Formulação Inteira para o Problema de Empacotamento em Faixa 2D com Restrições de Balanceamento e Ordem

Thiago Alves de Queiroz,
Departamento de Matemática, UFG-CAC,
75704-020, Catalão, GO,
e-mail: tqueiroz@ic.unicamp.br.

Flávio K. Miyazawa
Instituto de Computação, IC, UNICAMP,
13084-971, Campinas, SP,
e-mail: fkm@ic.unicamp.br.

RESUMO

Neste trabalho investigamos o problema de Empacotamento em Faixa na sua versão 2D em que o centro de gravidade do empacotamento deve estar em uma região segura e os itens devem satisfazer uma ordem de descarregamento. Como consequência, sempre que um subconjunto de itens é descarregado, o centro de gravidade do empacotamento restante deve permanecer dentro da região segura. Desenvolvemos uma heurística que empacota em níveis e um modelo de programação linear inteira, ambos para o caso em que os itens não podem ser rotacionados, mas devem ser empacotados de forma ortogonal aos lados da faixa. Os testes computacionais validam os algoritmos propostos, ainda mais quando a heurística é combinada com o modelo inteiro, mostrando eficiência deste último para resolver instâncias de até médio porte.

PALAVRAS CHAVE. Problema de Empacotamento em Faixa Bidimensional, Restrição de Ordem, Restrição de Balanceamento de Carga.

Área principal. Otimização Combinatória.

ABSTRACT

In this paper we deal with the Two-Dimensional Strip Packing Problem where the center of gravity of the packing must lie on a safety region and the items must satisfy a sequence of unloading. That is, whenever a subset of items is unloaded, the center of gravity of the remaining packing must still locate in the same safety region. We propose a level-oriented heuristic and a 0-1 integer linear programming model for this problem, considering only the orthogonal oriented case. Our computational experiments validate the proposed algorithms where the combination between the heuristic and the integer model allows to solve hard medium-size problems.

KEYWORDS. Two-Dimensional Strip Packing Problem, Multi-drop Requirements, Load Balancing Constraint.

Main area. Combinatorial Optimization.

1 Introdução

Alguns trabalhos sobre problemas de Corte e Empacotamento ainda não consideram restrições práticas presentes em cenários reais, como as restrições discutidas em Bischoff e Ratcliff (1995). Estas restrições caracterizam questões práticas, como caixas que não podem suportar outros itens pesados (Junqueira et al., 2010); manter o empacotamento estável seguindo leis físicas (Gendreau et al., 2006; Junqueira et al., 2010); satisfazer uma determinada ordem de carregamento ou descarregamento dos itens (Christensen e Rousøe, 2009; Gendreau et al., 2008); fazer o balanceamento da carga, isto é, manter o peso dos itens (centro de gravidade) distribuído adequadamente dentro do empacotamento (Imai et al., 2006; Kaluzny e Shaw, 2009; Mongeau e Bes, 2003), entre outras.

Neste trabalho o foco está em propor algoritmos para o problema de Empacotamento em Faixa na sua versão bidimensional (2D) que respeite uma determinada ordem de descarregamento e que satisfaça critérios de balanceamento de carga. O problema de empacotamento é enunciado da seguinte forma:

PROBLEMA DE EMPACOTAMENTO EM FAIXA 2D (2EF): SÃO DADOS UMA FAIXA BIDIMENSIONAL $B = (L, \infty)$, ISTO É, UM RECIPIENTE RETANGULAR COM LARGURA L E ALTURA INFINITA, E UM CONJUNTO DE n ITENS RETANGULARES, SENDO CADA ITEM i DE DIMENSÕES (l_i, h_i). O OBJETIVO É OBTER UM EMPACOTAMENTO ORTOGONAL DE TODOS OS ITENS DENTRO DA FAIXA B, DE MODO QUE QUAISQUER DOIS ITENS NÃO SE SOBREPONHAM E A ALTURA DA PARTE DA FAIXA QUE É USADA SEJA MINIMIZADA.

Um empacotamento em faixa é considerado viável se todos os itens forem empacotados de forma ortogonal, isto é, os lados dos itens devem ficar paralelos aos lados da faixa, e não há sobreposição de itens. Lidamos com o caso que não permite a rotação (ortogonal) dos itens e sujeito às restrições que serão discutidas logo em seguida.

O problema 2EF é NP-difícil, uma vez que ele generaliza o problema de empacotamento *Bin Packing*, veja Hopper e Turton (2001). Aplicações deste problema surgem na indústria, como no corte de rolos de metal ou papel, e em problemas de *scheduling*.

Grande parte das abordagens propostas para o problema 2EF consistem em heurísticas. Uma visão geral das várias heurísticas propostas pode ser encontrada em Ntene e van Vuuren (2009); Riff et al. (2009). Abordagens baseadas em metaheurísticas como algoritmos genéticos, recozimento simulado, GRASP, busca Tabu podem ser encontradas em Alvarez-Valdes et al. (2008); Hopper e Turton (2001); Zhang et al. (2007). Uma recente heurística baseada em empacotamento por níveis foi proposta por Ortmann et al. (2010).

Por outro lado, também existem estratégias exatas propostas para o problema 2EF. Kenmochi et al. (2009) desenvolveram algoritmos exatos do tipo *branch-and-bound*. Lodi et al. (2004); Martello et al. (2003) propuseram estratégias baseadas em branch-and-bound e modelos de programação linear inteira. Em 2008, Cintra et al. (2008) propuseram um algoritmo baseado em geração de colunas para a versão que lida com estágios de corte.

Vamos explicar agora as restrições consideradas dentro do problema 2EF. Na restrição que envolve o descarregamento dos itens (ou restrição de ordem), há um grupo de clientes (ordens) $\mathcal{D} = \{1, \ldots, D_{\max}\}$, em que cada item i tem um número $d_i \in \mathcal{D}$. Então, os itens do cliente k são aqueles itens i cujo valor de ordem é $d_i = k$. Desejamos que nenhum item j com ordem d_j seja empacotado de forma a bloquear a saída de qualquer item i, satisfazendo $d_i < d_j$ e levando em consideração que os itens serão descarregados pela parte superior da faixa.

A restrição de ordem tem sido explorada dentro do problema de Roteamento de Veículos Capacitados com Restrições de Empacotamento (Gendreau et al., 2008; Iori et al., 2007). Iori et al. (2007) apresentaram um algoritmo exato do tipo *branch-and-cut* para a versão 2D deste problema. Já Gendreau et al. (2008) trabalharam com a versão 3D e propuseram uma heurística baseada em busca Tabu.

A outra restrição considerada é a de balanceamento de carga (ou distribuição de peso). Nesta restrição o empacotamento deve ter seu centro de gravidade (CG) o mais próximo possível de um ponto "ideal", ou do meio da faixa, ou dentro de uma região (área) chamada de *envelope*. Em algumas situações práticas este ponto ideal corresponde ao ponto médio geométrico do recipiente em consideração. Centro de gravidade é definido como o ponto em que todo o peso do objeto pode ser considerado estar concentrado.

Algumas abordagens que consideraram a restrição de balanceamento de carga foram desenvolvidas por Imai et al. (2006); Kaluzny e Shaw (2009); Mongeau e Bes (2003). Tais abordagens resolveram instâncias à otimalidade dentro de um tempo aceitável para aplicações práticas. Mongeau e Bes (2003) apresentaram um modelo de programação inteira para o problema de decidir quais itens devem ser carregados dentro do compartimento de carga de aviões satisfazendo a restrição de balanceamento de carga: manter o CG dentro de um envelope e o mais próximo possível do CG ideal. O trabalho de Imai et al. (2006) considera o planejamento de cargas (em navios) com a restrição de satisfazer a estabilidade do navio. Eles formularam o problema através de um modelo de programação inteira multi-objetivo. Kaluzny e Shaw (2009) apresentaram um modelo de programação linear mista para o problema de empacotamento bidimensional que visa otimizar o balanceamento de carga, além de satisfazer restrições como: permitir a rotação dos itens e manter certos espaçamentos entre os itens.

Os trabalhos com respeito ao balanceamento de carga estão preocupados em obter um empacotamento final que satisfaça tal restrição. Ou seja, tais trabalhos não levam em consideração o fato de que ao descarregar um subconjunto de itens, o CG pode ser movido de sua posição original e, então, resultar em instabilidade para o empacotamento restante. Um cenário onde esta restrição surge consiste na entrega de mercadorias de clientes por veículos de carga. Nesta situação, a restrição de ordem representa a ordem na qual os clientes serão visitados. Após visitar o primeiro cliente da rota, seus itens serão descarregados e, consequentemente, o empacotamento restante terá seu CG movido de sua posição original. Este processo repete-se para cada cliente da rota.

Neste trabalho será estudado o problema 2EF com as restrições de balanceamento de carga e de ordem, de modo que não somente o empacotamento final, mas cada empacotamento intermediário (após descarregar alguns itens) deve satisfazer a restrição de balanceamento de carga. Este problema será denotado por 2EFBO.

O trabalho está organizado da seguinte forma. Na próxima seção será apresentado o modelo de programação linear inteira para o problema 2EF. Na Seção 3 será discutido as restrições sobre a ordem e o balanceamento de carga, usadas para formular o problema 2EFBO. A heurística que empacota em níveis, bem como o modelo de programação linear inteira proposto serão discutidos na Seção 4. Os testes computacionais serão apresentados na Seção 5. Por fim, a Seção 6 reporta as conclusões e trabalho futuros.

2 Formulação Inicial

Os empacotamentos serão representados no Plano Cartesiano $\mathbb{R}^2$. A posição de cada item dentro do empacotamento é dada pelo seu canto inferior esquerdo. O ponto $(0,0)$ no sistema de coordenadas representa o canto inferior esquerdo da faixa.

Seja $I = (L, \infty, l_{1,\ldots,n}, h_{1,\ldots,n}, d_{1,\ldots,n}, m_{1,\ldots,n})$ uma instância do problema 2EFBO, em que cada item i possui dimensões (l_i, h_i), valor d_i para a restrição de ordem e massa m_i, para $i = 1, \ldots, n$. Vamos considerar que todos os valores em I são inteiros positivos, exceto possivelmente, os valores de massa.

O modelo inteiro apresentado na eq. (1) formula o problema 2EF. A formulação considera a faixa $B = (L, \infty)$ discretizada em uma malha de pontos. A distância entre cada ponto no eixo x e y varia de acordo com a forma como a malha foi discretizada. Vamos assumir inicialmente que c_x e

c_y representam a distância entre quaisquer duas linhas consecutivas da malha em relação ao eixo x e y, respectivamente.

Seja $\mathcal{P}$ o conjunto de pontos obtidos após a discretização da faixa. Cada ponto $p \in \mathcal{P}$ é representado por um par $p = (a,b)$. Chamamos de $\mathcal{P}_i$ o conjunto de pontos onde um item i pode ser empacotado e por $\mathcal{R}_{ip}$ o conjunto de todos os pontos $r \in \mathcal{P}$ que são cobertos pelo item i, dado que i está empacotado em p. Não estão dentro do conjunto $\mathcal{R}_{ip}$ os pontos da borda superior e da borda esquerda de i. O conjunto $\mathcal{C}$ representa as linhas horizontais (na direção da altura, eixo y), enquanto em $\mathcal{L}$ temos as linhas verticais (na direção da largura, eixo x). Observamos que um item $i = (l_i, h_i)$ empacotado no ponto $p = (a,b) \in \mathcal{P}$ cobre todas as linhas horizontais no conjunto $\mathcal{N}_{ip} = [b,\ b + \lceil \frac{h_i}{c_y} \rceil]$.

Todas as variáveis no modelo são binárias, a saber: $z_e = 1$ indica que um item cobre pelo menos um ponto $p_e \in \mathcal{P}$ da linha horizontal $e \in \mathcal{C}$; $x_{ip} = 1$ representa que um item i está empacotado no ponto $p \in \mathcal{P}$; e, $y_{ir} = 1$ indica que o ponto $r \in \mathcal{P}$ está coberto pelo item i.

A função objetivo da eq. (1) visa minimizar a altura da faixa a ser utilizada. As restrições $(1.i)$ dizem que um item i deve ser empacotado exatamente uma vez, enquanto as restrições $(1.ii)$ garantem que cada ponto da malha só pode ser coberto por no máximo um item. As restrições $(1.iii)$ asseguram que todos os pontos do conjunto $\mathcal{R}_{ip}$ estão cobertos pelo item i, dado que este está empacotado em p, e as restrições $(1.iv)$ impõe que as linhas horizontais da malha cobertas por algum item devem ser consideradas (na função objetivo). As restrições $(1.v - 1.vii)$ dizem respeito às condições de integralidade do modelo.

$$
\begin{array}{lll}
\min & \displaystyle\sum_{e=1}^{|\mathcal{C}|} e z_e & \\
(i) & \displaystyle\sum_{p \in \mathcal{P}_i} x_{ip} = 1 & i = 1,\ \ldots,\ n. \\
(ii) & \displaystyle\sum_{i=1}^{n} y_{ip} \leq 1 & \forall\ p \in \mathcal{P}. \\
(iii) & x_{ip} \leq y_{ir} & i = 1,\ \ldots,\ n;\ \forall\ p \in \mathcal{P}_i;\ \forall\ r \in \mathcal{R}_{ip}. \\
sujeito\ a: \quad (iv) & x_{ip} \leq z_e & i = 1,\ \ldots,\ n;\ \forall\ p \in \mathcal{P}_i;\ \forall\ e \in \mathcal{N}_{ip}. \\
(v) & x_{ip} \in \{0,1\} & i = 1,\ \ldots,\ n;\ \forall\ p \in \mathcal{P}. \\
(vi) & y_{qi} \in \{0,1\} & i = 1,\ \ldots,\ n;\ \forall\ q \in \mathcal{P}. \\
(vii) & z_e \in \{0,1\} & e = 1,\ \ldots,\ |\mathcal{C}|.
\end{array}
\tag{1}
$$

2.1 Desigualdades Válidas

Algumas desigualdades podem ser inseridas no conjunto de restrições da formulação (1) com o intuito de limitar o número de combinações e, então, acelerar a resolução do modelo inteiro. Apresentamos na eq. (2) restrições que objetivam evitar os itens de "flutuar" dentro da faixa. Tais restrições dizem que um item i só pode ser empacotado se este estiver apoiado sobre a face de pelo menos um outro item j, mesmo que seja uma extensão de contato mínima.

$$
x_{ip} \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{q \in \mathcal{S}_{ijp}} x_{jq} \qquad i = 1,\ \ldots,\ n;\ \forall\ p \in \mathcal{P}_i. \tag{2}
$$

sendo $\mathcal{S}_{ijp}$ o conjunto de todos os pontos onde o item j pode ser empacotado de forma que o item i esteja em contato com a face superior do item j, dado que o item i foi empacotado no ponto p.

Como consequência, também podemos modificar a função objetivo de modo que os itens sempre estejam o mais próximo possível da borda inferior da faixa, a saber.

$$
\min \quad \sum_{e=1}^{|\mathcal{C}|} e z_e + \varepsilon \Big(\sum_{i=1}^{n} \sum_{p=(\bullet,b) \in \mathcal{P}_i} b x_{ip} \Big), \tag{3}
$$

sendo o valor de ε igual a $0,0001$ nos experimentos computacionais. Além do mais, esta função objetivo é

prática no sentido de evitar soluções de mesma altura, mas que porventura tenham itens deslocados para a direita ou esquerda no eixo x.

3 Restrição de Ordem e de Balanceamento de Carga

Nesta seção descreveremos as restrições a serem adicionadas ao problema 2EF visando obter o problema 2EFBO. Como comentado, se há itens de diferentes clientes (ordens) empacotados na faixa, cuidado deve ser tomado ao descarregar os itens da faixa. Então, ao visitar um determinado cliente, seus itens não devem estar obstruídos por itens de outros clientes durante o descarregamento.

A restrição de ordem, descrita na eq. (4), faz uso de variáveis binárias. As variáveis u_{pd} indicam se o ponto $p \in \mathcal{P}$ tem valor de ordem $d \in \mathcal{D}$, tal que $\mathcal{D} = \{1, \ldots, D_{\max}\}$. Itens de mesmo valor de ordem d_j são representados pelo conjunto $\mathcal{D}_j = j$.

$$\begin{array}{lll} (i) & \sum_{d=1}^{D_{max}} u_{pd} \leq 1 & \forall\ p \in \mathcal{P}; \\ (ii) & x_{ip} \leq u_{rd_i} & i = 1,\ \ldots,\ n;\ \forall\ p \in \mathcal{P}_i;\ \forall\ r \in \mathcal{R}_{ip}; \\ (iii) & u_{p'd'} \leq \sum_{d''=d'}^{D_{max}} u_{p''d''} & \forall\ p' \in \mathcal{P};\ \forall\ d',\ d'' \in \mathcal{D};\ tal\ que\ d' \leq d'',\ p'' = \lambda(p'); \\ (iv) & u_{pd} \in \{0,1\} & \forall\ p \in \mathcal{P};\ \forall\ d \in \mathcal{D}; \end{array} \tag{4}$$

em que $\lambda(p')$ representa, dado o ponto $p' = (a,b) \in \mathcal{P}$, o ponto $p'' = (a,b')$ satisfazendo $b' = \max\{\beta \in \mathcal{C} \mid \beta < b\}$.

As restrições (4.i) asseguram que cada ponto da malha tem no máximo um valor de ordem associado; as restrições (4.ii) garantem que os pontos em $\mathcal{R}_{ip}$ têm o mesmo valor de ordem; e, as restrições (4.iii) asseguram que qualquer item i, cujo valor de ordem é d_i, deve ser empacotado sobre itens j, com d_j, satisfazendo $d_i \leq d_j$. As restrições (4.iv) dizem que as variáveis u_{pd} devem ser binárias. Observe que as restrições (4.iii) consideram somente os pontos da malha que estão no eixo y e cuja distância entre eles é de c_y unidades, ou seja, os pontos $p' = (a,b)$ e $p'' = (a,b+c_y)$.

Neste trabalho combinamos a restrição de balanceamento de carga com a restrição de ordem, veja eq. (5), da seguinte forma: se o i-ésimo cliente da rota (isto é, o conjunto de itens com valor de ordem igual a d_i, com $d_i \leq d_j$, para $j = i+1, \ldots, D_{max}$) foi visitado (ou seja, todos os itens com valor de ordem menor que d_i foram removidos da faixa), então o centro de gravidade do empacotamento restante deve permanecer dentro do envelope.

Um envelope consiste em uma área retangular (que contém o CG ideal) dentro da faixa. Consideramos esta região determinada somente observando o eixo x, tal que a coordenada, no eixo x, do centro de gravidade do empacotamento deve estar no intervalo $[x_{\text{start}}, x_{\text{end}}]$. Observe que a coordenada, no eixo y, do CG foi desconsiderada. Isto ocorreu pelo fato da função objetivo visar minimizar a altura da parte da faixa que é usada, logo não há um valor exato para a altura da faixa que permita estabelecer valores em y para o envelope.

$$\begin{array}{lll} (i) & \dfrac{\sum_{\substack{\forall\ i\ com \\ d_i \in (\mathcal{D} - \cup_{j=1}^{k-1} \mathcal{D}_j)}} \sum_{\forall\ p=(a,b) \in \mathcal{P}_i} m_i (a + \frac{l_i}{2}) x_{ip}}{\sum_{\substack{\forall\ i\ com \\ d_i \in (\mathcal{D} - \cup_{j=1}^{k-1} \mathcal{D}_j)}} m_i} \geq x_{\text{start}} & k = 1,\ \ldots,\ D_{max}. \\ (ii) & \dfrac{\sum_{\substack{\forall\ i\ com \\ d_i \in (\mathcal{D} - \cup_{j=1}^{k-1} \mathcal{D}_j)}} \sum_{\forall\ p=(a,b) \in \mathcal{P}_i} m_i (a + \frac{l_i}{2}) x_{ip}}{\sum_{\substack{\forall\ i\ com \\ d_i \in (\mathcal{D} - \cup_{j=1}^{k-1} \mathcal{D}_j)}} m_i} \leq x_{\text{end}} & k = 1,\ \ldots,\ D_{max}. \end{array} \tag{5}$$

As restrições presentes na eq. (5) asseguram que, para cada valor de $k \in \mathcal{D}$, a faixa com todos os itens, exceto aqueles cujo valor de ordem seja menor que k, possui o valor da coordenada de seu centro de gravidade, no eixo x, dentro do intervalo $[x_{\text{start}}, x_{\text{end}}]$.

Os valores corretos de x_{start} e x_{end} dependem do problema em consideração. Para termos viabilidade neste problema, assumiremos que $x_{\text{start}} \leq \frac{L}{2} \leq x_{\text{end}}$. Segundo Mongeau e Bes (2003), tais valores devem ser fornecidos pelo tomador de decisões levando em consideração incertezas, como a geometria e o peso dos itens/recipiente. Assumiremos nos testes computacionais e, para os algoritmos descritos na próxima seção, que o CG ideal corresponde, no eixo x, ao ponto médio da largura do recipiente, isto é, $\frac{L}{2}$.

4 Modelo de Programação Inteira

O modelo de programação inteira para o problema 2EFBO consiste na função objetivo (3) sujeita às restrições: (1.i – 1.vii); (2); (4); e, (5).

Este modelo é dependente dos valores de c_x e c_y (distância entre duas linhas consecutivas no eixo x e y da malha, respectivamente). Deste modo, a exatidão do modelo está intimamente relacionada com a malha de pontos gerada inicialmente. A distância entre quaisquer duas linhas consecutivas no eixo x tem valor igual a c_x. Denotaremos por BBound o algoritmo em que o modelo inteiro do problema 2EFBO considera $c_x = 1$. Sem perda de generalidade, para o eixo y, ao invés de usar c_y fixo, podemos considerar somente as linhas horizontais obtidas da combinação entre todas as alturas dos itens.

Um segundo algoritmo pode ser derivado de BBound quando obtemos as linhas verticais a partir da combinação entre todos os valores de largura dos itens. Chamaremos este algoritmo de NonBB.

4.1 Heurística baseada em Níveis

Com o intuito de comparar os resultados obtidos ao executar os algoritmos anteriores, bem como obter uma altura inicial (limitante superior da solução ótima) para o modelo de programação inteira, desenvolvemos uma heurística que considera a restrição de ordem e o balanceamento de carga.

Seja $L = (1,\ldots,n)$ uma lista com n itens, em que $i \in L$ representa um item com dimensões $i = (l_i, h_i)$, ordem d_i e massa m_i. Dado um empacotamento $\mathcal{P}$ de L com n itens, denotamos por center$(\mathcal{P})$ o centro de gravidade deste empacotamento, calculado da seguinte maneira:

$$\text{center}(\mathcal{P}) = \frac{\sum_{i=1}^{n} \overrightarrow{r_i} m_i}{\sum_{i=1}^{n} m_i} \quad \Rightarrow \quad \begin{array}{l} \text{center}_x(\mathcal{P}) = \frac{\sum_{i=1}^{n} r_i^x m_i}{\sum_{i=1}^{n} m_i} \\ \\ \text{center}_y(\mathcal{P}) = \frac{\sum_{i=1}^{n} r_i^y m_i}{\sum_{i=1}^{n} m_i} \end{array} \tag{6}$$

em que center$(\mathcal{P})$, com componentes x e y, é o vetor que indica o centro de gravidade do empacotamento $\mathcal{P}$; m_i é a massa do i-ésimo item; e, $\overrightarrow{r_i}(r_i^x,\ r_i^y)$ é o vetor distância, contendo a distância entre um referencial inercial (1ª Lei de Newton) e o centro de gravidade do item i.

O ponto $(0,0)$ do recipiente foi adotado como referencial inercial. O centro de gravidade de um item retangular qualquer (homogêneo e sujeito ao mesmo campo gravitacional, que é o nosso caso) corresponde ao ponto médio das dimensões do item, isto é, center$(i) = (\frac{l_i}{2}, \frac{h_i}{2})$. Chamaremos de *nível* um empacotamento dentro da faixa, em que os itens estão organizados lado a lado (na direção do eixo x) e foram empacotados na mesma altura. O item mais alto dentro de um nível N_t determina a altura daquele nível h_{N_t}. Um empacotamento baseado em níveis é um empacotamento formado por uma sequência de níveis $\mathcal{P} = (N_1,\ldots,N_k)$, tal que o nível N_1 está na borda inferior da faixa, isto é, começa no ponto $(0,0)$, e o nível N_{i+1} está empacotado imediatamente acima do nível N_i, para $i = 1,\ldots,k-1$.

A partir disto, considere inicialmente o algoritmo FFDH *(First Fit Decreasing Height)* para o problema 2EF (Coffman, Jr. et al., 1980). Este algoritmo fornece um empacotamento baseado em níveis da seguinte forma: Primeiro, os itens são ordenados de forma decrescente pela altura. Então, ele empacota o próximo item no primeiro nível gerado (observando a ordem com que os níveis foram gerados) após o último item empacotado naquele nível. Caso não haja nível algum, o algoritmo FFDH empacota o item mais a esquerda possível dentro de um novo nível. Apresentaremos adiante uma heurística baseada no algoritmo FFDH para resolver o problema 2EFBO, que obtém um empacotamento cujo centro de gravidade, no eixo x, está no intervalo $[x_{\text{start}}, x_{\text{end}}]$. Chamaremos este algoritmo de *Balanced First Fit* (BFF).

O algoritmo BFF separa inicialmente os itens em listas, em que cada lista contém itens de mesma ordem. Os itens dentro de cada lista estão ordenados de forma decrescente pelo valor de largura (linhas $1.1 - 1.2$). No laço das linhas $1.3 - 1.22$, o algoritmo empacota os itens, que estão organizados por valor de ordem nas listas D_j, em níveis horizontais, tal que todos os itens tem mesma ordem dentro de um nível e, para

Algoritmo 1: Balanced First Fit (BFF)

Entrada: Instância $I = (L, \infty, l, h, d, m, x_{\text{start}}, x_{\text{end}})$ do problema 2EFBO.
Saída: Solução para I.

1.1 Seja $D = \{D_1, \ldots, D_p\}$ um conjunto com listas, onde D_j possui os itens de L tal que se $i \in D_j$, temos $d_i = j$.
1.2 Para cada lista D_j ($j = 1, \ldots, p$), ordene-a, de forma decrescente, observando os valores de largura dos itens na lista.
1.3 **para** $j \leftarrow 1$ **to** p **faça**
1.4 Seja um empacotamento $\mathcal{P}_j \leftarrow \{\}$.
1.5 **para cada** *item* $r \in D_j$, *seguindo a ordenação de* D_j **faça**
1.6 Seja $\mathcal{P}_j = (N_1, \ldots, N_k)$ a sequência dos níveis em $\mathcal{P}_j$.
1.7 Seja t o índice do primeiro nível, N_t, que possui somente itens de ordem d_r e tal que $h_r \leq h_{N_t}$ e $l_{N_t} + l_r \leq L$.
1.8 **se** *existe tal nível* t **então**
1.9 Seja N'_t (respectivamente, N''_t) o nível N_t adicionado do item r, o qual foi empacotado a esquerda (respectivamente, a direita) dos itens em N_t.
1.10 Crie $\mathcal{P}'_j \leftarrow (\mathcal{P}_j - N_t) \cup N'_t$.
1.11 Crie $\mathcal{P}''_j \leftarrow (\mathcal{P}_j - N_t) \cup N''_t$.
1.12 Organize os itens em N'_t de forma que $c' := |\frac{L}{2} - \text{center}_x(\mathcal{P}'_j)|$ seja mínimo.
1.13 Organize os itens em N''_t de forma que $c'' := |\frac{L}{2} - \text{center}_x(\mathcal{P}''_j)|$ seja mínimo.
1.14 **se** $c' < c''$ **então**
1.15 $\mathcal{P}_j \leftarrow \mathcal{P}'_j$.
1.16 **senão**
1.17 $\mathcal{P}_j \leftarrow \mathcal{P}''_j$.
1.18 **senão**
1.19 Empacote o item r, no canto inferior esquerdo, em um novo nível N_{k+1}.
1.20 Faça $\mathcal{P}_j \leftarrow \mathcal{P}_j \cup N_{k+1}$.
1.21 Organize o item r em N_{k+1} de forma que $|\frac{L}{2} - \text{center}_x(\mathcal{P}_j)|$ seja mínimo.
1.22 *Balance*$(L; x_{\text{start}}, x_{\text{end}}; \mathcal{P}_1, \ldots, \mathcal{P}_j)$.
1.23 **retorna** *a concatenação de* $\mathcal{P}_p, \ldots, \mathcal{P}_1$.

quaisquer dois níveis adjacentes N_i e N_{i+1}, a ordem dos itens em N_{i+1} é menor ou igual a ordem dos itens em N_i. Na iteração corrente do laço, o algoritmo BFF procura nas linhas $1.5 - 1.21$ o primeiro nível de mesma ordem deste item e que tenha espaço suficiente para ele. Se existir tal nível, o item r será empacotado mais a esquerda ou mais a direita dentro de tal nível, observando quem levará o CG do empacotamento atual para o mais próximo possível do CG ideal. Caso não exista tal nível (linhas $1.18 - 1.21$), um novo nível será criado para conter o item r, o qual será organizado de forma a manter o CG do empacotamento atual o mais próximo possível do CG ideal. Por fim, na linha 1.22, o algoritmo BFF invoca a sub-rotina *Balance*, Algoritmo 2, que tem o objetivo de fazer com que o CG do empacotamento na iteração atual, mas especificadamente, os níveis criados em tal iteração tenham exatamente seu CG igual ao CG ideal, que é $\frac{L}{2}$, observando o eixo x.

Lema 4.1 *O algoritmo Balance, Algoritmo 2, recebe um empacotamento baseado em níveis $\mathcal{P}$ e retorna um empacotamento $\mathcal{P}'$, tal que $Altura(\mathcal{P}') \leq Altura(\mathcal{P}) + Z$, sendo Z a altura do item mais alto em $\mathcal{P}$.*

Prova. Vamos considerar que $\text{center}_x(\mathcal{P}) < x_{\text{start}}$. A prova quando temos $\text{center}_x(\mathcal{P}) > x_{\text{end}}$ é feita de forma análoga.

Primeiro note, pelo passo 2.1, que o único empacotamento que é atualizado é o empacotamento $\mathcal{P}_j$ e supomos que o CG do empacotamento $\mathcal{P}_1 \cup \ldots \cup \mathcal{P}_{j-1}$ está no intervalo $[x_{\text{start}}, x_{\text{end}}]$. Observando o algoritmo Balance, nas linhas $2.2 - 2.14$, este algoritmo processa os níveis, indo do nível mais baixo de $\mathcal{P}_j$ em direção aos níveis no topo da faixa. Para cada nível, os itens são ordenados de forma decrescente pela altura e reorganizados lado a lado, dentro do nível, da direita para a esquerda. Então, deve existir um primeiro nível t no qual a reorganização deste nível modifica o empacotamento corrente de $\mathcal{P}'$ para um empacotamento $\mathcal{P}''$, tal que $\text{center}_x(\mathcal{P}') < x_{\text{start}}$ e $\text{center}(\mathcal{P}'') \geq x_{\text{start}}$. Este é o nível N_t com todos os itens, ordenados de forma decrescente pelo valor de altura, organizados lado a lado da direita para a esquerda, que faz com que o laço das linhas $2.4 - 2.7$ pare.

Algoritmo 2: Balance

Entrada: Largura L; delimitantes x_{start} e x_{end}; empacotamento $\mathcal{P} = \mathcal{P}_1, \ldots, \mathcal{P}_j$.

2.1 Sejam os níveis em $\mathcal{P}_j = \{N_1, \ldots, N_k\}$, organizados de forma que N_1 é o primeiro nível e N_k é o nível mais alto dentro do empacotamento.

2.2 **se** $\text{center}_x(\mathcal{P}_1 \| \ldots \| \mathcal{P}_j) < x_{\text{start}}$ **então**

2.3 Seja $t \leftarrow 0$.

2.4 **enquanto** $t < k$ **E** $\text{center}_x(\mathcal{P}_1 \| \ldots \| \mathcal{P}_j) < x_{\text{start}}$ **faça**

2.5 $t \leftarrow t + 1$.

2.6 Ordene os itens em N_t, de forma crescente, observando seus valores de altura.

2.7 Desloque os itens em N_t, iniciando da direita para o início do nível.

2.8 **se** $\text{center}_x(\mathcal{P}_1 \| \ldots \| \mathcal{P}_j) > x_{\text{end}}$ **então**

2.9 Gere um novo nível vazio N'_t em $\mathcal{P}_j$, entre o nível N_t e N_{t+1} (se existir). Se necessário, atualize a coordenada y dos outros itens acima de N_t. Atualize $\mathcal{P}_j$.

2.10 **enquanto** $\text{center}_x(\mathcal{P}_1 \| \ldots \| \mathcal{P}_j) > x_{\text{end}}$ **faça**

2.11 Remova o item r, que está mais a direita, em N_t (sem modificar a posição dos outros itens em N_t).

2.12 Insira r no nível N'_t logo a direita dos itens em N'_t (os itens em N'_t estão sendo empacotados da esquerda para a direita).

2.13 **se** $\text{center}_x(\mathcal{P}_1 \| \ldots \| \mathcal{P}_j) < x_{\text{start}}$ **então**

2.14 Mova o item r em N'_t da direita em direção ao fim do nível de modo que $\text{center}_x(\mathcal{P}_1 \| \ldots \| \mathcal{P}_j) = \frac{L}{2}$.

2.15 **se** $\text{center}_x(\mathcal{P}_1 \| \ldots \| \mathcal{P}_j) > x_{\text{end}}$ **então**

2.16 Proceda de forma semelhante como feito quando $\text{center}_x(\mathcal{P}_1 \| \ldots \| \mathcal{P}_j) < x_{\text{start}}$ (linhas $2.2 - 2.14$), só que agora deslocando os itens para a esquerda considerando um novo nível.

Após a execução desse primeiro laço *enquanto*, o empacotamento atual $\mathcal{P}$ tem seu centro de gravidade $\text{center}_x(\mathcal{P}) \geq x_{\text{start}}$. Então, caso $\text{center}_x(\mathcal{P}) > x_{\text{end}}$, seguindo a linha 2.9 do algoritmo, um novo nível N'_t é criado acima do nível N_t e abaixo do nível N_{t+1} (caso ela exista). Os níveis acima do nível N_t são movidos para cima de forma a termos espaço suficiente para inserir itens no novo nível N'_t. A cada iteração do segundo laço *enquanto* (linhas $2.10 - 2.12$), o algoritmo Balance remove o item de N_t de maior altura (item mais a direita dentro do nível) e o organiza dentro do nível N'_t colocando logo após os itens já existentes. Isto se repete até que CG do empacotamento atual, observando o eixo x, fique menor ou igual que x_{end}. Por acaso, pode existir um tal item r, que após ser transferido para o nível N'_t, fez com que o CG do empacotamento atual ficasse menor que x_{start} (linha 2.13). Visto que o item r é o item mais a direita dentro de N'_t, deve existir uma posição entre a posição atual de r e posição mais direita do nível N'_t, tal que o empacotamento atual terá seu CG, observando o eixo x, maior ou igual que x_{start}. Esta é a posição escolhida pelo algoritmo Balance para empacotar o item r. Note que apenas mais um nível foi criado no empacotamento inicial $\mathcal{P}$, tal que sua altura é limitada pela altura Z do item mais alto do empacotamento, mostrando que a altura do empacotamento final $\mathcal{P}'$ é menor ou igual a $Altura(\mathcal{P}) + Z$. □

5 Testes Computacionais

Os algoritmos foram codificados na linguagem *C* e o modelo de programação linear inteira resolvido pelo ILOG® CPLEX® 12 Callable Library (usando parâmetros padrões). A ideia do algoritmo principal é usar a heurística BFF para gerar a altura e uma solução inicial que, então, será usada como limitante superior pelo algoritmo NonBB. Assim, a solução retornada por NonBB será utilizada como solução inicial para o algoritmo BBound.

O resolvedor CPLEX usa um algoritmo *branch-and-cut* (referenciado em alguns contextos como *branch-and-bound*) para resolver problemas de programação linear mista. Ao utilizar este resolvedor com seus parâmetros padrões, ele automaticamente gerencia o processo de otimização e, se necessário, adiciona cortes. Chamaremos os cortes inseridos automaticamente pelo CPLEX de *cortes padrões*. Entre estes cortes, citam-se: cortes de clique, cortes de cobertura, cortes de fluxo, etc. Mais detalhes podem ser obtidos em IBM (2009).

Os testes computacionais ocorreram em um computador com processador Intel® Core^{TM} 2 Quad 2,4 GHz, 4 GB de memória RAM e sistema operacional *Linux*. Limitamos o tempo de execução para o CPLEX em 7200 segundos.

Consideramos duas classes principais de instâncias. A primeira consiste nas instâncias *cgcut01–cgcut03* e *ngcut01–ngcut12* presentes na OR-Library (Beasley, 1990). Estas instâncias foram utilizadas em testes computacionais para o problema da Mochila Restrita Bidimensional considerando cortes guilhotinados e não-guilhotinados, respectivamente.

A segunda classe de instâncias consiste em instâncias geradas de forma randômica para o problema 2EFBO. Esta classe está dividida em três grupos, cada qual com 6 instâncias, e a faixa com largura 20, 40 e 60, respectivamente. Em cada grupo há três instâncias com 8 itens e três instâncias com 15 itens. As dimensões de cada item i foram escolhidas de forma randômica variando em 10% a 40% da largura da faixa em consideração. Chamaremos estas instâncias de *rand01–rand03* mais as informações da largura da faixa e da quantidade de itens. Então, a instância rand03_{20}^{15} possui 15 itens e corresponde a uma instância do grupo em que a faixa tem largura 20.

Adotamos como valor ideal para o CG aquele correspondente ao ponto médio das dimensões da faixa, $\frac{L}{2}$. Deste modo, teremos nos experimentos numéricos: $x_{start} = 0,35L$ e $x_{end} = 0,65L$. O valor da massa de cada item i corresponde a sua área ($m_i = l_i h_i$), e o valor de ordem d_i é um valor inteiro obtido de forma randômica no intervalo $[1,2,3]$. Todas as instâncias utilizadas estão disponíveis via e-mail.

5.1 Resultados

Apresentamos inicialmente algumas informações sobre as instâncias. Estas informações incluem a altura inicial usada para computar a malha de pontos, o número de variáveis e de restrições para cada modelo presente nos algoritmos NonBB e BBound. Os valores referentes ao número de variáveis e de restrições foram obtidos após a fase de pré-processamento do CPLEX. As seguintes informações são apresentadas nas linhas da Tabela 1: nome da instância; largura da faixa L; quantidade de itens n; altura inicial usada para computar a malha de pontos *Altura*, número de variáveis *NVar* e restrições *NRest* para os modelos resolvidos pelos algoritmos NonBB e BBound, respectivamente; e, a diferença percentual no número de variáveis entre tais modelos de NonBB e BBound.

Como mencionado, a altura inicial usada pelo modelo do algoritmo NonBB corresponde à altura retornada pela heurística BFF. Por outro lado, a altura inicial para o modelo do BBound corresponde à altura retornada pelo algoritmo NonBB. Caso este último algoritmo não encontre uma solução para o problema dentro do tempo limite imposto, a altura inicial do BBound corresponde a altura encontrada pela heurística. Como resultado, algumas instâncias presentes na Tabela 1 possuem a coluna D_{var} com valor negativo.

Observando a Tabela 1, o número de instâncias em que o modelo do algoritmo BBound possui menos variáveis e restrições que o do NonBB é igual a 10 (de um total de 33), que representa 30,30% do total de instâncias. A diferença no número de variáveis é de 59,55%, na média. Por outro lado, o modelo do algoritmo NonBB possui menos variáveis e restrições que o do BBound somente para as instâncias nas quais o algoritmo NonBB não conseguiu resolver dentro do tempo limite. De qualquer forma, vale destacar que as instâncias são difíceis de serem resolvidas, visto terem milhares de variáveis e restrições no modelo inteiro de ambos os algoritmos.

A Tabela 2 mostra os resultados obtidos pela heurística e pelos algoritmos NonBB e BBound para cada instância utilizada. Em cada linha desta tabela há os seguintes dados: nome da instância; para a heurística BFF: altura do empacotamento resultante (H) e tempo gasto, em segundos, para resolver a instância (Tempo); para NonBB e BBound: número de cortes padrões inseridos pelo CPLEX (Cortes); altura do empacotamento resultante (H); centro de gravidade do empacotamento resultante (CG); tempo gasto, em segundos (Tempo).

De acordo com os dados na Tabela 2, as instâncias com a entrada "–"são aquelas em que o tempo limite para resolver foi alcançado pelo CPLEX e nenhuma solução viável foi encontrada. Por outro lado, existem instâncias em que o tempo limite foi alcançado, mas pelo menos uma solução viável foi encontrada durante a otimização (veja instâncias *ngcut06* e *ngcut11*).

Observando os dados da Tabela 2, vemos que o tempo requerido pela heurística, na média, é de 0,9 segundos, enquanto que o tempo requerido pelos algoritmos NonBB e BBound foi de 4599,12 e 3798,67 segundos, na média. Por outro lado, os algoritmos NonBB e BBound retornaram valores ótimos de altura para 36,37% e 45,46% das instâncias, respectivamente. A heurística BFF não conseguiu obter solução ótima para nenhuma das instâncias se comparado com os algoritmos anteriores. O número de cortes padrões aplicados pelo CPLEX foi de 378 e 356, na média, para os algoritmos NonBB e BBound, respectivamente.

Tabela 1: Informações sobre os modelos dos algoritmos NonBB e BBound.

Instância	L	n	NonBB			BBound			D_{var}
			Altura	*NVar*	*NRest*	*Altura*	*NVar*	*NRest*	(%)
cgcut01	15	7	23	2444	15881	8	2048	13057	-19,34
cgcut02	40	10	151	75276	4314235	0	103788	9019012	27,47
cgcut03	40	20	551	571419	37730292	551	913959	206733048	37,48
ngcut01	10	5	24	1185	5695	17	1441	7480	17,77
ngcut02	10	7	29	3234	20476	14	2394	14563	-35,09
ngcut03	10	10	36	6727	43277	16	3856	24725	-74,46
ngcut04	15	5	11	726	2085	8	1568	9311	53,70
ngcut05	15	7	25	3000	14281	15	3840	30729	21,88
ngcut06	15	10	46	10110	102447	30	10830	116246	6,65
ngcut07	20	5	18	1890	8394	11	2871	14224	34,17
ngcut08	20	7	29	3762	20678	19	6138	81808	38,71
ngcut09	20	10	63	17328	226403	62	23088	450816	24,95
ngcut10	30	5	55	1501	5650	41	5474	35616	72,58
ngcut11	30	7	61	10250	124976	33	14819	373378	30,83
ngcut12	30	10	169	57205	2933332	0	61285	3298010	6,66
$rand01^{8}_{20}$	20	8	32	4459	44985	9	3048	25644	-46,29
$rand02^{8}_{20}$	20	8	40	7240	99807	12	3810	35165	-90,03
$rand03^{8}_{20}$	20	8	32	4459	42054	9	3048	23152	-46,29
$rand01^{15}_{20}$	20	15	47	11646	94729	16	10215	82133	-14,01
$rand02^{15}_{20}$	20	15	47	7764	55223	8	4767	28342	-62,87
$rand03^{15}_{20}$	20	15	49	9705	71466	7	4086	21219	-137,52
$rand01^{8}_{40}$	40	8	57	14234	301899	15	8371	114289	-70,04
$rand02^{8}_{40}$	40	8	97	11304	230581	0	13698	320503	17,48
$rand03^{8}_{40}$	40	8	69	19980	687342	49	22830	842379	12,48
$rand01^{15}_{40}$	40	15	113	48831	1637200	0	55801	2007967	12,49
$rand02^{15}_{40}$	40	15	85	37770	973085	0	40830	1084487	7,49
$rand03^{15}_{40}$	40	15	97	55396	1811693	0	59884	2019407	7,49
$rand01^{8}_{60}$	60	8	75	24192	750064	49	27384	872802	11,66
$rand02^{8}_{60}$	60	8	84	12782	238866	25	15974	319128	19,98
$rand03^{8}_{60}$	60	8	76	18640	654576	23	19397	624165	3,90
$rand01^{15}_{60}$	60	15	142	56947	2655964	0	63271	3064060	10,00
$rand02^{15}_{60}$	60	15	139	72120	3963316	0	81640	4717746	11,66
$rand03^{15}_{60}$	60	15	148	74840	3764584	0	81640	4212799	8,33

O algoritmo BBound mostrou ser mais eficiente, em quantidade de soluções encontradas, que o NonBB. Porém, como apresentado na Tabela 1, a altura usada por BBound para calcular a malha de pontos foi diferente daquela usada por NonBB para 54.55% das instâncias. Dentro do tempo limite imposto, 15 das 33 instâncias foram resolvidas à otimalidade pelo BBound, enquanto que com o NonBB este número cai para 12 instâncias no total.

Podemos afirmar que o algoritmo BBound requer mais tempo de processamento comparado ao NonBB quando ambos trabalham com a mesma altura inicial. Então, fica evidente que a combinação feita entre a heurística BFF e estes algoritmos é vantajosa no sentido de permitir os últimos lidarem com instâncias maiores dentro do tempo computacional desejado.

6 Conclusões e Trabalhos Futuros

Este trabalho lida com o problema de Empacotamento em Faixa Bidimensional sujeito às restrições de ordem e balanceamento de carga. Foram propostas algumas abordagens para resolver o problema: uma heurística que realiza o empacotamento em níveis horizontais e um modelo de programação linear, que forneceu dois algoritmos que geram a malha de pontos de forma diferente.

As abordagens propostas foram testadas em duas classes de instâncias: a primeira obtida da literatura, e a segunda gerada de forma randômica. Notamos que a precisão do modelo está intimamente relacionada

Tabela 2: Soluções obtidas por BFF, NonBB e BBound.

Instância	BFF		NonBB				BBound			
	H	Tempo (s)	*Cortes*	*H*	CG	Tempo (s)	*Cortes*	*H*	CG	Tempo (s)
cgcut01	13	2,00	423	8	(7,01; 3,86)	12,00	187	8	(7,01; 3,86)	6,00
cgcut02	122	0,00	-	-	(- ; -)	7214,00	0	122	(21,14; 57,20)	7490,00
cgcut03	534	1,00	-	-	(- ; -)	7201,00	-	-	(- ; -)	7209,00
ngcut01	24	8,00	53	17	(4,10; 8,99)	1,00	4	17	(4,10; 8,99)	0,00
ngcut02	21	0,00	306	14	(4,82; 6,14)	17,00	104	14	(4,82; 6,14)	4,00
ngcut03	31	1,00	155	16	(4,55; 7,85)	1306,00	88	16	(4,55; 7,85)	84,00
ngcut04	11	1,00	0	8	(6,92; 4,25)	0,00	5	8	(6,92; 4,25)	1,00
ngcut05	25	1,0	338	15	(6,72; 7,23)	33,00	157	15	(6,72; 7,23)	27,00
ngcut06	30	0,00	309	30	(7,47; 14,23)	7201,00	38	30	(7,47; 14,23)	7200,00
ngcut07	18	1,00	107	11	(10,70; 4,06)	1,00	6	11	(10,70; 4,06)	1,00
ngcut08	23	0,00	331	19	(9,09; 9,38)	12,00	159	19	(9,09; 9,38)	2221,00
ngcut09	49	1,00	-	-	(- ; -)	7200,00	1149	49	(10,48; 25,83)	7202,00
ngcut10	55	0,00	0	41	(13,20; 19,86)	0,00	21	41	(13,20; 19,86)	2,00
ngcut11	54	0,00	779	33	(15,88; 15,15)	7200,00	1295	33	(15,88; 15,15)	7201,00
ngcut12	92	0,00	-	-	(- ; -)	7203,00	0	92	(14,08; 43,75)	7205,00
$rand01^{8}_{20}$	14	0,00	132	9	(9,86; 4,03)	672,00	159	9	(9,86; 4,03)	139,00
$rand02^{8}_{20}$	22	0,00	764	12	(9,96; 6,00)	4703,00	22	12	(9,96; 6,00)	11,00
$rand03^{8}_{20}$	14	1,00	54	9	(10,39; 4,22)	932,00	175	9	(10,39; 4,22)	81,00
$rand01^{15}_{20}$	19	1,00	476	16	(11,61; 5,51)	7200,00	13	16	(11,61; 5,51)	7200,00
$rand02^{15}_{20}$	13	1,00	54	8	(10,31; 3,62)	7200,00	104	8	(10,31; 3,52)	1605,00
$rand03^{15}_{20}$	16	1,00	44	7	(9,95; 3,45)	7200,00	270	7	(10,18; 3,43)	570,00
$rand01^{8}_{40}$	40	1,00	589	15	(19,46; 5,12)	7200,00	807	15	(19,46; 5,12)	877,00
$rand02^{8}_{40}$	35	1,00	-	-	(- ; -)	7200,00	-	-	(- ; -)	7202,00
$rand03^{8}_{40}$	36	1,00	-	-	(- ; -)	7202,00	-	-	(- ; -)	7205,00
$rand01^{15}_{40}$	44	0,00	-	-	(- ; -)	7204,00	-	-	(- ; -)	7211,00
$rand02^{15}_{40}$	33	1,00	-	-	(- ; -)	7201,00	-	-	(- ; -)	7203,00
$rand03^{15}_{40}$	47	1,00	-	-	(- ; -)	7208,00	-	-	(- ; -)	7202,00
$rand01^{8}_{60}$	35	1,00	-	-	(- ; -)	7206,00	-	-	(- ; -)	7201,00
$rand02^{8}_{60}$	25	0,00	1155	25	(30,41; 11,17)	7201,00	1683	22	(30,93; 7,15)	7200,00
$rand03^{8}_{60}$	26	1,00	1490	23	(33,80; 10,25)	7211,00	1271	23	(33,80; 10,25)	7200,00
$rand01^{15}_{60}$	41	1,00	-	-	(- ; -)	7205,00	-	-	(- ; -)	7210,00
$rand02^{15}_{60}$	45	0,00	-	-	(- ; -)	7222,00	-	-	(- ; -)	7234,00
$rand03^{15}_{60}$	46	1,00	-	-	(- ; -)	7203,00	-	-	(- ; -)	7201,00

com a malha de pontos, isto é, a distância entre os pontos na malha. Como consequência, trabalhar com uma malha de pontos mais refinada exige um maior tempo de processamento. Porém, obtemos maior proveito do modelo inteiro quando este é combinado com a heurística que empacota em níveis.

Através dos testes, notamos que o modelo de programação linear inteira é viável e consistente para situações práticas, porém é limitado para resolver, de forma rápida, instâncias cujo número de pontos na malha é relativamente pequeno. Por outro lado, a heurística é capaz de lidar com instâncias maiores, porém, não conseguimos obter com ela a solução ótima para as instâncias utilizadas.

Por fim, a heurística e o modelo de programação inteira apresentados permitem resolver, de maneira satisfatória, o problema de Empacotamento em Faixa sobre um cenário mais realístico (com restrições práticas). Em todo o caso, consideráveis melhorias e extensões das abordagens se fazem necessárias para lidar com instâncias maiores e outras restrições práticas. Veja, por exemplo, Bischoff e Ratcliff (1995) para uma lista de restrições de interesse prático que podem ser consideradas dentro de problemas de empacotamento.

Agradecimentos

Os autores agradecem o apoio financeiro recebido pelo CNPq e pela FAPESP.

Referências

R. Alvarez-Valdes, F. Parre no e J. M. Tamarit. (2008), Reactive grasp for the strip-packing problem. *Computers & Operations Research*, 35:1065–1083.

J. E. Beasley. (1990), OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072.

E.E. Bischoff e M.S.W. Ratcliff. (1995), Issues in the development of approaches to container loading. *OMEGA*, 23(4):377–390.

S. G. Christensen e D. M. Rousøe. (2009), Container loading with multi-drop constraints. *International Transactions in Operational Research*, 16(6):727–743.

G. F. Cintra, F. K. Miyazawa, Y. Wakabayashi e E. C. Xavier. (2008), Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. *European Journal of Operational Research*, 191:59–83.

E. G. Coffman, Jr., M. R. Garey, D. S. Johnson e R. E. Tarjan. (1980), Performance bounds for level oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9:808–826.

M. Gendreau, M. Iori, G. Laporte e S. Martello. (2006), A tabu search algorithm for a routing and container loading problem. *Transportation Science*, 40(3):342–350.

M. Gendreau, M. Iori, G. Laporte e S. Martello. (2008), A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. *Networks*, 51:14–18.

E. Hopper e B. C. H. Turton. (2001), A review of the application of meta-heuristic algorithms to 2d strip packing problems. *Artificial Intelligence Review*, 16(4):257–300.

IBM. (2009), *ILOG® CPLEX® V12.1 - User's Manual for CPLEX®*. IBM Corporation.

A. Imai, K. Sasaki, E. Nishimura e S. Papadimitriou. (2006), Multi-objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks. *European J. Operational Research*, 171(2):373–389.

M. Iori, J. Salazar-González e D. Vigo. (2007), An exact approach for the vehicle routing problem with two-dimensional loading constrains. *Transportation Science*, 41(2):253–264.

L. Junqueira, R. Morabito e D. S. Yamashita. (2010), Three-dimensional container loading models with cargo stability and load bearing constraints. *Computers & Operations Research*, doi:10.1016/j.cor.2010.07.017.

B. L. Kaluzny e R. H. A. David Shaw. (2009), Optimal aircraft load balancing. *International Transactions in Operational Research*, 16(6):767–787.

M. Kenmochi, T. Imamichi, K. Nonobe, M. Yagiura e H. Nagamochi. (2009), Exact algorithms for the two-dimensional strip packing problem with and without rotations. *European Journal of Operational Research*, 198:73–83.

A. Lodi, S. Martello e D. Vigo. (2004), Models and bounds for two-dimensional level packing problems. *Journal of Combinatorial Optimization*, 8:363–379.

S. Martello, M. Monaci e D. Vigo. (2003), An exact approach to the strip-packing problem. *INFORMS Journal on Computing*, 15(3):310–319.

M. Mongeau e C. Bès. (2003), Optimization of aircraft container loading. *IEEE Transactions on Aerospace and Electronic Systems*, 39:140–150.

N. Ntene e J.H. van Vuuren. (2009), A survey and comparison of guillotine heuristics for the 2d oriented offline strip packing problem. *Discrete Optimization*, 6:174–188.

F. G. Ortmann, N. Ntene e J. H. van Vuuren. (2010), New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems. *European Journal of Operational Research*, 203: 306–315.

M. C. Riff, X. Bonnaire e B. Neveu. (2009), A revision of recent approaches for two-dimensional strip-packing problems. *Engineering Applications of Artificial Intelligence*, 22:823–827.

D. F. Zhang, S. D. Chen e Y. J. Liu. (2007), An improved heuristic recursive strategy based on genetic algorithm for the strip rectangular packing problem. *Acta Automatica Sinica*, 33:911–916.