Gest. Prod., São Carlos, v. 22, n. 3, p. 611-623, 2015
http://dx.doi.org/10.1590/0104-530X1258-14

Resolução de problemas de *Bin Packing* utilizando rede neural aumentada e *Minimum Bin Slack*

Solving Bin Packing problems with augmented neural networks and Minimum Bin Slack

Ricardo de Almeida[1]
Maria Teresinha Arns Steiner[1]

Resumo: O objetivo do presente trabalho é comparar a utilização da meta-heurística Rede Neural Artificial Aumentada (RNAA) com a heurística *Minimum Bin Slack* (MBS) para resolução de Problemas de Otimização Combinatória, mais especificamente, em problemas de *Bin Packing*, uma classe de 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 a otimização dos parâmetros da RNAA, fez-se uso de um projeto de Delineamento de Experimento (DOE) do tipo Fatorial Completo. Os testes, realizados em diversos problemas *benchmark* da literatura, mostraram que, de uma forma geral, a heurística MBS foi superior tanto em relação à qualidade das respostas (cerca de 70% melhor), quanto em relação ao tempo computacional (aproximadamente 90% menor).

Palavras-chave: Rede neural; Corte e empacotamento; Delineamento de experimentos; *Bin Packing*; *Minimum Bin Slack*.

Abstract: *The objective of this work is to compare the Augmented Neural Network (AugNN) metaheuristic with Minimum Bin Slack (MBS) heuristic to solve Combinatorial Optimization Problems, specifically, in this case, the bin packing problem, a class of 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 optimize the parameters of AugNN, a Full Factorial Design of Experiment (DOE) was applied. The tests, developed in many benchmark problems found in the literature, showed that MBS heuristic was generally superior, both in terms of quality of solution (approximately 70% better) and computational time (about 90% shorter).*

Keywords: *Neural network; Cutting and packing; Design of experiments; Bin Packing; Minimum Bin Slack.*

1 Introdução

O processo diário de tomada de decisão nas empresas frequentemente envolve o objetivo de maximizar ou minimizar alguma função, respeitando um determinado conjunto de restrições. Como produzir de forma mais eficiente respeitando as capacidades dos recursos humanos, máquinas e insumos disponíveis num dado momento? Qual a melhor rota a seguir para entregar produtos minimizando a distância percorrida e evitando atrasos ao cliente? Qual a forma de acomodar produtos em um contêiner de modo a maximizar o espaço utilizado, respeitando as dimensões do contêiner? Qual a melhor forma de distribuir tarefas entre recursos considerando suas capacidades? Quando o domínio da função objetivo modelada para estes problemas é finito, envolvendo numerosas possibilidades de combinações, temos então Problemas de Otimização Combinatória (POC). Muitos destes problemas são classificados como NP-difíceis, dado que, conforme cresce o número de restrições, como ocorre em situações práticas, a resolução por meio de métodos exatos se torna inviável e até intratável. Desta forma, diversos métodos heurísticos e meta-heurísticos têm sido criados, adaptados e aperfeiçoados de forma a se conseguir um tratamento adequado dos problemas encontrados na prática das indústrias, auxiliando-as a aumentar sua eficiência e consequentemente sua lucratividade e competitividade.

Uma categoria relevante de POCs são os Problemas de Corte e Empacotamento (PCE). Tanto os problemas de corte como os de empacotamento compartilham a mesma estrutura e consistem em, dado um estoque de objetos grandes, designar a estes objetos itens pequenos, formando subconjuntos. É necessário que sejam respeitadas condições geométricas, ou

[1] Programa de Pós Graduação em Engenharia de Produção e Sistemas, Pontifícia Universidade Católica do Paraná – PUCPR, Rua Imaculada Conceição, 1155, Prado Velho, CEP 80215-901, Curitiba, PR, Brasil, e-mail: r_almeida80@hotmail.com; maria.steiner@pucpr.br

Recebido em Fev. 10, 2014 - Aceito em Nov. 20, 2014

Suporte financeiro: Nenhum.

seja, que o somatório das dimensões consideradas dos itens designados a um subconjunto não exceda a dimensão do objeto e que não haja sobreposição de itens, além disso, uma dada função objetivo deve ser otimizada.

Vários métodos meta-heurísticos têm sido propostos para resolução de PCEs. Exemplos podem ser encontrados em Woodcock & Wilson (2010), que testam um método híbrido de Busca Tabu e *Branch & Bound* para problemas de designação. O problema de corte de estoque com objetos variados e limitados é tratado por Araújo et al. (2011), com a aplicação de um algoritmo evolucionário. Falkenauer (1996) utiliza um algoritmo de grupamento genético híbrido e Loh et al. (2008), um algoritmo de *Weight Annealing* para problemas de *Bin-Packing*. Ainda para problemas de *Bin-Packing*, métodos baseados em Redes Neurais Artificiais são utilizados por Kasap & Agarwal (2012) e Almeida & Steiner (2013a, b).

Dentro da categoria de PCEs, o problema do tipo *Bin-Packing* é uma variação que tem como característica uma forte heterogeneidade de itens. Estes itens devem ser designados a objetos que podem ser idênticos, fracamente heterogêneos ou fortemente heterogêneos, caracterizando os problemas do tipo *Single Bin Size Bin Packing Problem* (SBSBPP), *Multiple Bin Size Bin Packing Problem* (MBSBPP) e *Residual Bin Packing Problem* (RBPP), respectivamente, de acordo com a tipologia proposta por Wäscher et al. (2007).

Depois de ajustar e analisar os parâmetros da Rede Neural Artificial Aumentada (RNAA) de Kasap & Agarwal (2012), utilizando delineamento de experimentos (Almeida & Steiner, 2013a), Almeida & Steiner (2013b) propõem alterações na função de aprendizagem, resultando em melhora de aproximadamente 15% na qualidade de solução em relação à RNAA original, e melhora de 50,4% em relação às heurísticas *best-fit decreasing* (BFD) e *first-fit decreasing* (FFD), apresentadas por Johnson et al. (1974). Scholl et al. (1997), analisaram o pior caso de desempenho das heurísticas BFD e FFD e mostraram que estas heurísticas precisam de no máximo 22,2% mais objetos que a solução ótima. Também se observa que a eficiência destas heurísticas aumenta conforme a relação item/objeto aumenta. A complexidade computacional das heurísticas FFD e BFD é O(n.log.n). Como estas heurísticas requerem o arranjo dos itens em ordem decrescente, depois de feito este arranjo a complexidade diminui para O(n). A complexidade computacional da RNAA é a mesma da heurística (O(n.log.n)), multiplicada pela quantidade de iterações, lembrando que a RNAA possui mecanismos de parada que podem limitar a quantidade de iterações.

Neste contexto, o objetivo deste trabalho é comparar o desempenho da RNAA aprimorada por Almeida & Steiner (2013a, b) a um método que produza melhores resultados que as heurísticas BFD e FFD. Um método comprovadamente superior às heurísticas BFD e FFD em termos de qualidade de solução é a heurística *Minimum Bin Slack* (MBS), proposta por Gupta & Ho (1999), a qual será analisada neste trabalho e comparada com a RNAA, sendo esta comparação a contribuição original deste trabalho. Partindo dos resultados obtidos em Almeida & Steiner (2013b), um novo delineamento de experimento (DOE) é realizado na busca de parâmetros ótimos para a RNAA. Exemplos de ajuste de meta-heurísticas por meio de DOE podem ser encontrados em Ruiz et al. (2005), que aplicam DOE para ajuste dos parâmetros de dois Algoritmos Genéticos; em Demirel & Toksari (2006), que utilizam um experimento do tipo 2k para ajuste de um algoritmo de Colônia de Formigas; e em Pan & Ruiz (2012), que utilizam DOE para calibração de um algoritmo de busca local para resolver o problema de Programação de Fluxo de Produção.

O restante do artigo está organizado da seguinte forma: na seção 2, a heurística MBS é explorada; na seção 3 é descrita a meta-heurística RNAA; e na seção 4 é apresentada a metodologia utilizada. Na seção 5, os resultados dos testes realizados são analisados; e, por fim, na seção 6, serão discutidas as conclusões.

2 *Minimum Bin Slack* (MBS)

A heurística MBS foi apresentada por Gupta & Ho (1999), como alternativa para resolução de problemas do tipo *Bin Packing* unidimensional. Este algoritmo é orientado ao objeto/*bin* (*bin-focus*), ou seja, procura de forma lexicográfica um conjunto de itens que melhor se ajuste à capacidade do objeto. Primeiramente uma lista de itens não designados, classificados em ordem decrescente, é criada. A designação dos itens a cada objeto é determinada por um procedimento de busca que testa todos os subconjuntos possíveis, iniciando pelos itens de maior tamanho. Quando o processo de busca encontra um subconjunto que preencha o objeto de forma completa, a busca é interrompida e o preenchimento de um novo objeto é iniciado. A notação utilizada é apresentada a seguir:

n = quantidade de itens;

n' = quantidade de itens atualizada;

i = número do item;

t_i = tamanho do item i;

t^{mim} = tamanho do menor item;

Z = lista de itens;

r,q = índice da lista de itens Z;

Zr' = r-ésimo item da lista Z;

A = conjunto de itens;

s (A) = espaço remanescente em um objeto devido à designação dos itens contidos em A.

No Quadro 1, a seguir, tem-se a implementação recursiva do algoritmo MBS.

Gupta & Ho (1999) demonstram que a heurística MBS é superior às heurísticas BFD e FFD em termos de qualidade de solução com pouco acréscimo de tempo computacional, sendo que as instâncias testadas possuíam não mais que 70 itens. A complexidade computacional deste procedimento é O (2^n) e testes preliminares mostraram que, em instâncias com 200 itens, o tempo de processamento pode ser superior a 13 minutos. Este fato poderia inviabilizar a utilização do MBS em instâncias com muitos itens. Entretanto, Fleszar & Hindi (2002) propõem modificações no algoritmo com o objetivo de reduzir o tempo computacional, eliminando cálculos redundantes, sem influenciar no resultado final. Com as modificações propostas, testes preliminares mostraram que problemas de 1000 itens são resolvidos em cerca de 2 segundos, tornando o algoritmo bastante eficiente em relação ao tempo de processamento. As modificações propostas por Fleszar & Hindi (2002) são as seguintes:

1. No *loop for*, se $r > q$ e o tamanho dos itens Z'_{r-1} e Z'_r é igual, o processamento do item Z'_r é ignorado, considerando que o item Z'_r não irá gerar empacotamento melhor que aqueles gerados com Z'_{r-1};
2. Fazendo T_r a soma dos tamanhos dos itens $Z'_r ... Z'_n$, se no *loop for*, $s(A) - Tr \geq s(A^*)$, então o procedimento é interrompido, visto que não é possível construir um empacotamento melhor que $s(A^*)$ com os itens remanescentes. Esta restrição é particularmente útil quando o tamanho do objeto é muito grande em relação aos tamanhos dos itens;
3. No início do procedimento, se $s(A) < t^{min}$, nenhum item pode ser adicionado ao empacotamento A e o *loop for* é então ignorado.

3 Rede neural aumentada

A Rede Neural Artificial (RNA) é uma técnica poderosa para solução de problemas de previsão, classificação e reconhecimento de padrões, entretanto não tem alcançado o mesmo sucesso na resolução de POCs (Smith, 1999). Os resultados obtidos por meio 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 POC utilizando RNA foi feita por Hopfield & Tank (1985), em um Problema de Caixeiro Viajante (PCV). Este trabalho serviu de inspiração para a RNAA proposta por Agarwal et al. (2003).

A primeira 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, em que funções de entrada são transformadas em funções de saída por meio 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 na 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 & Agarwal (2012) propõem uma RNAA para resolução de um PCE, fazendo uso de uma estrutura como a mostrada na Figura 1.

Quadro 1. Procedimento MBS (Adaptado de Fleszar & Hindi, 2002).

```
rotina MBSOnePackingSearch(q)
início
    para r := q até n' faça
    início
        i: = Z'r ;
        se t_i ≤ s(A) então
        início
            A:= A U {i}
            MBSOnePackingSearch(r + 1);
            A:= A \ {i};
            se s(A*) = 0 então Saia;
        fim;
    fim;
    se s(A) < s(A*) então A*:= A;
fim;
```

A notação utilizada para a RNAA da Figura 1, com seus respectivos significados é a seguinte:

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;

$alpha$ = 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, $j \in B$

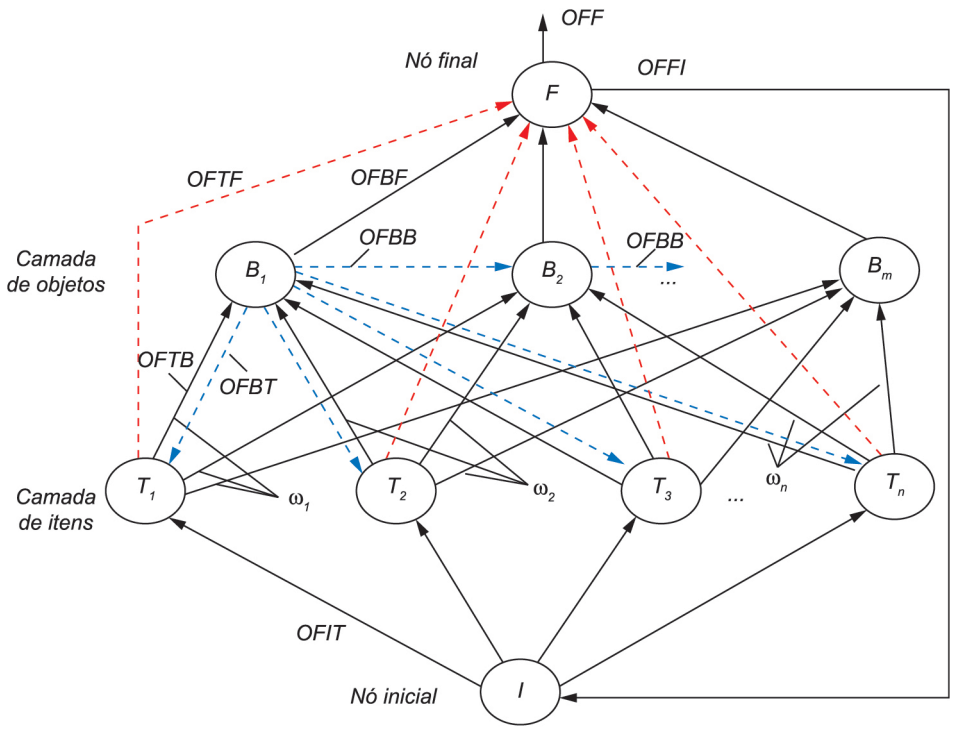

Figura 1. Estrutura da RNAA para PCEs. Kasap & Agarwal (2012).

$OFBT_{ji}(t)$ = 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;

$\theta T(t)$ = 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 a FFD. Este é um algoritmo bastante eficiente que consiste em, após organizar os itens em ordem decrescente de comprimento, 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 & Toth, 1990). 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 são inicializados em 1,00. Também são calculados os limites inferior *LB* (Equação 1) e superior *UB* (Equação 2), ou seja, a quantidade mínima e máxima de objetos necessários para colocação dos itens, respectivamente. O limite inferior pode ou não ser a solução ótima. O limite superior será usado para definir a quantidade de nós na camada dos objetos.

$$LB = (\sum_{i \in T} S_i) / C \tag{1}$$

$$UB = n / (C / \max_i(S_i)) \tag{2}$$

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" (Equação 3) para inicializar a primeira designação da primeira iteração. A seguir, quando $t > 0$, a função de entrada recebe sinal do nó final, conforme Equação 4.

$$IFI(0) = 1, \tag{3}$$

$$IFI(t) = OFFI(t), para\, t > 0. \tag{4}$$

Função de ativação. O estado do nó inicial é definido por t e k, que são inicializados em "1" conforme Equação 5.

$$\theta I(0) : \{t = 1,\ k = 1\}. \tag{5}$$

Para t > 0, t e k são atualizados de acordo com o sinal da função de entrada, $IFI(t)$, conforme Equação 6.

$$\theta I(t) = \begin{Bmatrix} t = t + 1\ e\ k = k, \{se\ IFI(t) = 1, \\ t = 1\ e\ k = k + 1, \{se\ IFI(t) = 2, \\ t = 0\ e\ k = 0, \{se\ IFI(t) = 3. \end{Bmatrix}, \tag{6}$$

em que $IFI = 1$ 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, conforme apresentado em Equação 7.

$$OFI(t) = \begin{Bmatrix} 1, \{se\ t > 0, \\ 0, \{caso\ contrário. \end{Bmatrix}. \tag{7}$$

3.1.2 Camada de itens

Função de entrada. Dada pela Equação 8, recebe sinal da função de saída do nó inicial (Equação 7).

$$IFT_i(t) = OFI(t),\ i \in T. \tag{8}$$

Função de ativação. O estado "1" indica que o "nó" correspondente ao 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", como em Equação 9, em seguida, o estado da função é dado pela Equação 10.

$$\begin{gathered} \forall\ i \in T, j \in B, \\ \theta T_i(0) = 1, \end{gathered} \tag{9}$$

$$\theta T_i(t) = \begin{Bmatrix} 0, \begin{Bmatrix} se\ \theta T_i(t-1) = 0 \vee \begin{pmatrix} \theta T_i(t-1) = \\ 1 \wedge\ OFBT_{ji}(t) = 1 \end{pmatrix}, \end{Bmatrix} \\ 1, \begin{Bmatrix} se\ \theta T_i(t-1) = 1 \vee \begin{pmatrix} \theta T_i(t-1) = \\ 0 \wedge\ IFI_i(t) = 2 \end{pmatrix}. \end{Bmatrix} \end{Bmatrix}, t > 0 \tag{10}$$

Função de saída. O sinal *OFTB* envia um sinal, com o comprimento do item multiplicado pelo peso (Equação 11), para a camada de objetos, em que 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 (Equação 12), informando se o item está ("1") ou não ("0") designado.

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

$$OFTB_i(t) = \theta T_i(t) * S_i * \omega_l(k), \tag{11}$$

$$OFTF_i(t) = \begin{Bmatrix} 1, \{se\ \theta T_i(t) = 0, \\ 0, \{caso\ contrário. \end{Bmatrix}. \tag{12}$$

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* (Equação 14) e os demais estão *não abertos* (Equação 15), sendo que a capacidade residual do primeiro objeto é igual a sua capacidade total (Equação 13). Um novo objeto é aberto quando recebe sinal do objeto anterior ($OFBB_{j-1}(t)$). Este sinal é enviado quando nenhum dos objetos abertos possui capacidade residual suficiente para designar o item com máximo *OFTB* (*t*). Quando a capacidade residual é inferior ao menor item não designado, o objeto é fechado. A capacidade residual é atualizada de acordo com a Equação 17.

$$RC_i(1) = C, \tag{13}$$

$$\theta B_1(1) = 1, \begin{pmatrix} o\ estado\ do\ primeiro\ objeto \\ na\ primeira\ iteração\ é\ 1(aberto) \end{pmatrix}, \tag{14}$$

para

$$\theta B_j(1) = 0, \tag{15}$$

$$\theta B_j(t) = \begin{Bmatrix} 0, \{se\ \theta B_j(t-1) = 0 \vee \\ IFI(t) = 2 : objeto\ não\ aberto, \\ 1, \{\theta B_j(t-1) = 1 \vee \begin{pmatrix} \theta B_j(t) = 0 \wedge \\ OFBB_{j-1}(t) = 1 \end{pmatrix} : objeto\ aberto, \\ 2, \{se\ \theta B_j(t-1) = 1 \wedge\ RC_j(t) < \min[S_l], \\ l \in SUI : objeto\ fechado. \end{Bmatrix} \tag{16}$$

$$RC_j(t) = RC_j(t) - S_i, \quad \text{em que i é o índice para} max\ OFTB_i(t). \tag{17}$$

Função de entrada. Se o objeto está aberto (Equação 16), este aceita como entrada a saída máxima da camada dos itens (Equação 18).

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

$$IFB_j(t) = \begin{Bmatrix} \max_i \left(OFTB_j(t)\right), \{se \theta B_j(t) = 1, \\ 0, \{se\ \theta B_j(t) = 0 \vee\ \theta B_j(t) = 2. \end{Bmatrix}. \tag{18}$$

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 (Equação 19).

$$assign_{ij}(t) = \begin{Bmatrix} 0, \{se\ S_i >\ RC_j(t), \\ 1, \{se\ S_i \leq\ RC_j(t). \end{Bmatrix}, \tag{19}$$

sendo i o índice para max $OFTB_i(t)$.

Função de saída. Quando um objeto é fechado (estado "2"), o sinal $OFBF_j(t)$ (Equação 20) é enviado para o nó final, que a partir deste sinal faz a contagem de objetos fechados.

$$OFBF_j(t) = \begin{Bmatrix} 1, \{se\ \theta B_j(t) = 2, \\ 0, \{caso\ contrário. \end{Bmatrix},\ \forall\ i \in T,\ j \in B. \tag{20}$$

O sinal *OFBB* é enviado para a abertura do próximo objeto quando o tamanho do item referente à $maxOFTB_i(t)$ é maior que a capacidade residual do objeto atual (Equação 21).

$$OFBB_j(t) = \begin{Bmatrix} 1, & \begin{Bmatrix} se\ \theta B_j(t-1) = 1 \wedge RC_j(t) < S_i(t), \\ i\ é\ o\ índice\ para\ max\ OFTB_i(t), \end{Bmatrix} \\ & 0, \{caso\ contrário. \end{Bmatrix} \tag{21}$$

Se o item é designado para o objeto, o sinal "1" é enviado para o nó do item, conforme Equação 22.

$$OFBT_{ji}(t) = \begin{Bmatrix} 1, \{se\ assign_{ij}(t) = 1, \\ 0, \{caso\ contrário. \end{Bmatrix}. \tag{22}$$

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 (Equação 23). 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* (Equação 24).

$$IFFT(t) = \sum_{i=1}^{n} OFTF_i(t), \tag{23}$$

$$IFFB(t) = \sum_{j=1}^{m} OFBF_j(t). \tag{24}$$

Função de ativação. Existem três estados possíveis como pode ser observado na Equação 25. 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.

$$\theta F(t) = \begin{Bmatrix} 0, \{se\ IFFT(t) < n, \\ 1, \{se\ IFFT(t) = n, \\ 2, \{se\ IFFT(t) = n \wedge IFFB(t) = LB. \end{Bmatrix}. \tag{25}$$

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 *LB* não foi atingido; e fim do problema (Equação 26).

$$OFFI(t) = \left\{ \begin{array}{c} 1, \{se\ \theta F(t) = 0, \\ 2, \{se\ \theta F(t) = 1 \wedge k < k_{max}, \\ 3, \{se\left(\theta F(t) = 1 \wedge k = k_{max}\right) \vee \theta F(t) = 2. \end{array} \right\}, \quad (26)$$

O valor de saída dado pela Equação 27 representa a solução do problema, ou seja, a quantidade de objetos utilizada, tanto no fim de uma designação ($OFFI(t) = 2$), como no fim do problema ($OFFI(t) = 3$).

$$OFF(k) = IFFB(t), se\ OFFI(t) = 2\ ou\ 3. \quad (27)$$

3.1.5 Estratégia de busca

A característica de busca local da RNAA é conseguida pela modificação controlada dos pesos (ω_i). Os pesos têm como função alterar o comprimento aparente do item, acarretando mudança na ordem de designação dos itens nos objetos, na busca por soluções melhores. A estratégia de busca apresentada por Agarwal (2009) consiste em, a cada iteração, alterar os pesos em função do erro (ε), medido pela diferença entre a solução anterior e o *LB*. A estratégia de busca considerada neste trabalho é aquela apresentada por Almeida & Steiner (2013b), visto se mostrar superior em relação à qualidade de solução quando comparada à estratégia de Agarwal (2009). Nesta abordagem, além de se considerar ε, também é considerada a capacidade residual do objeto ao qual o item está designado (*RC*), aumentando a variação do comprimento aparente de itens alocados em objetos mal empacotados, ou seja, com maiores valores de *RC*. Para um dado número aleatório $Rnd \in [0, 1]$, os pesos são alterados como mostrado na Equação 28:

$$\omega_i(k+1) = \left\{ \begin{array}{l} \omega_i(k) + \left(alpha * Rnd * \varepsilon * S_i * RC_j\right), \\ \{se\ Rnd < 0.5, \\ \omega_i(k) - \left(alpha * Rnd * \varepsilon * S_i * RC_j\right), \\ \{caso\ contrário. \end{array} \right\}, \forall i \in T, \quad (28)$$

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

Quando há melhora no resultado em relação à iteração anterior, um reforço (*RF*) é aplicado no vetor de pesos, seguindo a regra apresentada na Equação 29:

$$\omega_i(k) = \omega_i(k) + RF * \left(\omega_i(k) - \omega_i(k-1)\right) \forall i \in T. \quad (29)$$

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.

Na seção 4, a seguir, são apresentadas as condições em que a rede foi avaliada no presente artigo.

4 Medotologia

Partindo dos resultados obtidos pela aplicação de DOE por Almeida & Steiner (2013a), foi realizado um novo DOE com o objetivo de buscar um conjunto de parâmetros mais acurados para a RNAA. O projeto escolhido foi do tipo Fatorial Completo, em que todas as combinações entre os fatores são avaliadas e espera-se, com este teste, estimar os efeitos de cada fator, comparar os efeitos das combinações de fatores, estimar os efeitos das interações entre os fatores e estimar a variância.

Os fatores controláveis da RNAA são: *taxa de aprendizagem (alpha)*; *fator de reforço (RF)*, *fator de retorno (BF)* e *número de iterações (k)*. Os níveis dos fatores representam o intervalo de cada parâmetro, no qual se pretende analisar as respostas do algoritmo, e foram definidos a partir dos resultados obtidos por Almeida & Steiner (2013a, 2013b), sendo que os valores dos parâmetros que apresentaram melhores resultados foram: *alpha* = 0,0000005; *RF* = 2; *BF* = 1000; e *k* = 3000 (Almeida & Steiner, 2013b). Os níveis de cada fator, analisados neste trabalho, são os seguintes:

- ***alpha*:** 11 níveis (0,00000010; 0,00000059; 0,00000108; 0,00000157; 0,00000206; 0,00000255; 0,00000304; 0,00000353; 0,00000402; 0,00000451 e 0,00000500);
- ***RF*:** 7 níveis (0, 1, 2, 3, 4, 5 e 6);
- ***BF*:** 8 níveis (250, 500, 750, 1.000, 1.250, 1.500, 1.750 e 2.000);
- ***k*:** 5 níveis (1.500, 2.000, 2.500, 3.000 e 3.500).

Duas variáveis de resposta são avaliadas na análise estatística: a quantidade de resultados ótimos (ótimo) e o *gap*, que representa a diferença entre a quantidade total de objetos na solução encontrada e a quantidade de objetos na resposta ótima, sendo ótimo uma variável do tipo maior-é-melhor e *gap* do tipo menor-é-melhor. A quantidade de experimentos realizados no projeto Fatorial Completo é dada pelo produto dos níveis dos fatores e o número de repetições, ou seja, *alpha*RF*BF*k*n*, que resulta em 11*7*8*5*1. = 3080 experimentos. Toda a análise estatística foi feita com auxílio do *software* Minitab® 16.1.0.

A RNAA, bem como a heurística MBS, foram codificadas em Visual Basic® 6.0. Para a realização dos testes, foi utilizado o conjunto de problemas *benchmark* disponível em *OR-Library* (http://www.wiwi.uni-jena.de/Entscheidung/binpp/bin2dat.htm) da *Technische Universitat Darmstadt*. Este conjunto de problemas

é dividido em duas classes. Na primeira classe (*u*), o tamanho dos itens é uniformemente distribuído entre 20 e 100 e o tamanho dos objetos (*bins*) é 150. Esta classe é dividida em 4 conjuntos, u120, u250, u500 e u1000, cada conjunto contém 20 instâncias e a quantidade de itens a serem alocados nos objetos é de 120, 250, 500 e 1.000, respectivamente. A segunda classe (t) consiste em *triplets*, com itens de tamanhos variando entre 25 e 50, para serem empacotados em objetos de tamanho igual a 100. A classe *t* também é dividida em quatro conjuntos, t60, t120, t249 e t501, cada conjunto contendo 20 problemas com 60, 120, 249 e 501 itens a serem empacotados, respectivamente. Esta classe foi projetada de modo que, a cada objeto, sejam designados 3 itens, o que implica que a solução ótima seja igual à quantidade total de itens dividida por 3, além de não existir folga nos objetos na designação ótima, ou seja, o somatório dos tamanhos dos itens é igual ao somatório do tamanho dos objetos na quantidade mínima necessária, o que torna o problema difícil de ser resolvido.

O principal objetivo da análise estatística da variação dos níveis de cada fator do experimento é definir quais fatores ou interação entre fatores são mais importantes, e em que direção estes fatores devem ser ajustados para melhorar a resposta do algoritmo. Como testes preliminares mostraram que existe pouca variabilidade nas respostas para um mesmo tratamento (combinação de parâmetros) e a eficiência relativa de um experimento fatorial cresce conforme o número de fatores (Calegare, 2001), foi realizado apenas um ensaio de cada tratamento, com poucas chances de conclusões errôneas sobre os efeitos dos fatores (Montgomery, 2001).

5 Resultados

A Tabela 1 mostra o resultado da análise de variância do experimento obtido no Minitab® para a quantidade de respostas ótimas obtidas pela RNAA.

Como este experimento foi realizado sem réplicas, o valor do erro residual não pode ser estimado. Este erro serve de parâmetro para o cálculo do valor de *F*, que é a razão entre a variância explicada pelo fator e a variância relacionada à variabilidade (Ruiz et al., 2005). Quando não se possui o valor do erro residual, uma das práticas é utilizar a interação de maior ordem, neste caso, *alpha*RF*BF*k*, como estimador do erro. Como o valor de *F* é obtido pela divisão dos quadrados médios (*MS*) dos fatores pelo quadrado médio dos resíduos, o valor dos quadrados médios foi utilizado para avaliar quais fatores são mais significativos na variação das respostas. Da Tabela 1, observa-se que os fatores, taxa de aprendizagem (*alpha*), seguido pelo número de iterações (*k*) e pela interação *alpha*k*, são os maiores responsáveis pela variação nas respostas, contribuindo com aproximadamente 99,5% da variabilidade.

Os valores dos quadrados médios obtidos pela análise da variável *gap*, mostrado na Tabela 2, corroboram com as conclusões obtidas para a variável de resposta ótimo.

Nas Figuras 2 e 3, são mostrados os efeitos do fator *alpha* nas variáveis de resposta ótimo e *gap*, respectivamente. Para a variável ótimo, os melhores

Tabela 1. Análise de variância para variável ótimo.

Análise de Variância para *ótimo*, utilizando Soma de Quadrados Ajustada						
Fonte	**DF**	**Seq SS**	**Adj SS**	**Adj MS**	**F**	**P**
alpha	10	132166,6	132166,3	13216,7	*	*
RF	6	27,9	27,9	4,7	*	*
BF	7	384,3	384,3	54,9	*	*
k	4	5741,7	5741,7	1435,4	*	*
alpha*RF	60	359,0	359,0	6,0	*	*
alpha*BF	70	635,5	635,5	9,1	*	*
alpha*k	40	4688,4	4688,4	117,2	*	*
RF*BF	42	182,8	182,8	4,4	*	*
RF*k	24	159,9	159,9	6,7	*	*
BF*k	28	541,4	541,4	19,3	*	*
alpha*RF*BF	420	2416,3	2416,3	5,8	*	*
alpha*RF*k	240	1315,9	1315,9	5,5	*	*
alpha*BF*k	280	1918,8	1918,8	6,9	*	*
RF*BF*k	168	864,9	864,9	5,1	*	*
alpha*RF*BF*k	1680	9236,6	9236,6	5,5	*	*
Error	0	*	*	*		
Total	3079	160640,1				

*Valores não podem ser calculados.

resultados foram obtidos para taxas de aprendizagem maiores, com uma tendência à estabilidade para valores entre 0,0000035 e 0,0000050. Já para a variável *gap*, o algoritmo responde melhor em taxas intermediárias, entre 0,0000016 e 0,0000026.

O número de iterações (*k*), tanto aumenta a quantidade de respostas ótimas como diminui o *gap*, conforme seu valor aumenta. Entretanto, ao se aumentar o valor de *k*, deve-se levar em consideração que o tempo computacional cresce proporcionalmente (*k**[O.(n.log.n)]). As Figuras 4 e 5 mostram os efeitos de *k* nas variáveis ótimo e *gap*, respectivamente.

A interação entre *k* e *alpha*, apresentada nas Figuras 6 e 7 para as variáveis ótimo e *gap*, respectivamente, indica que, para maiores valores de *alpha*, a quantidade de iterações tem pouca influência. Para valores intermediários de *alpha*, em que se conseguiram os menores valores médios de *gap*, um aumento no número de iterações pode compensar a perda de eficiência. Para efeito de aplicações práticas, a variável de maior interesse é o *gap*, pois é a que traz maior resultado financeiro ao longo do tempo, por exemplo, reduzindo a quantidade global de estoques utilizados para produção de itens. As demais interações possuem baixo valor de *F*, o

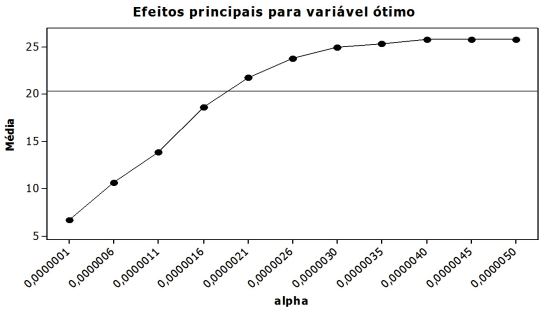

Figura 2. Efeito principal do fator *alpha* na variável de resposta ótimo.

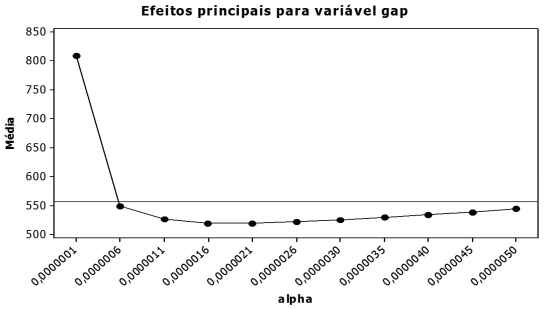

Figura 3. Efeito principal do fator *alpha* na variável de resposta *gap*.

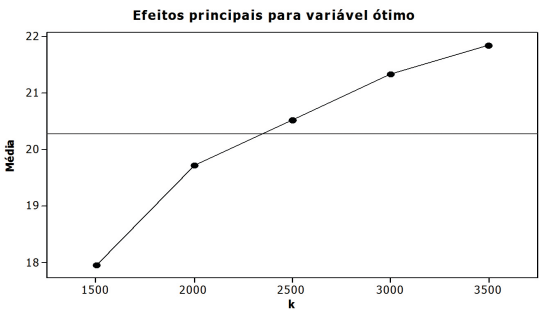

Figura 4. Efeito principal do fator *k* na variável de resposta ótimo.

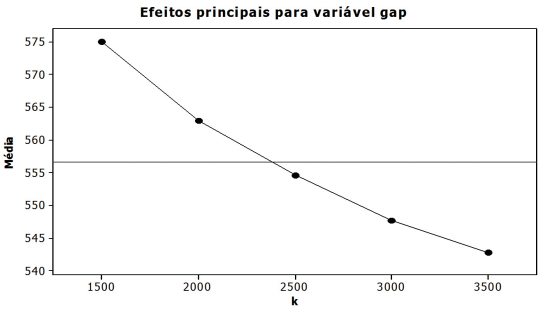

Figura 5. Efeito principal do fator *k* na variável de resposta *gap*.

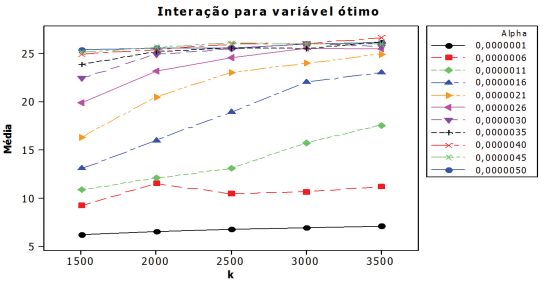

Figura 6. Efeito da interação *k*alpha* na variável de resposta ótimo.

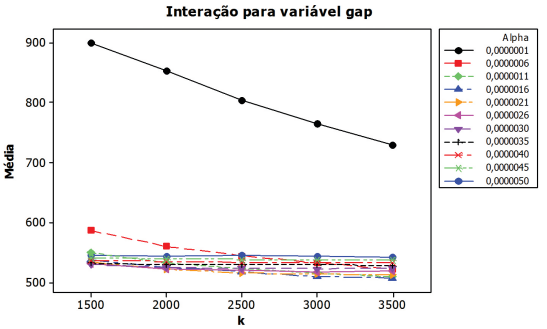

Figura 7. Efeito da interação *k*alpha* na variável de resposta *gap*.

Tabela 2. Análise de variância para variável *gap*.

Análise de Variância para *gap*, utilizando Soma de Quadrados Ajustada						
Fonte	**DF**	**Seq SS**	**Adj SS**	**Adj MS**	**F**	**P**
alpha	10	20052551	20052551	2005255	*	*
RF	6	115	115	19	*	*
BF	7	1198	1198	171	*	*
k	4	405307	405307	101327	*	*
alpha*RF	60	1477	1477	25	*	*
alpha*BF	70	3147	3147	45	*	*
alpha*k	40	879381	879381	21985	*	*
RF*BF	42	918	918	22	*	*
RF*k	24	565	565	24	*	*
BF*k	28	1397	1397	50	*	*
alpha*RF*BF	420	8643	8643	21	*	*
alpha*RF*k	240	5125	5125	21	*	*
alpha*BF*k	280	6021	6021	22	*	*
RF*BF*k	168	3105	3105	18	*	*
alpha*RF*BF*k	1680	35170	35170	21	*	*
Error	0	*	*	*		
Total	3079	21404120				

*Valores não podem ser calculados.

que indica que seus efeitos não são significativos para os resultados e, portanto, não serão analisados.

Com as conclusões obtidas pela análise estatística, foram definidas três combinações de parâmetros buscando otimização das respostas pela RNAA. Na primeira, é utilizado *alpha* = 0,0000021, que resultou nos menores valores médios de *gap*, e número de iterações igual a 3.500, que tende a produzir as melhores médias em todos os níveis dos demais parâmetros. Na segunda combinação, é utilizado *alpha* igual a 0,0000040, que produz bons resultados médios para qualquer nível do fator *k*, e *k* igual a 1.500, visando melhor eficiência computacional. Na terceira combinação, é utilizado *alpha* de 0,0000021 e número de iterações igual a 10.000, buscando melhor qualidade na solução. O baixo valor de *F* para o fator de retorno e fator de reforço indica que não existe diferença significativa entre as médias de cada nível destes fatores, ou seja, qualquer valor utilizado para estes valores produz as mesmas repostas médias. Entretanto, preferiu-se eliminar estas variáveis, tornando o algoritmo mais "enxuto".

Cada combinação foi testada 10 vezes e os resultados são comparados com os resultados da heurística MBS acrescida das restrições propostas por Fleszar & Hind (2002). Os testes foram executados em processador Intel® Core™ i3-2310M CPU @ 2.1GHz com memória de 4,00 GB.

A Tabela 3 mostra o resultado das médias obtidas com os 10 experimentos realizados para cada combinação e cada tipo de problema, para as variáveis ótimo e *gap*. A variável *tempo* representa o tempo médio, em milissegundos, necessário para a resolução de cada problema.

As médias apresentadas na Tabela 3, para as combinações 1 e 2, ficaram de acordo com as observações obtidas com a análise estatística, confirmando a significância dos fatores *alpha* e *k*, além da interação entre estes fatores. Na classe *t*, a RNAA não conseguiu encontrar nenhum resultado ótimo e os *gaps* foram bastante superiores à classe *u*, indicando ineficiência da RNAA na resolução de problemas difíceis.

Analisando a classe *u*, a combinação 1 foi 2,8% superior em relação à combinação 2 na busca por soluções ótimas. Entretanto, a combinação 2 reduziu o valor médio do *gap* em 3,5%, apesar de aumentar o tempo computacional em aproximadamente 136%. Os resultados da combinação 3 mostram que a tendência de melhor qualidade das respostas com o aumento do número de iterações, não se confirmou, dada uma diminuição de apenas 1,8% no *gap* em relação à combinação 2. Este fato pode ser um indicativo de que o algoritmo fique "preso" a soluções locais. O aumento no tempo computacional decorrente da grande quantidade de iterações foi da ordem de 186%. Na classe *t*, o *gap* médio obtido na combinação 3 foi 0,3% pior que a combinação 2, com tempo médio de processamento 200% superior.

A seguir, na Tabela 4, são apresentados os resultados obtidos com a heurística MBS.

Considerando a qualidade das respostas, a RNAA foi superior ao MBS nos conjuntos u120 e u250, com um *gap* menor em relação ao MBS na ordem de 40%, 36% e 42% no conjunto u120 e 17,1%, 15,7%

Tabela 3. Resultados médios das combinações de fatores por classe de problema.

	Combinação 1			**Combinação 2**			**Combinação 3**		
alpha	**0,0000050**			**0,0000021**			**0,0000021**		
k	**1**.500			**3**.500			**10**.000		
	Ótimo	**GAP**	**tempo**	Ótimo	**GAP**	**tempo**	Ótimo	**GAP**	**Tempo**
u120	14,0	6,0	264,0	13,6	6,4	638,7	14,2	5,8	2.014,8
u250	8,6	11,6	1.121,5	8,6	11,8	2.657,4	8,1	12,1	6.981,3
u500	2,9	18,8	5.531,6	2,6	19,2	13.456,9	2,6	18,3	36.099,3
u1000	0,0	42,6	23.237,2	0,0	38,8	54.555,3	0,0	38,6	158.884,2
Total classe u	**25,5**	**79,0**	**7.538,6**	**24,8**	**76,2**	**17.827,1**	**24,9**	**74,8**	**50.994,9**
t60	0,0	21,0	96,2	0,0	20,0	227,7	0,0	20,3	678,3
t120	0,0	48,3	342,0	0,0	45,6	789,3	0,0	45,5	2341,6
t249	0,0	122,4	1354,7	0,0	113,5	3158,5	0,0	113,5	9458,3
t501	0,0	275,9	5327,8	0,0	257,7	12381,3	0,0	258,9	37197,5
Total classe t	**0,0**	**467,6**	**1.780,2**	**0,0**	**436,8**	**4.139,2**	**0,0**	**438,2**	**12.418,9**
Total	**25,5**	**546,6**	**37.275,0**	**24,8**	**513,0**	**87.865,1**	**24,9**	**513,0**	**253.655,3**

Tabela 4. Resultados da heurística MBS por conjunto de problemas.

Problema	**Ótimo**	**GAP**	**Tempo**
u120	12	10	16,5
u250	10	14	64,5
u500	11	12	272,9
u1000	7	16	1330,6
Total classe u	**40**	**52**	**421,1**
t60	0	20	8,8
t120	0	20	40,3
t249	0	23	287,4
t501	0	41	2197,3
Total classe t	**0**	**104**	**633,5**
Total	**40**	**156**	**4218,3**

e 13,5% no conjunto u250, para as combinações 1, 2 e 3, respectivamente, sendo que no conjunto u120, a RNAA foi superior ao MBS também em relação à quantidade de respostas ótimas. O tempo computacional gasto pela RNAA foi superior, no conjunto u120, em 1.500%, 3.770% e 12.110%, respectivamente, para as combinações 1, 2 e 3. Apesar de o tempo de processamento ser percentualmente muito superior, estes tempos são menores que 1 segundo, nas combinações 1 e 2, e de aproximadamente 2 segundos, na combinação 3. No conjunto u250, o tempo foi de 1,1, 2,6, chegando a aproximadamente 7 segundos, em média, nas combinações 1, 2 e 3, nesta ordem. Estes resultados são bastante razoáveis para aplicações práticas, em que o corte de um padrão de perfil metálico, por exemplo, pode durar dezenas ou até centenas de minutos.

Nos conjuntos de problemas u500 e u1000, a heurística MBS foi mais efetiva na resolução. O tempo computacional médio gasto pela heurística MBS para resolução de cada problema do conjunto u500 foi de aproximadamente 0,3 segundos e, para cada problema do conjunto u1000, de aproximadamente 1,3 segundos. Considerando a combinação 2 da RNAA, que tem uma quantidade razoável de iterações (3.500), o tempo chega a 13,5 segundos no conjunto u500, e 54,5 segundos no conjunto u1000. Para o caso extremo analisado de 1.000 itens e 10.000 iterações, o tempo computacional pode chegar a 158,9 segundos. Considerando ainda o tempo computacional necessário para processamento e apresentação das respostas, criação de relatórios, etc., uma aplicação prática com este gasto em tempo de processamento deve ser analisada com mais critério, avaliando outras variáveis como quantidade de vezes que o problema precisa ser resolvido e disponibilidade de mão de obra. Vale ressaltar que em aplicações práticas de corte ou empacotamento, problemas que consideram 500 ou 1.000 itens são pouco comuns.

A qualidade das respostas obtidas pelo MBS nos conjuntos u500 e u1000 foi bastante superior, tanto em relação à quantidade de ótimos, quanto em relação ao *gap*. Considerando o *gap* da combinação que obteve as melhores respostas (combinação 3), a heurística MBS reduziu o *gap* em 34,4% no conjunto u500 e 78,6% no conjunto u1000.

Nos problemas da classe *t*, nenhum dos métodos foi capaz de chegar a uma resposta ótima, entretanto o *gap* obtido pela heurística MBS é drasticamente menor que aquele produzido pelo melhor caso da RNAA (combinações 2 e 3), com uma economia de aproximadamente 333 objetos, ou 76,2%. Observa-se que nos problemas mais difíceis (classe *t*) o tempo computacional cresce em relação aos problemas da classe *u*. Por exemplo, nos conjuntos u120 e t120, o tempo necessário para a resolução dos problemas de classe *t* é de aproximadamente 144% maior em relação à classe *u*, entretanto os tempos ainda são bastante inferiores aos tempos de processamento da RNAA.

Em geral, a heurística MBS foi bastante superior à meta-heurística RNAA, em todos os quesitos.

O tempo computacional médio gasto pelo MBS foi de 4,2 segundos, enquanto a RNAA, na combinação 1, que tem o menor número de iterações, registrou média de 37,3 segundos. O *gap* total obtido pela heurística MBS, que pode representar, por exemplo, uma medida de eficiência de aproveitamento de matéria-prima em processos de corte, foi de 156 objetos, enquanto que, no melhor caso da RNAA, o *gap* médio foi de 513 objetos, ou seja, num processo de corte que utilizasse RNAA para geração dos padrões de corte, seriam necessários 357 objetos a mais que um processo que utilizasse MBS, resultando em maiores custos, tanto em gasto matéria-prima, como no tratamento das consequentes sobras ocasionadas pela menor eficiência no aproveitamento dos estoques.

6 Conclusões

Este trabalho descreve os passos da implementação de uma meta-heurística RNAA para resolução de problemas do tipo *Bin Packing* unidimensional. Também é apresentada uma avaliação dos parâmetros da meta-heurística, por meio de delineamento de experimentos, com o objetivo de se buscar melhor efetividade da rede pela otimização de seus parâmetros.

A análise de significância, feita a partir de DOE do tipo Fatorial Completo colaborou para verificar como os parâmetros podem influenciar o funcionamento da RNAA. Observou-se que a *alpha* e *k*, além da interação entre estes fatores, são os maiores responsáveis pela variação nas respostas produzidas pela RNAA. Foi observada uma tendência de melhora das respostas conforme se aumenta o número de iterações, entretanto a análise da interação entre os fatores permitiu definir níveis de fatores em que o número de iterações fosse consideravelmente menor, privilegiando o tempo computacional, sem comprometer significativamente as respostas. A análise também permitiu concluir que a variação do fator de reforço e fator de retorno não causam variações nas médias, subsidiando a eliminação destes fatores sem prejudicar a resposta do algoritmo, inclusive aumentando sutilmente a eficiência computacional da RNAA. Isto coloca o DOE como um poderoso aliado no ajuste de algoritmos, em especial os meta-heurísticos. A pouca melhora obtida nas respostas com a extrapolação de *k* (10.000 iterações) indica que, possivelmente, o algoritmo fica "preso" em regiões específicas de solução, ou ótimos locais. Este fato abre espaço para uma potencial melhora do algoritmo, com a inclusão de mecanismos de fuga de ótimos locais. Este papel deveria ser realizado pelo mecanismo de retorno, entretanto os testes estatísticos mostraram que este mecanismo não produz os resultados desejados. Alguns testes foram realizados com reinicialização de pesos aleatória e os resultados foram bastante insatisfatórios.

A RNAA mostrou ser um método competitivo para a resolução de problemas com até 250 itens, exceto para a classe *t*, de problemas difíceis. Entretanto, no resultado geral, a heurística MBS foi superior em relação à qualidade das respostas e tempo computacional, reduzindo o gap em 69,6% em relação à RNAA, com tempo 88,7% menor. Vale ressaltar que, no trabalho de Fleszar & Hindi (2002), os autores propõem outras heurísticas, entre elas *perturbation MBS* e *Variable Neighbourhood Search* (VNS), cuja combinação gera excelentes resultados com pouco impacto no tempo computacional, chegando a reduzir o gap para apenas um objeto nos problemas considerados neste trabalho.

Referências

Agarwal, A. (2009). Theoretical insights into the augmented-neural-network approach for combinatorial optimization. *Annals of Operations Research*, 168(1), 101-117. http://dx.doi.org/10.1007/s10479-008-0364-8.

Agarwal, A., Pirkul, H., & Jacob, V. S. (2003). Augmented neural networks for task scheduling. *European Journal of Operational Research*, 151(3), 481-502. http://dx.doi.org/10.1016/S0377-2217(02)00605-7.

Almeida, R., & Steiner, M. T. A. (2013a). Aplicação de uma rede neural aumentada ajustada por delineamento de experimentos para resolução de problemas de corte e empacotamento. In *Anais do 16 Simpósio de Pesquisa Operacional e Logística da Marinha - SPOLM*. Rio de Janeiro.

Almeida, R., & Steiner, M. T. A. (2013b). Aplicação de uma rede neural aumentada para resolução de problemas de corte e empacotamento utilizando novas estratégias de aprendizagem. In *Anais do 45 Simpósio Brasileiro de Pesquisa Operacional - SBPO*. Natal.

Araújo, S. A., Constantino, A. A., & Poldi, K. C. (2011). An evolutionary algorithm for the one-dimensional cutting stock problem. *International Transactions in Operational Research*, 18(1), 115-127. http://dx.doi.org/10.1111/j.1475-3995.2009.00760.x.

Calegare, A. J. A. (2001). *Introdução ao delineamento de experimentos*. São Paulo: Blücher. 130 p.

Demirel, N. Ç., & Toksari, M. D. (2006). Optimization of the quadratic assignment problem using an ant colony algorithm. *Applied Mathematics and Computation*, 183(1), 427-435. http://dx.doi.org/10.1016/j.amc.2006.05.073.

Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2(1), 5-30. http://dx.doi.org/10.1007/BF00226291.

Fleszar, K., & Hindi, K. S. (2002). New heuristics for one-dimensional bin-packing. *Computers & Operations Research*, 29(7), 821-839. http://dx.doi.org/10.1016/S0305-0548(00)00082-4.

Gupta, J. N. D., & Ho, J. C. (1999). A new heuristic algorithm for the one-dimensional bin-packing problem. *Production Planning and Control*, 10(6), 598-603. http://dx.doi.org/10.1080/095372899232894.

Hopfield, J. J., & Tank, D. W. (1985). "Neural" computation of decisions in optimization problems. *Biological Cybernetics*, 52(3), 141-152. PMid:4027280.

Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R., & Graham, R. L. (1974). Worst-case performance bounds for simple one-dimensional packing algorithm. *SIAM Journal on Computing*, 3(4), 299-326. http://dx.doi.org/10.1137/0203025.

Kasap, N., & Agarwal, A. (2012). Augmented neural networks and problem structure-based heuristics for the bin-packing problem. *International Journal of Systems Science*, 43(8), 1412-1430. http://dx.doi.org/10.1080/00207721.2010.549587.

Loh, K., Golden, B., & Wasil, E. (2008). Solving the one-dimensional bin packing problem with a weight annealing heuristic. *Computers & Operations Research*, 35(7), 2283-2291. http://dx.doi.org/10.1016/j.cor.2006.10.021.

Martello, S., & Toth, P. (1990). *Knapsack problems: algorithms and computer implementations*. Cichester: John Wiley & Sons.

Montgomery, D. C. (2001). *Design and analysis of experiments* (5th ed.). New York: John Wiley & Sons.

Pan, Q., & Ruiz, R. (2012). Local search methods for the flowshop scheduling problem with flowtime minimization. *European Journal of Operational Research*, 222(1), 31-43. http://dx.doi.org/10.1016/j.ejor.2012.04.034.

Ruiz, R., Maroto, C., & Alcaraz, J. (2005). Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. *European Journal of Operational Research*, 165(1), 34-54. http://dx.doi.org/10.1016/j.ejor.2004.01.022.

Scholl, A., Klein, R., & Jürgens, C. (1997). BISON: a fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers & Operations Research*, 24(7), 627-645. http://dx.doi.org/10.1016/S0305-0548(96)00082-2.

Smith, K. A. (1999). Neural networks for combinatorial optimization: a review of more than a decade of research. *INFORMS Journal on Computing*, 11(1), 15-34. http://dx.doi.org/10.1287/ijoc.11.1.15.

Wäscher, G., Haußner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3), 1109-1130. http://dx.doi.org/10.1016/j.ejor.2005.12.047.

Woodcock, A. J., & Wilson, J. M. (2010). A hybrid tabu search/branch & bound approach to solving the generalized assignment problem. *European Journal of Operational Research*, 207(2), 566-578. http://dx.doi.org/10.1016/j.ejor.2010.05.007.