (PCV). Este trabalho serviu de inspiração para a Rede Neural Aumentada, proposta por Agarwal*et al.* (2003).

A primeira Rede Neural Artificial Aumentada (RNAA) foi utilizada como alternativa para resolução de um problema de agendamento de tarefas(Agarwal*et al.*,2003). 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. Kasap e Agarwal (2012)propõem uma RNAA para resolução de um PCE, que é estruturada como mostra a figura 1.

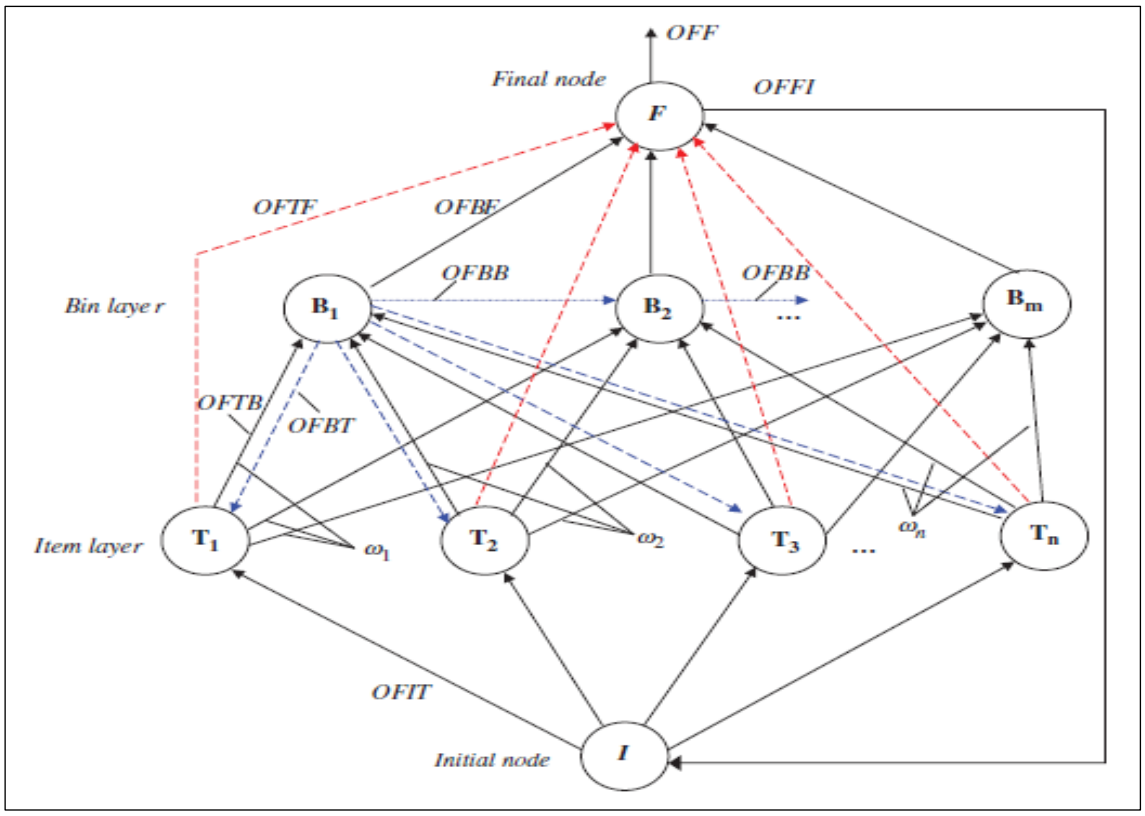


Figura 1: Estrutura da Rede Neural Aumentada para Problema de Corte e Empacotamento(Kasap e Agarwal, 2012).

A seguir, a notação utilizada na 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, em ordem crescente de índice, no primeiro objeto aberto em que a capacidade residual (RC_j) seja suficiente. Caso o tamanho do item a ser designado seja maior que a capacidade residual dos objetos abertos, um novo objeto é aberto (Martello e Toth, 1990).Outrasheurísticasquepoderiamserutilizadassão: *Next-Fit* (*NF*), *First-Fit* (*FF*), *Best-Fit* (*BF*), *Next-Fit-Descending* (*NFD*) e *Best-Fit-Descending* (*BFD*).

3.1. Funcionamento da RNAA.

Inicialmente os pesos $\omega_i(0)$ são ajustados 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. *LB* pode ou não ser a solução ótima.*UB*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, a função de entrada recebe sinal "1" para inicializar a primeira designação da primeira iteração. Após, 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".$OFTF$enviaum 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 a RC_jnão é suficiente para designar o item com máximo $OFTB_i(t)$. Quando RC_j é inferior ao menor item não designado, o objeto é fechado.

$$RC_i(1) = C,$$

$$\theta B_1(1) = 1, \left(o\ estado\ do\ primeiro\ objeto\ na\ primeira\ iteração\ é\ 1\ (aberto)\right),$$

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

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

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

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

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

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

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

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*.

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

Funções de saída.

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

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

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

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

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

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*.

$$IFFT(t) = \sum_{i=1}^{n} OFTF_i(t). \qquad IFFB(t) = \sum_{j=1}^{m} OFBF_j(t),$$

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.

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

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.

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

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

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 em Agarwal (2009)para ajuste de pesos tem como objetivo buscar soluções próximas à ótima local, ou até mesmo a ótima local, que pode ou não ser a ótima global. Para isso, parte 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 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*,onde, após um determinado número de iterações semmelhoria da solução, os pesos são reiniciados com o melhor vetor de pesosencontrado até o momento.

A função de atualização de pesos depende de cinco informações: o peso anterior ($\omega_i(k-1)$), α, Rnd, ε e S_i. Destas, Rnd e S_i são responsáveis pela perturbação na permutação dos itens e α e ε tem a função de amplificar ou reduzir esta perturbação. Enquanto α é um parâmetro manipulado pelo usuário, εincorpora uma característica da solução. Conforme observado por Kasap e Agarwal (2012),εtem como objetivo alterar, em maior ou menor grau, a permutação dos itens nos objetos, quanto maior o valor deε, maior perturbação na permutação dos itens.

Sabe-se que nem sempre o *LB*caracteriza a solução ótima,desta forma, quanto maior for a diferença entre a solução ótima e *LB*, mais amplificada será a atualização de pesos, podendo prejudicar a busca por ótimos locais nos problemas onde esta diferença for grande. Testes preliminares mostraram que a rede tem dificuldade para melhorar a solução nestas situações. Uma diminuição no valor de α pode compensar uma maior diferença entre a solução ótima e *LB*, entretanto, em situações práticas não se conhece a solução ótima previamente.

NasRNAAs construídas sob a estrutura da heurística *FFD*, observa-se queos primeiros objetos utilizados tendem a valores bastante baixos de RC, caracterizando objetos bem utilizados. Sabe-se ainda, que a melhor solução possível para qualquer problema de corte é aquela em que os itens são arranjados nos objetos de forma que a RCde todos os objetos utilizados seja zero. Partindo dessa observação e almejando uma atualização mais eficiente dos pesos, buscou-se uma forma de incorporar informações mais específicas sobre a solução, neste caso, a própria RC.

Dessa forma, são propostas neste trabalho, quatro estratégias de aprendizagem que incorporam RC, com objetivo de alterar com maior grau os pesos dos itens que estiverem designados a objetos com maiores valores de RC.

Na primeira estratégia foram eliminadas da função de atualização de pesos, as variáveis ε e S_i, sendo incluída a RC_jdo objeto onde o item *i* está designado, segundo a função a seguir:

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

$$Senão\ \ \omega_i(k+1) = \omega_i(k) - (\alpha * Rnd * RC_j)\ \forall\ i \in T, j \in B\ e\ OFTB_{ji}(t) = 1.$$

Asegunda estratégia consiste emsubstituir ε por RC_j:

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

$$Senão\ \ \omega_i(k+1) = \omega_i(k) - (\alpha * Rnd * S_i * RC_j)\ \forall\ i \in T, j \in B\ e\ OFTB_{ji}(t) = 1.$$

Uma terceira alternativa é considerar ε e RC_j, descartando S_i:

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

$$Senão\ \ \omega_i(k+1) = \omega_i(k) - (\alpha * Rnd * \varepsilon * RC_j)\ \forall\ i \in T, j \in B\ e\ OFTB_{ji}(t) = 1.$$

Na quarta estratégia é incluída a RC_j, conforme indicado a seguir:

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

$$Senão\ \ \omega_i(k+1) = \omega_i(k) - (\alpha * Rnd * \varepsilon * S_i * RC_j)\ \forall\ i \in T, j \in B\ e\ OFTB_{ji}(t) = 1.$$

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

4. Metodologia

Neste artigo são comparadas quatro novas funções para atualização de pesos e os resultados são confrontados entre si, e com a função de atualização original,apresentada em Agarwal (2009). Os parâmetros controláveis da RNAA são: α; *RF*, *BF* e *k*.Testes preliminares, realizados em uma rede utilizando a função de atualização de pesos original, mostraram que a rede tende a ser mais eficiente quando se aumenta o valor de *k* e/ou se diminui o valor de α. Nestes testes *k* foi avaliado com valores entre 2000 e 3000 iterações e α foi avaliada com valores entre 0,00001 e 0,5. Para *BF*, foram utilizados os valores 500 e 1000, sem tendência definida, sendo o desempenho médio de 1000 ligeiramente melhor. *RF* foi testado com os valores 2 e 3, não apresentando diferença significativa entre eles.

Nos testes apresentados neste artigo são utilizados oito valores de α: 0,0000001, 0,0000005, 0,000001, 0,000005, 0,00001, 0,00005, 0,0001 e 0,0005. *k*foi fixado em 3000. Valores maiores tendem a produzir melhores resultados, entretanto prejudicam o tempo computacional para resolução do problema. *BF* e o *RF* foram ajustados em 1000 e 2, respectivamente.

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 *TechnischeUniversitat 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 (236) 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 por *FFD*, limitando a análise. O conjunto de problemas difíceis consiste em apenas 10 problemas.

5. Resultados

Na tabela 1 são apresentados os resultados dos testes para a estratégia original e mostra, dos 244 problemas não resolvidos de forma ótima pela heurística *FFD*, quantos problemas tiveram a

solução melhorada e quantos foram resolvidos de forma ótima. O melhor resultado foi obtido para αigual a 0,00001 (163 problemas melhorados) e 0,00005 (162 problemas melhorados), em linha com os resultados obtidos para α igual a 0,00005, no trabalho de Kasap e Agarwal (2012). Testes preliminares indicaram que menores valores de α melhoram os resultados, porém, como pode ser observado no gráfico 1, taxas muito baixas produzem resultados insatisfatórios.

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

	Estratégia original w = w +- (α*Rnd*ε*S)		
	Ótimo	**Melhor**	**Total**
0,0000001	1	18	19
0,0000005	6	54	60
0,000001	11	72	83
0,000005	51	91	142
0,00001	71	**92**	**163**
0,00005	**72**	90	162
0,0001	68	90	158
0,0005	65	91	156

Tabela 1: Resultados para função de atualização de pesos original.

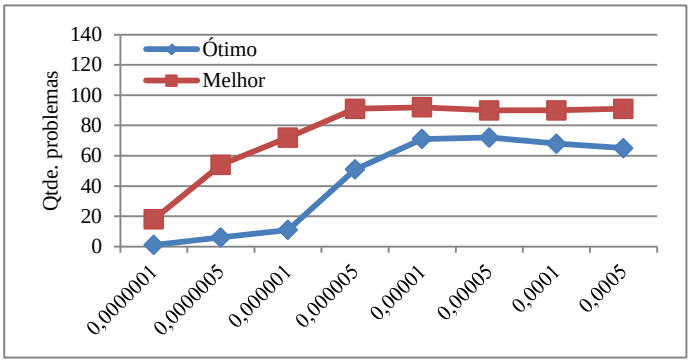


Gráfico 1: Resultados para função de atualização de pesos original.

Na tabela 2 e gráfico 2, são mostrados os resultados para a estratégia 1. Os melhores resultados foram obtidos com valores de α ajustados em 0,00005 e 0,0001, chegando a um total de 215 problemas melhorados, sendo que, destes, 114 foram resolvidos de forma ótima, resultados superiores àqueles obtidos com a função de aprendizagem original.

	Estratégia 1 w = w +- (α*Rnd*RC)		
	Ótimo	**Melhor**	**Total**
0,0000001	1	13	14
0,0000005	8	53	61
0,000001	21	76	97
0,000005	74	95	169
0,00001	91	100	191
0,00005	**114**	101	**215**
0,0001	**114**	100	214
0,0005	105	**104**	209

Tabela 2: Resultados para função de atualização de pesos1.

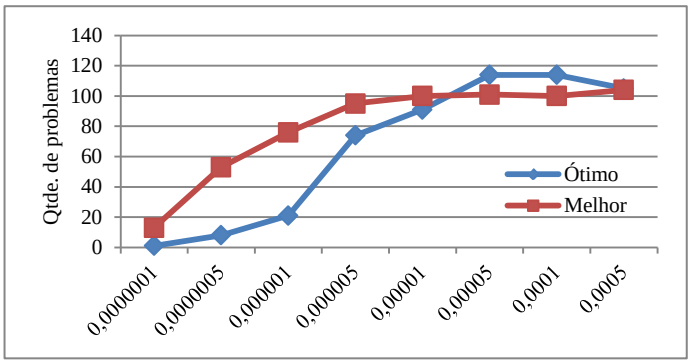
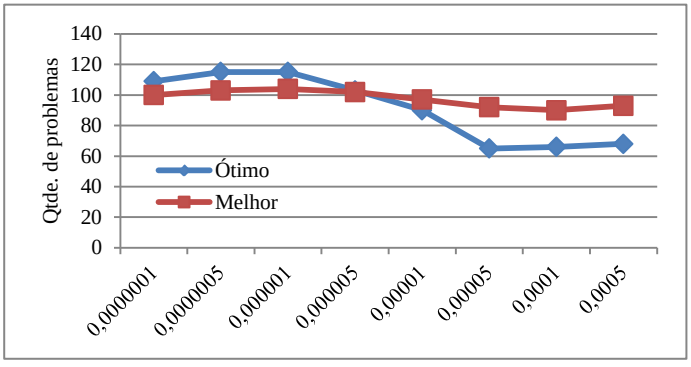


Gráfico 2: Resultados para função de atualização de pesos 1.

Na tabela 3, os resultados para a função de atualização3 são mostrados. Com esta função foi possível atingir um total de 219 problemas melhorados, sendo 115 resolvidos de forma ótima com α ajustado em 0,000001. O segundo melhor resultado, 218 problemas melhorados, foi conseguido com α igual a 0,0000005. As taxas são menores que as mais eficientes da estratégia 1, compensando uma maior perturbação da solução causada pela inclusão deS_i.O gráfico 3 mostra que os resultados pioram à medida que aumenta o valor de α.

	Estratégia 2 w = w +- (α*Rnd*S*RC)		
	Ótimo	**Melhor**	**Total**
0,0000001	109	100	209
0,0000005	**115**	103	218
0,000001	**115**	**104**	**219**



0,000005	103	102	205
0,00001	90	97	187
0,00005	65	92	157
0,0001	66	90	156
0,0005	68	93	161

Tabela 3: Resultados para função de atualização de pesos2.

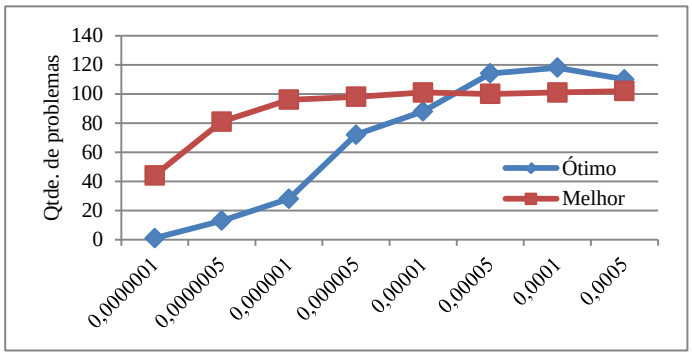


Gráfico 3: Resultados para função de atualização de pesos 2.

A tabela 4 mostra os resultados para estratégia 3. O total de problemas melhorados foi igual ao obtido pela estratégia 2, porém, 118 foram resolvidos de forma ótima, contra 115 da estratégia anterior. Maiores valores de α se mostram mais eficientes, como pode ser observado no gráfico 4.

	Estratégia 3		
	w = w +- (α*Rnd*ε*RC)		
	Ótimo	**Melhor**	**Total**
0,0000001	1	44	45
0,0000005	13	81	94
0,000001	28	96	124
0,000005	72	98	170
0,00001	88	101	189
0,00005	114	100	214
0,0001	**118**	101	**219**
0,0005	110	**102**	212

Tabela 4: Resultados para função de atualização de pesos3.

Gráfico 4: Resultados para função de atualização de pesos 3.

Na tabela e gráfico 5, os resultados para estratégia 4. Esta estratégia apresentou os melhores resultados em relação às demais estratégias, atingindo o total de 119 problemas resolvidos de forma ótima,de um total de 220 problemas melhorados, contra 219 para as estratégias 2 e 3, 215 para estratégia 1 e 163 para a estratégia original. Nesta estratégia, valores de α menores são mais efetivos, sendo 0,0000005 o valor que resultou em respostas melhores.

	Estratégia 4		
	w = w +- (α*Rnd*ε*S*RC)		
	Ótimo	**Melhor**	**Total**
0,0000001	110	101	211
0,0000005	**119**	101	**220**
0,000001	113	**104**	217
0,000005	92	101	193
0,00001	87	95	182
0,00005	69	88	157
0,0001	66	89	155
0,0005	62	89	151

Tabela 5: Resultados para função de atualização de pesos4.

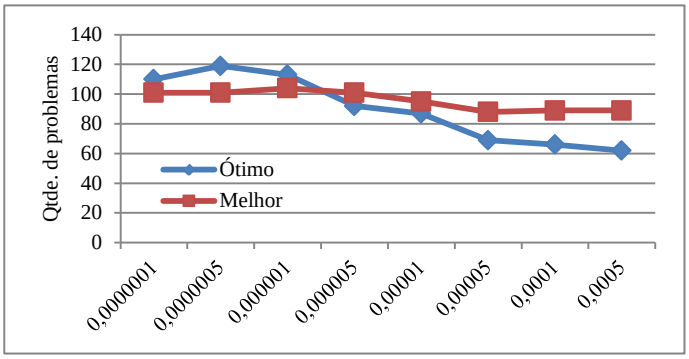


Gráfico 5: Resultados para função de atualização de pesos 4.

No gráfico 6, é mostrada a quantidade total de problemas resolvidos de forma ótima para cada valor de α analisado. A estratégia original utilizada neste trabalho resultou em 308 problemas resolvidos de forma ótima, o que representa 64,2% do total de 480 problemas analisados, em linha com as 312 respostas ótimas obtidas por Kasap e Agarwal (2012). A estratégia 4 resultou em um total de 355 respostas ótimas (73,9%), seguido pela estratégia 3, com 354 (73,5%), estratégia 2 com 351 (73,1%) e estratégia 1 com 350 (72,9%). Os dados são compilados na tabela 6.

	Estratégia 1	**Estratégia 2**	**Estratégia 3**	**Estratégia 4**	**Estratégia original**
0,0000001	237	345	237	346	237
0,0000005	244	**351**	249	**355**	242

0,000001	257	**351**	264	349	247
0,000005	310	339	308	328	287
0,00001	327	326	324	323	307
0,00005	**350**	301	350	305	**308**
0,0001	**350**	302	**354**	302	304
0,0005	341	304	346	298	301

Tabela 6: Total de problemas de resolvidos de forma ótima pela RNAA.

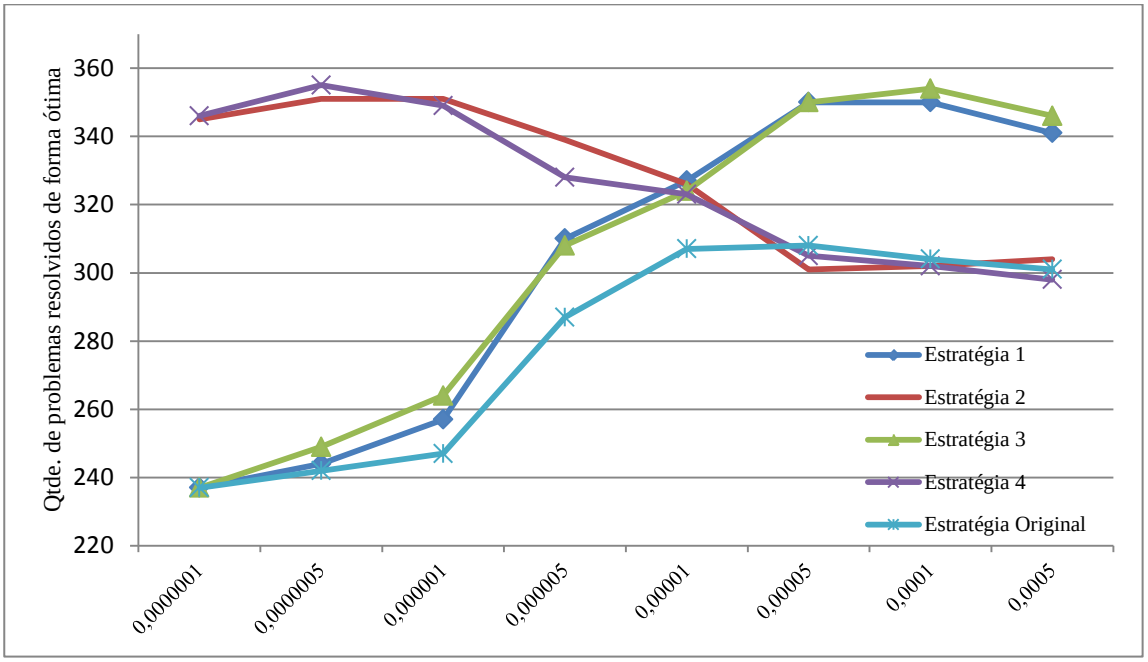


Gráfico 6: Total de problemas resolvidos de forma ótima pela RNAA para todas as estratégias.

6. Conclusões

Este trabalho descreve os passos da implementação dameta-heurística RNAA para resolução de Problemas de Corte e Empacotamento.Foi verificado que a rede se apresenta como uma alternativa viável para resolução dos PCEs, visto que, conforme relata Smith (1999), 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.

As quatro estratégias de aprendizagempropostas neste trabalho, alternativas àquela apresentada em Kasap e Agarwal (2012), e que incorporam a capacidade residual do objeto à função de atualização de pesos e priorizam perturbações mais localizadas na solução, preservando as regiões “boas” da solução, aumentaram a eficiência da função de atualização de pesos em 14,2%, em média. Todas as estratégias propostas tiveram resultados bastante superiores à estratégia original, com destaque a função de atualização de pesos4, que resolveu de forma ótima, 73,9% dos problemas analisados, contra 64,3% resultados ótimos obtidos pela função de atualização original e 49,2% obtidos pela heurística *FFD*.

Os valores de α onde se conseguiu omelhor desempenho de cada estratégia, ficaram próximos aos extremos analisados, com valores maiores para as estratégias 1 e 3, e valores menores para as estratégias 2 e 4. Isto comprova a suposição de que α apenas influencia na amplitude de permutação dos itens, sendo que o valor de αque melhor se ajusta, depende da estratégia utilizada. Estratégias com α variável, como por exemplo, a estratégia de *annealingschedule*, indicada por Agarwal (2009),onde αinicia com valores altos e diminui ao decorrer das iterações, poderiam ser testadas.

Outras estratégias considerando alterações na reinicialização de pesos poderiam ser exploradas. Ao atingir o número determinado de iterações, a rede reinicia os pesos utilizando aqueles que resultaram na melhor resposta. Com estratégias de atualização de pesos que exploram de forma mais eficiente soluções locais, como as propostas neste trabalho, uma reinicialização de pesos que direcione a outras regiões da solução em busca de novos ótimos locais, aumentaria a chance de se obter um ótimo global. Isto pode ser feito com uma reinicialização de pesos aleatória, ou ainda, partindo de outras heurísticas, como aquelas indicadas anteriormente neste trabalho.

Referências

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.

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

Agarwal, A.,Pirkul, H. e Jacob, V. S. (2003), Augmented neural networks for task scheduling, *European Journal of Operational Research*, 151, 481-502.

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.

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

Eisemann, K. (1957), The trim problem, *Management Science*, 3, 279-284.

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

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

Kasap, N. eAgarwal, A. (2012), Augmented neural networks and problem structure-based heuristics for the bin-packing problem, *International Journal of Systems Science*, 43, 1412-1430.

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.

Martello, S. e Toth, P. (1990), Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, Cichester.

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.

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.

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.