Algoritmos Híbridos para Problemas de Corte Unidimensional

Glauber Ferreira Cintra

Instituto de Matemática e Estatística — Universidade de São Paulo
Rua do Matão, 1010 — CEP 05508-900 — São Paulo, SP
E-mail: glauber@ime.usp.br

Resumo. Mencionamos os principais resultados apresentados na dissertação de mestrado[1] de Glauber Ferreira Cintra[10], desenvolvida sob a orientação da Professora Dra. Yoshiko Wakabayashi. Inicialmente apresentamos uma visão geral dos problemas de corte e empacotamento, analisando suas principais características, a partir das quais introduzimos a classificação proposta por Dickhoff. Discutimos brevemente as principais estratégias utilizadas na resolução destes problemas, citando algumas referências para o leitor interessado neste tópico.

Investigamos o problema de corte de estoque unidimensional (PCE_1), formulando-o como um problema de programação linear inteira, e propomos um algoritmo híbrido, baseado no método de geração de colunas e num algoritmo exato. O algoritmo exato proposto é adequado para resolver instâncias do PCE_1 quando se conhece previamente um limitante inferior para o valor da solução inteira ótima. Mostramos ainda que o algoritmo híbrido proposto encontra uma solução inteira cujo valor objetivo difere do valor objetivo ótimo de no máximo 1, se a conjectura MIRUP (Modified Integer Round-Up Property) for verdadeira. Variações são propostas no algoritmo híbrido de modo a diminuir o tempo gasto na resolução dos problemas. Adaptamos ainda o algoritmo híbrido para o PCE_1 no qual a quantidade de itens distintos nos padrões é limitada por uma constante.

Os resultados obtidos ao resolver um expressivo número de instâncias práticas e instâncias aleatórias são analisados, indicando um desempenho bastante satisfatório do algoritmo híbrido e suas variações.

Abstract. We mention the main results presented in the master's thesis of Glauber Ferreira Cintra[10], developed under the supervision of Professor Dr. Yoshiko Wakabayashi. Initially we present an overview of cutting and packing problems, analyzing their main features and then we introduce the tipology proposed by Dickhoff. We briefly discuss the main strategies used in the resolution of these problems, mentioning some references for the reader interested in this topic.

We investigate the unidimensional cutting stock problem (CSP_1), modelled as an integer linear program, and propose a hybrid algorithm that is based in the column generation method and in an exact algorithm. The exact algorithm we use is appropriate to solve instances of the CSP_1 when we know previously a lower bound for the value of the optimal integer solution. We show that the proposed hybrid algorithm finds an integer solution whose objective value differs from the optimal value by at most 1, under the assumption that the MIRUP (Modified Integer Round-Up Property) conjecture is true. Variations are proposed for the hybrid algorithm in order to speed up its runtime. We adapt the hybrid algorithm for a special case of the CSP_1 in which the amount of distinct itens in the patterns is limited by a constant.

The results obtained in solving an expressive number of real world instances and randomly generated instances indicate that the hybrid algorithm and its variations have a very satisfactory performance.

Introdução

Nas últimas quatro décadas, os problemas de corte e empacotamento têm sido largamente estudados por um número crescente de pesquisadores, gerando e tendo se beneficiado de avanços significativos em diversas áreas, tais como *programação linear*, *programação dinâmica*, *teoria da complexidade computacional* e *algoritmos de aproximação*. O interesse por estes problemas é em parte explicado por sua aparente simplicidade e grande aplicabilidade prática. São porém, problemas de natureza complexa, *$\mathcal{NP}$-difíceis* [24]. A pesquisa sobre este assunto deu origem a diversos modelos matemáticos e o desenvolvimento de variadas técnicas para lidar com a intratabilidade destes problemas.

Quanto à aplicabilidade desses problemas, especialmente nas indústrias, podemos citar os processos de corte de barras de aço e alumínio, bobinas de papel, chapas de metal e madeira, lâminas de

[1] Pode-se obter cópia da dissertação no endereço http://www.ime.usp.br/dcc/posgrad/teses/glauber.ps.gz

vidro, peças de carpete, blocos de isopor, etc. Já os processos de carregamento de contêineres de navio, carrocerias de caminhões e vagões de trem são alguns exemplos práticos de problemas de empacotamento. Pequenas melhorias nestes processos podem levar a ganhos substanciais, dependendo da escala de produção, e representar uma vantagem decisiva na competição com outras empresas do setor.

Este artigo está organizado da seguinte maneira. Na seção 1 introduzimos a classificação proposta por Dickhoff. Na seção 2 citamos as principais estratégias propostas para resolver problemas de corte e empacotamento. Na Seção 3 apresentamos os algoritmos híbridos e outros algoritmos relacionados. Na Seção 4 discutimos os resultados de testes computacionais. Na seção 5 introduzimos uma nova variante para o PCE_1. Para finalizar, na última seção tecemos algumas considerações a respeito da dissertação. Devido a limitação de espaço, omitimos as demonstrações dos teoremas, explicamos superficialmente os algoritmos e comentamos brevemente as tabelas e figuras.

1 Classificação de Dyckhoff

Dyckhoff [19] propôs um esquema que permite integrar os diversos tipos de problemas de corte e empacotamento de maneira consistente e sistemática, estabelecendo uma classificação abrangente. Tal esquema define classes de problemas combinando-se os tipos principais de quatro características básicas. Estas características assim como seus tipos principais, denotados por seus respectivos símbolos, são:

1. Dimensionalidade:
   - (1) Unidimensional
   - (2) Bidimensional
   - (3) Tridimensional
   - (N) N-dimensional ($N > 3$)
2. Tipo de alocação:
   - (B) Todos os objetos e uma parte dos itens
   - (V) Uma parte dos objetos e todos os itens
3. Sortimento dos objetos:
   - (O) Um objeto
   - (I) Objetos de figuras idênticas
   - (D) Objetos de diferentes figuras
4. Sortimento dos itens:
   - (F) Poucos itens (ou figuras diferentes)
   - (M) Muitos itens de muitas figuras diferentes
   - (R) Muitos itens de relativamente poucas figuras distintas
   - (C) Figuras congruentes (mesma forma e tamanho)

Cada tipo de problema de corte e empacotamento é definido como sendo uma quádrupla $\alpha/\beta/\gamma/\delta$, onde α é a dimensionalidade, β o tipo de alocação, γ o sortimento dos objetos e δ o sortimento dos itens. É possível obter classes mais abrangentes deixando de especificar parte dos símbolos da quádrupla, indicando que as características não especificadas podem ser de qualquer um dos tipos principais. Na Tabela 1 listamos os principais problemas de corte e empacotamento abordados na literatura, indicando a classe a que pertencem. Pela tabela podemos perceber que dentro de uma mesma classe podemos encontrar diversos problemas que se diferenciam por outras características além das quatro principais adotadas na classificação.

2 Estratégias de Resolução

Várias abordagens têm sido propostas para se resolver as diversas variantes do problema de corte e empacotamento, sendo vasta a literatura a este respeito. Diversos artigos de revisão bibliográfica sobre o assunto têm sido escritos nos últimos anos. Recomendamos ao leitor interessado os artigos de Dyckhoff et al. [20], Sweeny e Paternoster [57], Dowsland [18], Coffman, Garey e Johnson [13], Hinxman [33] e Golden [30].

Problema	Classe
Mochila clássico	1/B/O/
Mochila multidimensional	/B/O/
Carregamento de páletes	2/B/O/C
Carregamento de veículos	1/V/I/F ou 1/V/I/M
Carregamento de contêineres	3/V/I/ ou 3/B/O/
Bin packing clássico	1/V/I/M
Bin packing dual	1/B/O/M
Bin packing bidimensional	2/V/D/M
Empacotamento em faixa	2/V/O/M
Empacotamento em altura	3/V/O/M
Corte de estoque unidimensional	1/V/I/R
Corte de estoque bidimensional	2/V/I/R
Corte de estoque generalizado	1///, 2/// ou 3///
Alocação de tarefas em multiprocessador	1/V/I/M
Alocação de memória	1/V/I/M
Planejamento de investimentos multiperiódicos	N/B/O/

Tabela 1: Problemas comuns na literatura e sua classificação.

Podemos dividir as abordagens em três categorias: algoritmos exatos, algoritmos de aproximação e métodos heurísticos. Algoritmos exatos conduzem à uma solução ótima. Porém os algoritmos exatos conhecidos têm complexidade de tempo exponencial, sendo portanto aplicáveis apenas a problemas de pequeno e médio porte. Algoritmos de aproximação encontram em tempo polinomial uma solução que se aproxima da solução ótima e possuem uma *garantia de desempenho*. Tal garantia, no caso de desempenho absoluto, é uma constante α tal que, no caso de problemas de minimização, a razão entre o valor da solução encontrada pelo algoritmo e o valor de uma solução ótima, para qualquer instância do problema, é no máximo α. Os métodos heurísticos se propõem a achar uma solução "boa" em tempo "aceitável", geralmente polinomial, mas não proporcionam uma garantia para a qualidade da solução fornecida em relação à solução ótima. Uma tendência recente é atacar o problema de corte e empacotamento usando algoritmos híbridos, que combinam algoritmos exatos com esquemas de aproximação e métodos heurísticos.

Na Tabela 2 listamos um expressivo número de algoritmos encontrados na literatura para os problemas de corte e empacotamento. Indicamos os autores que propuseram ou abordaram os algoritmos e listamos referências bibliográficas onde o leitor poderá obter detalhes dos algoritmos e as classes de problemas para as quais o algoritmo é indicado, seguindo a classificação de Dyckhoff.

3 Algoritmos Híbridos para o PCE_1

O PCE_1 consiste em: dado um objeto, genericamente denominado de *barra*, de comprimento L, e uma lista de m itens, cada item i com comprimento l_i e demanda $d_i \in I\!N$ $(i = 1, \ldots, m)$, determinar o menor número de barras necessário para atender a demanda. Obviamente também estamos interessados em determinar como as barras devem ser cortadas. Este problema pertence à classe 1/V/I/R segundo a classificação de Dyckhoff [19].

Como é bem conhecido, o PCE_1 pode ser formulado como um problema de programação linear inteira (PLI). Descrevemos a seguir como fazer isso. Primeiramente, consideramos cada possível forma de cortar uma barra, chamado de *padrão de corte* (ou simplesmente *padrão*), e o representamos por um vetor-coluna. Supondo que haja m itens, cada padrão j é representado por um vetor-coluna a_j, com m elementos, cujo i-ésimo elemento indica o número de vezes que o item i ocorre nesse padrão. Dizemos que um padrão é *viável* se $\sum_{i=1}^{m} (a_j)_i l_i \leq L$. O problema agora consiste em considerar os padrões viáveis

e decidir quantas vezes cada padrão deve ser utilizado de modo a atender a demanda, minimizando o total de barras utilizadas.

Autores (algoritmo)	Ano	Referências	Classes
Algoritmos Exatos			
Herz	1972	[32]	2/B/O/M
Diegel (TRIMOPT)	1988	[17]	1/V/I/R
Martello e Toth (MTP)	1990	[41, 53]	1/V/I/M
Ashford e Daniel	1992	[2]	1/V/I/R
Vance *et al.*	1994	[59]	1/V/I/M
Arenales e Morábito	1995	[1]	2/V/I/R
Scheithauer e Terno	1995	[49]	1/V/I/R
Vanderbeck	1996	[60]	1/V/I/M
Scholl, Klein e Jürgens (BISON)	1996	[52]	1/V/I/M
Vance	1998	[58]	1/V/I/R
Algoritmos de Aproximação			
Johnson, Csirik e Simchi-Levi (FFD)	1973	[34, 54]	1/V/I/M
Johnson *et al.* (NF e FF)	1974	[35]	1/V/I/M
Coffman *et al.* (FFDH, NFDH e SF)	1980	[14]	2/V/O/M
Baker, Coffman e Rivest (BLDW)	1980	[13]	2/V/O/M
Fernandez de la Vega e Lueker	1981	[22]	1/V/I/M
Baker, Brown e Katseff (UD)	1981	[3]	2/V/O/M
Karmakar e Karp	1982	[36]	1/V/I/M
Chung, Garey e Johnson (HFF)	1982	[8]	2/V/I/M
Coffman *et al.* (MFFD e BFB)	1984	[13, 54]	1/V/I/M
Lee e Lee (H_m)	1985	[37]	1/V/I/M
Li e Cheng (FF^2 e $FFLSr, s$)	1990	[38]	2/V/I/M e 3/V/I/M
Schiermeyer e Steinberg (M,S)	1994	[51, 56]	2/V/O/M
Miyazawa e Wakabayashi ($\mathcal{A}_k$)	1994	[43]	3/V/O/M
Csirik e Wöginger (Shelf($H_{m,p}$))	1997	[16]	2/V/O/M
Miyazawa	1997	[42]	2/V// e 3/V//
Métodos Heurísticos			
Gilmore e Gomory	1961	[28, 29]	/V/I/R
Pierce (RPE-Technique)	1964	[47]	/V/I
Christofides e Whitlock	1977	[7]	2/B/I/R
Heicken e König	1980	[31]	/V/I/R
Wang	1983	[61]	2/B/I/R
Stadtler	1990	[55]	1/V/I/R
Oliveira e Ferreira	1990	[46]	2/B/I/R
Morábito e Arenales	1995	[44]	2/V/I/R
Wäscher e Gau	1996	[63]	1/V/I/R
Falkenauer	1996	[21]	1/V/I/M
Bortfeldt e Gehring	1997	[5]	1/V/I/R

Tabela 2: Alguns algoritmos encontrados na literatura e sua aplicação.

Para isso, introduzimos uma variável x cujos elementos são inteiros x_j ($j = 1, \ldots, n$), que indicam quantas vezes o padrão j deve ser cortado. Note que se x é uma solução viável do problema, então o valor $\sum_{j=1}^{n} x_j$ corresponde ao número de barras a serem cortadas. Assim, denotando por A a matriz $m \times n$ cujas colunas são os vetores $a_1, \ldots, a_n$, e representando por d o vetor das demandas, o problema pode ser formulado como:

$$\begin{aligned} &\min \textstyle\sum_{j=1}^{n} x_j \\ &\text{sujeito a } Ax = d \qquad x_j \geq 0 \text{ e inteiro} \qquad j = 1, \ldots, n. \end{aligned} \tag{1}$$

3.1 Geração de Colunas

A formulação (1) traz consigo duas dificuldades em termos computacionais. A primeira é determinar a matriz A (que pode ter um número exponencial de colunas); a segunda é resolver um problema de programação linear inteira (que é sabido ser $\mathcal{NP}$-difícil). Para lidar com estas dificuldades, Gilmore e Gomory propuseram o método de geração de colunas [28, 29]. A idéia do método de geração de colunas consiste, como o próprio nome sugere, em ir gerando gradativamente as colunas da matriz A (dos padrões viáveis). Iniciamos com a matriz A correspondendo a uma solução básica viável. Com essa matriz A, que chamaremos de *base*, resolvemos a relaxação linear de (1) usando o algoritmo *simplex revisado* [9], onde a cada iteração uma nova coluna é gerada resolvendo-se um problema da mochila. Este algoritmo é conhecido como *simplex revisado com geração de colunas*, que chamaremos simplesmente de SimplexGC, descrito a seguir.

Algoritmo SimplexGC

Entrada: $(L, l_1, \ldots, l_m, d_1, \ldots, d_m)$.
Saída: Uma solução ótima da relaxação linear de (1).

1 Para $i = 1$ até m faça
  1.1 Para $j = 1$ até m se $i = j$ então faça $a_{ij} = \lfloor \frac{L}{l_i} \rfloor$; caso contrário, faça $a_{ij} = 0$.
2 Faça $x_i = \frac{d_i}{a_{ii}}$ $(i = 1, \ldots, m)$.
3 Resolva $yA = c$.
4 Execute o algoritmo MOCHILA com parâmetros $L, l_1, \ldots, l_m, y_1, \ldots, y_m$.
5 Se $\sum_{i=1}^{m} y_i z_i \leq 1$, retorne x e pare.
6 Caso contrário, resolva $Ab = z$.
7 Calcule $t = min\{\frac{x_i}{b_i} \mid b_i > 0 \ (i = 1, \ldots, m)\}$ e $s = min\{i \mid \frac{x_i}{b_i} = t \ (i = 1, \ldots, m)\}$.
8 Para $i = 1$ até m faça $a_{is} = z_i$ $(i = 1, \ldots, m)$.
  8.1 Se $i = s$ então faça $x_i = t$; caso contrário, faça $x_i = x_i - b_i t$.
9 Retorne ao passo 3.

Eis a descrição do algoritmo MOCHILA, usado no algoritmo SimplexGC.

Algoritmo MOCHILA

Entrada: $(L, l_1, \ldots, l_m, y_1, \ldots, y_m)$.
Saída: $z_1, \ldots, z_m \in \mathbb{N}$ tais que $\sum_{i=1}^{m} l_i z_i \leq L$ e $\sum_{i=1}^{m} y_i z_i$ é máximo.

1 Classifique os itens em ordem decrescente de custo relativo $(\frac{y_i}{l_i})$.
2 Faça $z_j = \lfloor (L - \sum_{i=1}^{j-1} l_i z_i)/l_j \rfloor$ para $j = 1, \ldots, m$, $z^* = z$ e $M = \sum_{i=1}^{m} y_i z_i^*$.
3 Faça $k = max\{i \mid z_i > 0 \text{ e } \sum_{j=1}^{i} y_j \bar{z}_j + \frac{y_{i+1}}{l_{i+1}}(L - \sum_{j=1}^{i} l_j \bar{z}_j) > M \ (i = m-1, \ldots, 1)\}$.
  3.1 Se não existe tal k então devolva z^* e pare.
  3.2 Caso contrário faça $z_k = z_k - 1$ e $z_j = \lfloor (L - \sum_{i=1}^{j-1} l_i z_i)/l_j \rfloor$, para $j = k+1, \ldots, m$.
4 Se $M \leq \sum_{i=1}^{m} y_i z_i$ faça $z^* = z$ e $M = \sum_{i=1}^{m} y_i z_i^*$.
5 Retorne ao passo 3.

O SimplexGC fornece uma solução ótima, não necessariamente inteira. Para contornar este problema, Gilmore e Gomory [28] propuseram arredondar para cima o valor das variáveis, após achar uma solução ótima, obtendo assim uma solução inteira. No entanto, este procedimento usualmente acarreta a produção de itens em quantidade superior à demanda, e conduz a uma solução inteira eventualmente longe da ótima. Na subseção seguinte descrevemos como, a partir da solução fornecida pelo SimplexGC, podemos obter uma solução inteira que se aproxime bem mais da solução inteira ótima do que se apenas arredondarmos a solução fracionária para cima.

3.2 Obtendo uma Solução Inteira

Aplicando o SimplexGC obtemos uma solução ótima x para a relaxação linear de (1). Chamemos de v_{rl} o valor desta solução. O método proposto para encontrar uma solução inteira de (1) consiste em arredondar x para baixo obtendo $\bar{x}$, ou seja, $\bar{x}_j = \lfloor x_j \rfloor$ $(j = 1, \ldots, m)$. Seja $\bar{v}$ o valor da solução $\bar{x}$. Claramente $\bar{v} \leq v_{rl}$. Se $\bar{v} = v_{rl}$, então x é uma solução inteira ótima, caso contrário, alguma parte da demanda não foi atendida. Temos assim um problema residual onde cada item i possui demanda inteira $d'_i = d_i - \sum_{j=1}^{m} a_{ij}\bar{x}_j$ $(i = 1, \ldots, m)$. Após calcular $\bar{x}$ e d', se $\bar{v} > 0$ então aplicamos recursivamente o algoritmo ao problema residual; caso contrário, resolvemos o problema residual utilizando um algoritmo que forneça uma solução inteira, como o *First Fit Decreasing (FFD)*, ou o *First Fit Decreasing especializado (FFDe)* ou o *FFDWB (First Fit Decreasing With Backtracking)*.

O algoritmo FFD é bastante simples e apresenta um desempenho bastante satisfatório. Em 1973, Johnson[34] provou que o algoritmo FFD tem limite de desempenho assintótico $\frac{11}{9}$. Mais precisamente, denotando por $FFD(I)$ o valor da solução encontrada pelo algoritmo FFD e por $OPT(I)$ o valor de uma solução ótima para uma instância I, então para toda instância I, $FFD(I) \leq \frac{11}{9}OPT(I) + 4$. Sabe-se também que o FFD apresenta desempenho empírico no caso médio de 1,02 (*cf.* Bramel *et al.* [6]) e pode ser implementado de forma a ter complexidade de tempo $\mathcal{O}(n \log n)$. Descrevemos a seguir o algoritmo FFD, adotando as seguintes notações: $\wp_i$ representa o padrão de corte i e $c(\wp_i)$ é uma função que retorna o comprimento da barra não utilizado pelo padrão $\wp_i$.

Algoritmo FFD

Entrada: $(L, l_1, \ldots, l_n)$.
Saída: Uma solução inteira $\wp_1, \ldots, \wp_k$ que atende a demanda.

1 Classifique $l_1, \ldots, l_n$ em ordem decrescente e faça $k = 1$ e $\wp_k = \emptyset$.
2 Para $i = 1$ até n procure $j = min\{h \mid c(\wp_h) \geq l_i \ (h = 1, \ldots, k)\}$
  2.1 Se existir tal j então empacote o item i no padrão $\wp_j$, ou seja, faça $\wp_j = \wp_j \cup \{i\}$.
  2.2 Caso contrário, faça $k = k + 1$ e empacote o item i em $\wp_k$ fazendo $\wp_k = \{i\}$.
3 Retorne $\wp_1, \ldots, \wp_k$ e pare.

Apresentamos a seguir uma variação do algoritmo FFD, que chamaremos de *FFD especializado*, denotado por FFDe. Na descrição do algoritmo FFDe, denotamos por $f(\wp_k)$ a função que retorna a frequência do item i em $\wp_k$.

Algoritmo FFDe

Entrada: $(L, l_1, \ldots, l_m, d_1, \ldots, d_m)$.
Saída: Uma solução inteira que atende a demanda.

1 Classifique $l_1, \ldots, l_m$ em ordem decrescente e faça $k = 1$ e $\wp_k = \emptyset$.
2 Repita até que $d_i = 0$ $(i = 1, \ldots, m)$.
  2.1 Procure $j = min\{h \mid l_h \leq c(\wp_k) \ (h = 1, \ldots, m)\}$.
  2.2 Se existir tal j então faça $f = min(d_j, \lfloor \frac{c(\wp_k)}{l_j} \rfloor)$ e empacote f vezes o item j em $\wp_k$, fazendo f vezes $\wp_j = \wp_j \cup \{i\}$.
  2.3 Caso contrário, faça $r_k = min\{\lfloor \frac{d_i}{f_i(\wp_k)} \rfloor, \ i \in \wp_k\}$.
    2.3.1 Para todo $i \in \wp_k$ faça $d_i = d_i - f_i(\wp_k)r_k$. Faça $k = k + 1$ e $\wp_k = \emptyset$.
3 Retorne $\wp_1, \ldots, \wp_k$ e $r_1, \ldots, r_k$ e pare.

Esta versão que propomos é especialmente apropriada por requerer menor tempo de execução e por permitir que a implementação do algoritmo seja feita utilizando-se estruturas de dados mais simples. Os seguintes resultados podem ser provados a respeito dos algoritmos FFD e FFDe.

Teorema 3.1. *A solução do algoritmo FFDe é equivalente à solução do algoritmo FFD.*

Teorema 3.2. *Para uma dada lista que possui m itens de comprimentos distintos, ambos os algoritmos FFDe e FFD encontram uma solução com no máximo 2m padrões distintos.*

Apesar de ser empiricamente bom no caso médio, Simchi-Levi [54] mostrou que o limite de desempenho absoluto do FFD é 1,5 no pior caso; e que este limite é justo a não ser que $\mathcal{P} = \mathcal{NP}$. Foi também observado que o FFD encontra dificuldade, no que diz respeito à qualidade da solução, em instâncias pequenas ou quando os itens têm comprimento de aproximadamente $\frac{1}{3}L, \frac{1}{4}L, \frac{1}{5}L, \ldots$, onde L é o tamanho da barra (*cf.* Schwerin e Wäscher [53]) ou ainda nas instâncias que Falkenauer [21] chamou de *triplets*. Apresentamos a seguir o algoritmo *First Fit Decreasing with Backtracking (FFDWB)*, baseado no algoritmo FFDe, que encontra uma solução inteira ótima.

Na Subseção 3.1 vimos que o SimplexGC fornece uma solução ótima para a relaxação linear de (1) cujo valor chamaremos de v_{rl}. Podemos então usar $v_{ip} = \lceil v_{rl} \rceil$ como um limite inferior para o valor de qualquer solução inteira ótima. Como veremos na Subseção 3.3, este limitante é justo na grande maioria dos problemas. O algoritmo FFDWB utiliza este limitante para encontrar mais rapidamente soluções inteiras ótimas para o PCE_1.

A idéia básica do algoritmo FFDWB, que propomos, é aplicar um procedimento semelhante ao algoritmo FFDe para gerar um padrão de cada vez. O desperdício deste padrão é então comparado com o desperdício de uma solução cujo valor é v_{ip}, da forma que explicitaremos mais à frente. Se o padrão gerado não for promissor executamos um retrocesso, diminuindo a frequência dos itens dentro do padrão, do último item que foi empacotado até o primeiro, e aplicamos recursivamente o procedimento na parte da barra não utilizada pelo padrão.

O algoritmo FFDWB, utiliza como subrotina o procedimento PACKING. Este procedimento recebe como parâmetros de entrada um valor D_k, que representa o comprimento que pode ser desperdiçado numa solução que utiliza C barras depois de gerados os padrões $\wp_1, \ldots, \wp_{k-1}$; um valor k, representando o índice do padrão corrente; e ainda os valores f e i, onde f representa quantas vezes o item de comprimento l_i deve ser empacotado no padrão corrente. No procedimento assumimos como constantes globais o valor C, que representa a quantidade máxima de barras que podem ser utilizadas na solução, os comprimentos $l_1, \ldots, l_m$ dos itens e suas demandas $d_1, \ldots, d_m$. O procedimento PACKING com parâmetros D_k, k, f e i, encontra, se existir, uma solução que utiliza no máximo $C - k + 1$ barras onde a primeira barra tem comprimento $c(\wp_k)$ e as demais barras têm comprimento L. Em [11] mostramos como especializar o algoritmo MOCHILA obtendo o algoritmo MOCHILAe, usado no procedimento PACKING. O algoritmo FFDWB e o procedimento PACKING são descritos a seguir.

Algoritmo FFDWB

Entrada: $(L, l_1, \ldots, l_m, d_1, \ldots, d_m, C)$.
Saída: Uma solução inteira de valor no máximo C, se existir; caso contrário, o valor 0.

1 Classifique os itens em ordem não decrescente de $min(d_i, \lfloor \frac{L}{l_i} \rfloor)$ $(i = 1, \ldots, m)$.
2 Faça $D = CL - \sum_{i=1}^{m} l_i d_i$ e $f = min(d_1, \lfloor \frac{L}{l_1} \rfloor)$.
3 Para $f' = f$ até 0 execute PACKING$(D, 1, f', 1)$.
4 Retorne 0 e pare.

Procedimento PACKING(D_k, k, f, i)

1 Empacote f vezes o item i em $\wp_k$, fazendo f vezes $\wp_k = \wp_k \cup \{i\}$, e faça $d_i = d_i - f$.
2 Procure $j = min\{h \mid d_h > 0 \ (h = 1, \ldots, m)\}$. Se não existir tal j retorne $\wp_1, \ldots, \wp_k$ e pare.
3 Procure $i' = min\{h \mid d_h > 0 \text{ e } l_h \leq c(\wp_k) \ (h = i+1, \ldots, m)\}$.
  3.1 Se existir i' então faça $f = min(d_{i'}, \lfloor \frac{c(\wp_k)}{l_{i'}} \rfloor)$ e vá para o passo 5.
  3.2 Execute o algoritmo MOCHILAe com parâmetros $L, l_1, \ldots, l_m, l_1, \ldots, l_m, d_1, \ldots, d_m$.
  3.3 Faça $t = L - \sum_{i=1}^{m} l_i z_i$.
  3.4 Se $k < C$ e $c(\wp_k) \leq \lfloor \frac{D_k}{C-k+1} \rfloor$ e $t \leq \lfloor \frac{D_k - c(\wp_k)}{C-k} \rfloor$ então execute
    PACKING$(D_k - c(\wp_k), k+1, min(d_j, \lfloor \frac{L}{l_j} \rfloor), j)$.
4 Faça $f = -1$. {Para que o laço no passo seguinte não seja efetuado}
5 Para $f' = f$ até 0 execute PACKING(D_k, k, f', i'), faça $d_{i'} = d_{i'} + f'$ e remova de $\wp_k$ as f' ocorrências do item i', fazendo f vezes $\wp_k = \wp_k - \{i'\}$.

Basicamente, o algoritmo FFDWB enumera todos os padrões viáveis, buscando uma coleção de padrões de cardinalidade no máximo C que atenda toda a demanda. O algoritmo executa uma busca em profundidade numa árvore, onde cada nó da árvore representa um padrão, e a solução encontrada é um ramo da árvore, desde a raiz até uma folha. Esta árvore tem altura no máximo C. Utilizando as desigualdades (8) e (9), podamos a árvore, diminuindo bastante o espaço de busca. Note que, ao percorrer um ramo da árvore com profundidade k, temos que $D_1 \leq \ldots \leq D_k$ (não necessariamente $c(\wp_1) \leq \ldots \leq c(\wp_k)$). Isto significa que o FFDWB prefere gerar inicialmente os padrões que apresentam pequeno desperdício; com isso, os primeiros níveis da árvore possuem poucas ramificações, quando comparados com os níveis posteriores. Omitimos aqui a prova de que o algoritmo FFDWB com as entradas especificadas, encontra de fato, se existir, um empacotamento para os itens de S que utiliza no máximo C barras de comprimento L. Tal prova, assim como uma explicação detalhada do algoritmo HÍBRIDO, descrito adiante, pode ser encontrada em [12].

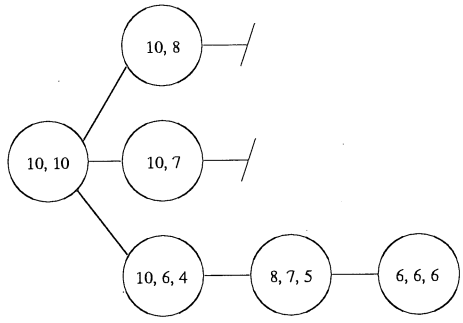

Figura 1: Árvore percorrida pelo algoritmo FFDWB ao resolver a instância caracterizada por $L = 20$, $l = \{10, 8, 7, 6, 5, 4\}$, $d = \{3, 1, 1, 4, 1, 1\}$ e $C = 4$.

A Figura 1 mostra a árvore percorrida pelo algoritmo FFDWB ao resolver a instância: $L = 20$, $l = \{10, 8, 7, 6, 5, 4\}$, $d = \{3, 1, 1, 4, 1, 1\}$ e $C = 4$. Cada nó da árvore representa um padrão. Observe que os primeiros dois ramos da árvore (olhando de cima para baixo) foram podados no segundo nível, pois os dois primeiros padrões gerados neste nível apresentam desperdício de 2 e 3, enquanto que o passo 4 do procedimento PACKING impõe que o segundo padrão possua desperdício zero ($c(\wp_2) \leq \lfloor \frac{D_2}{C-2+1} \rfloor = 0$). A solução encontrada pelo FFDWB é representada pelo terceiro ramo da árvore, e utiliza 4 barras. É interessante notar que o algoritmo FFD aplicado a esta mesma instância encontra uma solução que utiliza 5 barras.

Descrevemos a seguir o algoritmo híbrido, doravante referido como algoritmo HÍBRIDO.

Algoritmo HÍBRIDO

Entrada: $(L, l_1, \ldots, l_m, d_1, \ldots, d_m)$
Saída: Uma solução do problema (1).

1 Faça $v_{rl} = 0$ e $v_{ip} = 0$. (v_{rl} é o valor da solução ótima do problema relaxado; e v_{ip} representa o valor da solução inteira corrente)
2 Execute o algoritmo SimplexGC com parâmetros $L, l_1, \ldots, l_m, d_1, \ldots, d_m$.
Se $\sum_{i=1}^{m} x_i > v_{rl}$ então faça $v_{rl} = \sum_{i=1}^{m} x_i$.
3 Para $i = 1$ até m faça $x_i^* = \lfloor x_i \rfloor$. Faça $v_{ip} = v_{ip} + \sum_{i=1}^{m} x_i^*$
4 Se $x_i^* > 0$ para algum $i = 1, ..., m$ então
4.1 Retorne A e $x_1^*, \ldots, x_m^*$. Faça $m' = 0$.
4.2 Para $i = 1$ até m faça
4.2.1 Para $j = 1$ até m faça $d_i = d_i - a_{ij} x_j^*$.
4.3 Para $i = 1$ até m faça
4.3.1 Se $d_i > 0$ faça $m' = m' + 1$, $l_{m'} = l_i$ e $d_{m'} = d_i$.
4.4 Se $m' = 0$ então pare.
4.5 Faça $m = m'$ e retorne ao passo 2.
5 Faça $C = \lceil v_{rl} \rceil - v_{ip}$. Execute o FFDWB com parâmetros $L, l_1, \ldots, l_m, d_1, \ldots, d_m, C$.
6 Se o FFDWB não retornou 0 então retorne $\wp_1, \ldots, \wp_C$ e pare.
7 Caso contrário, execute o FFDWB com parâmetros $L, l_1, \ldots, l_m, d_1, \ldots, d_m, C + 1$.
8 Se o FFDWB não retornou 0 então retorne $\wp_1, \ldots, \wp_{C+1}$ e pare.
9 Caso contrário, execute o FFDe com parâmetros $L, l_1, \ldots, l_m, d_1, \ldots, d_m$,
retorne a solução encontrada e pare.

3.3 A Conjectura MIRUP e o algoritmo HÍBRIDO

Seja I uma instância de um problema de programação linear inteira P, no qual queremos minimizar a função objetivo. Seja $v(I)$ o valor de uma solução inteira ótima de I e $v_{rl}(I)$ o valor de uma solução ótima da relaxação linear do problema. Baum e Trotter [4] definiram a *Integer Round-Up Property (IRUP)* da seguinte forma:

Definição 3.1. *Um problema de programação linear inteira P (de minimização) possui a integer round-up property (IRUP) se $v(I) = \lceil v_{rl}(I) \rceil$ para toda instância $I \in P$.*

Também dizemos que a IRUP é válida (ou se verifica) para uma instância I de um PLI se $v(I) = \lceil v_{rl}(I) \rceil$. Baum e Trotter [4] estabeleceram que esta propriedade é válida para certas classes de matrizes que surgem no contexto da teoria dos polimatróides. Em 1985, Marcotte [39] mostrou que várias classes de instâncias do PCE_1 possuem a IRUP. Dyer, Frieze e McDiarmid [40] provaram que é $\mathcal{NP}$-difícil determinar se uma instância qualquer do PCE_1 possui ou não a IRUP.

Em 1986 Marcotte [40] apresentou uma instância para a qual a IRUP não vale. Para esta instância I, $v(I) = \lceil v_{rl}(I) \rceil + 1$. Tal instância foi construída artificialmente a partir do problema da 4-PARTIÇÃO e apresenta coeficientes da ordem de 10^7, o que levou Marcotte a conjecturar que qualquer instância que não tivesse a IRUP deveria possuir coeficientes da ordem de 10^7, sendo portanto uma instância dissociada de problemas práticos. Fieldhouse [23], no entanto, apresentou uma instância bastante simples para a qual a IRUP não vale: $L = 30$, $l = \{15, 10, 6\}$ e $d = \{21, 32, 54\}$. Para esta instância I, temos $v_{rl}(I) = 31,9666\ldots$ e $v(I) = 33$.

Scheithauer e Terno [48], Gau [25], Schwerin e Wäscher [53] também exibiram instâncias com coeficientes pequenos, cuja diferença entre $v(I)$ e $v_{rl}(I)$ é maior que 1. Em nossos testes computacionais, também encontramos diversas instâncias cujo *gap* é maior que 1. Em [53] Schwerin e Wäscher apresen-

taram uma instância com 200 itens cujo *gap* entre o valor de uma solução inteira ótima e o valor de uma solução ótima da relaxação linear é 1,14435. Este é o maior *gap* conhecido até o momento. Estes resultados levaram Scheithauer e Terno [48] a definir a *Modified Integer Round-Up Property (MIRUP).*

Definição 3.2. *Um problema de programação linear inteira P (de minimização) possui a modified integer round-up property (MIRUP) se* $v(I) \leq \lceil v_{rl}(I) \rceil + 1$ *para toda instância* $I \in P$.

Analogamente, dizemos que a MIRUP é válida (ou se verifica) para uma instância I de um PLI se $v(I) \leq \lceil v_{rl}(I) \rceil + 1$. Em [48], Scheithuaer e Terno provaram que toda instância do PCE_1 (i.e., seus correspondentes PLI) com $m \leq 5$ possuem a MIRUP. Em experimentos computacionais realizados por diversos autores [50, 62], onde foram determinadas soluções inteiras ótimas, verificou-se que todas as instâncias testadas (incluindo as geradas aleatoriamente e aquelas mencionadas na literatura) apresentavam a MIRUP. Não se conhece até o momento nenhuma instância do PCE_1 que não possua a MIRUP. Em 1995, Scheithauer e Terno [50] introduziram a seguinte conjectura.

Conjectura MIRUP: *O* PCE_1 *possui a MIRUP.*

O seguinte resultado relaciona essa conjectura com o algoritmo HÍBRIDO.

Teorema 3.3. *Se a conjectura MIRUP for verdadeira, a diferença entre o valor da solução encontrada pelo algoritmo HÍBRIDO e o valor de uma solução inteira ótima é no máximo 1.*

Este teorema implica que o limite de desempenho assintótico do algoritmo HÍBRIDO é 1, se verdadeira conjectura MIRUP. Dizemos então que o algoritmo HÍBRIDO é um algoritmo *quase-exato.*

Podemos introduzir uma mudança na regra de arredondamento utilizada no passo 3 do algoritmo HÍBRIDO, objetivando diminuir o tamanho do problema residual. Como veremos na seção seguinte, esta modificação leva a um ganho *médio* considerável no tempo requerido para resolver o problema, sem que haja uma diminuição sensível na qualidade da solução. Chamaremos este algoritmo de *HÍBRIDOm*. A modificação consiste em primeiramente arredondar para baixo as variáveis cuja parte fracionária é menor ou igual a $\frac{1}{2}$. Depois arredondamos as variáveis ainda fracionárias, uma de cada vez, para cima, se isto não ocasionar produção de itens em excesso, ou para baixo, caso contrário.

O Teorema 3.3 não vale para o algoritmo HÍBRIDOm, ainda assim podemos estabelecer um limite para o valor $H'(I)$ da solução encontrada pelo algoritmo HÍBRIDOm para uma instância I. Usando o fato de que o limite de desempenho absoluto do FFD no pior caso é 1,5 (Simchi-Levi [54]), podemos concluir que vale o seguinte resultado.

Teorema 3.4. $H'(I) \leq OPT(I) + \frac{m}{2}$ *para toda instância* I *do* PCE_1.

4 Resultados Computacionais

Avaliamos os algoritmos híbridos propostos na seção anterior resolvendo 4000 instâncias geradas aleatoriamente e mais cerca de duas centenas de instâncias práticas tiradas das plantas de ferragem de obras de uma construtora. Os algoritmos híbridos foram implementados usando-se a linguagem C e os testes foram executados numa estação Sun Sparc 1000, clock de 50 mhz e 704 MB de memória principal. Utilizamos o CPLEX 2.0 [15] para resolver os sistemas de equações lineares que aparecem nos passos 3 e 6 do algoritmo SimplexGC.

4.1 Instâncias Aleatórias

Resolvemos 4000 instâncias geradas aleatoriamente utilizando o método delineado por Wäscher e Gau em [63] e que denotaremos por *instâncias de W & G.* Tais instâncias foram geradas usando o algoritmo *CUTGEN1*, descrito em [27]. A Tabela 3 apresenta os resultados da aplicação do algoritmo HÍBRIDO às instâncias de W & G. Nela estão indicados o número de itens (m), o tamanho dos itens em relação à

barra, o número de instâncias para as quais foi encontrada uma solução que satisfaz a IRUP (portanto, uma solução ótima), o número de instâncias para as quais foi encontrada uma solução que satisfaz a MIRUP mas não satisfaz a IRUP, o número médio de colunas geradas pelo SimplexGC em cada instância, o tempo médio requerido para resolver cada instância, o ganho médio percentual da solução encontrada pelo algoritmo HÍBRIDO em relação à solução encontrada pelo FFD e finalmente o desperdício médio em cada instância.

m	Tamanho dos itens	IRUP	MIRUP	Colunas geradas	Tempo (seg)	Ganho sobre FFD	Desperdício
10	0%-100%	200	0	7,21	0,04	0,345%	13,734%
	0%-75%	200	0	11,92	0,07	0,569%	15,084%
	0%-50%	200	0	23,77	0,14	2,727%	3,091%
	0%-25%	200	0	41,02	0,34	1,590%	1,388%
20	0%-100%	199	1	26,01	0,17	0,273%	9,873%
	0%-75%	199	1	47,37	0,32	0,641%	7,450%
	0%-50%	200	0	108,98	0,97	2,322%	0,695%
	0%-25%	200	0	105,60	1,24	0,864%	0,665%
30	0%-100%	199	1	56,70	0,45	0,225%	10,046%
	0%-75%	200	0	112,53	1,02	0,545%	6,376%
	0%-50%	198	2	293,29	3,87	1,887%	0,362%
	0%-25%	200	0	187,85	2,71	0,551%	0,451%
40	0%-100%	200	0	99,31	0,85	0,200%	9,901%
	0%-75%	198	2	226,31	2,66	0,633%	4,301%
	0%-50%	200	0	585,84	9,50	1,456%	0,206%
	0%-25%	200	0	289,55	4,14	0,435%	0,348%
50	0%-100%	200	0	159,15	1,55	0,227%	7,175%
	0%-75%	197	3	375,12	4,78	0,582%	4,399%
	0%-50%	198	2	969,48	20,28	1,157%	0,157%
	0%-25%	200	0	397,06	17,80	0,363%	0,275%

Tabela 3: O Algoritmo HÍBRIDO aplicado às instâncias de W & G.

Foram encontradas soluções inteiras ótimas para pelo menos 3988 instâncias e o tempo médio necessário para resolver cada instância foi bastante satisfatório. Resolvemos essas mesmas 4000 instâncias com o algoritmo HÍBRIDOm. Os resultados obtidos foram bastante semelhantes, com apenas uma diferença significativa: observamos que o tempo gasto pelo algoritmo HÍBRIDOm foi, de uma forma geral, 20% menor do que o tempo gasto pelo algoritmo HÍBRIDO.

A Tabela 4 compara os resultados obtidos pelos algoritmos híbridos aqui propostos e diversos métodos encontrados na literatura, quando aplicados às instâncias de W & G. Adotamos para estes métodos os nomes sugeridos por Wäscher e Gau [63] e indicamos as referências onde o leitor pode obter detalhes deste métodos. Dentre todos eles, o algoritmo HÍBRIDO foi o que encontrou soluções ótimas para o maior número de instâncias. É pertinente observar que os resultados obtidos por Wäscher e Gau em 1996 eram os melhores até então.

Os resultados dos testes com essas 4000 instâncias aleatórias vêm fortalecer a conjectura MIRUP. Além destas instâncias, resolvemos também *alguns milhões* de instâncias aleatórias, com m variando entre 10 e 20, com o objetivo de encontrar um contra-exemplo para a conjectura MIRUP. Não encontramos uma instância sequer com *gap* entre v_{ip} e v_{rl} maior que 1,14435.

Algoritmo	$m = 10$	$m = 20$	$m = 30$	$m = 40$	$m = 50$	Total
Total de instâncias	800	800	800	800	800	4000
HÍBRIDO	800	798	797	798	795	3988
HÍBRIDOm	798	796	796	790	791	3971
RSUC [25]	797	792	790	779	763	3921
ROPT [63]	796	788	782	770	738	3874
CSTAOPT [55]	758	754	731	732	734	3709
RFFD [63]	779	758	743	726	695	3701
CSTAFFD [63]	752	746	715	716	701	3630
FFD	469	378	359	321	318	1845
BOPT [63]	428	321	262	251	210	1472
BRURED [45]	325	193	129	94	80	821
BRUSUC [55]	257	142	74	53	52	578
BRUSIM [63]	93	47	23	15	10	188

Tabela 4: Soluções inteiras ótimas obtidas por diversos algoritmos aplicados às instâncias de W & G.

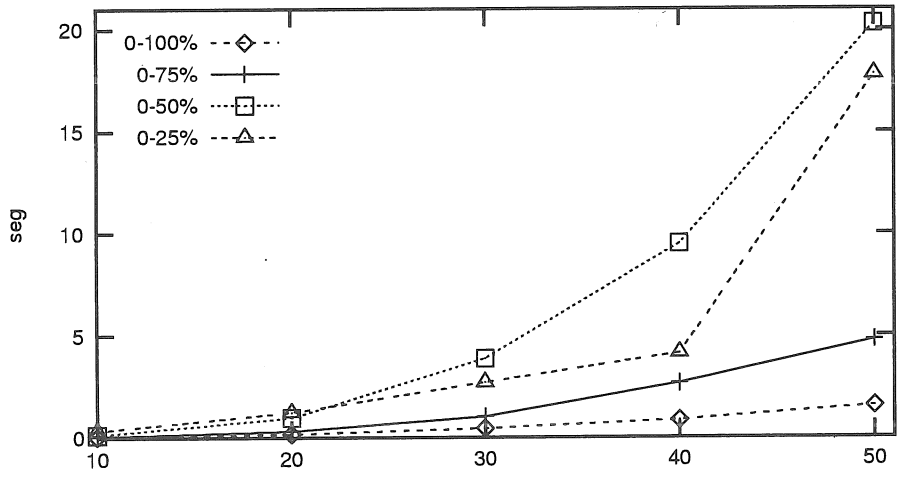

Figura 2: Tempo médio requerido para resolver as instâncias de W & G.

4.2 Instâncias Reais

Apresentamos nesta subseção os resultados obtidos ao aplicar o algoritmo HÍBRIDO a cerca de duas centenas de instâncias reais tiradas das plantas de ferragem de duas obras de uma construtora com atuação em várias cidades do país. Tratam-se de edifícios gêmeos de 9 andares, construídos lado a lado. As instâncias consideradas são relativas ao problema de determinar como cortar o aço a ser utilizado nas estruturas de concreto armado de modo a minimizar o desperdício de aço. Nestas obras foi utilizado aço CA-60B com 5mm de bitola, e aço CA-50A com bitolas 6.3mm, 8mm, 10mm, 12.5mm, 16mm e 20mm. Dessa forma houve necessidade de utilizar 7 tipos de barras de aço, originando 7 categorias de problemas. A partir das 42 plantas de ferragem analisadas, obtivemos 209 instâncias, que chamaremos de *instâncias práticas.*

A Tabela 5 mostra os resultados obtidos ao resolver as instâncias práticas usando o algoritmo HÍBRIDO. Foram encontradas, dentro de um tempo bastante curto, soluções ótimas para todas as instâncias. Ademais, o algoritmo HÍBRIDO obteve um ganho considerável em relação ao FFD.

No entanto, apesar de termos encontrado soluções ótimas para todas as instâncias, em algumas cate-

Bitola	Nº de problemas	m médio	Tamanho dos itens	IRUP	Colunas geradas	Tempo (seg)	Ganho sobre FFD	Desperdício
5.0	13	7	2%-50%	13	14,54	0,23	0,945%	0,324%
6.3	43	11	2%-98%	43	23,86	0,33	1,868%	1,438%
8.0	40	12	2%-90%	40	30,10	0,35	1,128%	6,281%
10.0	35	20	4%-92%	35	78,00	1,03	0,825%	7,843%
12.5	36	14	6%-92%	36	46,03	0,64	2,207%	5,467%
16.0	26	13	16%-75%	26	25,12	0,31	1,353%	20,550%
20.0	16	8	20%-74%	16	9,69	0,12	0,894%	12,023%

Tabela 5: Resultados obtidos pelo algoritmo HÍBRIDO aplicado às instâncias práticas.

gorias o desperdício médio foi bastante alto. Com o intuito de diminuir o desperdício, agrupamos todas as instâncias de cada categoria gerando apenas 7 instâncias, que chamaremos de *instâncias agrupadas*. Os resultados obtidos pelo algoritmo HÍBRIDO ao resolver as instâncias agrupadas são apresentados na Tabela 6.

Bitola	m	Tamanho dos itens	IRUP	Colunas geradas	Tempo (seg)	Ganho sobre FFD	Desperdício
5.0	48	2%-50%	Sim	118	41	0,607%	0,020%
6.3	209	2%-98%	Sim	622	37	0,474%	0,017%
8.0	264	2%-90%	Sim	2706	190	1,278%	0,009%
10.0	365	4%-92%	Sim	10758	1576	1,365%	0,061%
12.5	301	6%-92%	Sim	8333	782	1,481%	0,268%
16.0	218	16%-75%	Sim	1231	164	1,673%	4,995%
20.0	105	20%-74%	Sim	573	29	0,489%	2,507%

Tabela 6: Resultados obtidos pelo algoritmo HÍBRIDO aplicado às instâncias agrupadas.

Para ter uma idéia mais aproximada do economia, precisamos levar em conta o peso das barras de aço. A Tabela 7 apresenta a quantidade de aço especificada nas plantas e as quantidades utilizadas nas soluções apresentadas nas tabelas 5 e 6. Vale salientar que esta construtora desperdiça em média, em obras deste tipo, algo em torno de 10% do aço, e este percentual é bom se comparado ao desperdício médio verificado na indústria de construção civil. Agrupando as instâncias, reduzimos o desperdício de aço a cerca de 1%, o que representa uma economia considerável.

Bitola	Peso (kg/m)	Quantidade de aço nas plantas (kg)	Tabela 5		Tabela 6	
			Quantidade de aço (kg)	Desperdício	Quantidade de aço (kg)	Desperdício
5.0	0,16	5062,008	5078,400	0,324%	5063,040	0,020%
6.3	0,25	15834,380	16062,000	1,438%	15837,000	0,017%
8.0	0,40	24415,352	25948,800	6,281%	24417,600	0,009%
10.0	0,63	23799,497	25666,200	7,843%	23814,000	0,061%
12.5	1,00	24235,060	25560,000	5,467%	24300,000	0,268%
16.0	1,60	15305,872	18470,400	20,675%	16070,400	4,995%
20.0	2,50	17969,525	20130,000	12,023%	18420,000	2,507%
Totais		126621,695	136915,800	8,130%	127922,040	1,027%

Tabela 7: Desperdício de aço nas soluções encontradas para as instâncias práticas.

Os resultados obtidos, tanto nos testes realizados com instâncias geradas aleatoriamente, quanto nos testes com instâncias práticas, demonstraram a eficiência do algoritmo HÍBRIDO em termos de

qualidade da solução e tempo de execução. Ademais, os resultados destes testes serviram para fortalecer a conjectura MIRUP.

5 Uma Nova Variante do PCE_1

Visando conhecer de perto o processo de corte do aço utilizado nas estruturas de concreto armado, visitamos construtoras e canteiros de obra e conversamos com engenheiros, mestres-de-obra e ferreiros. Conhecendo mais a fundo o processo de corte manual do aço, percebemos não ser desejável que a solução do PCE_1 associado ao processo de corte apresente padrões com um grande número de itens distintos. No corte manual do aço, os padrões devem conter poucos itens diferentes, digamos no máximo 4 ou 5, pois do contrário os ajustes necessários no equipamento utilizado para o corte acabariam por tornar a solução desinteressante e, eventualmente, impraticável.

Dessa forma resolvemos estudar o PCE_1 impondo uma nova restrição: *o número de itens distintos que podem ocorrer num padrão viável é limitado por uma constante inteira* δ. É fácil perceber que esta restrição faz com que o número de padrões viáveis esteja limitado por $\lfloor \frac{L}{l_{min}} \rfloor^{\delta}$, onde L é o comprimento da barra e l_{min} o comprimento do menor item. Como desejamos que δ assuma valores pequenos (no máximo 5), esta restrição limita bastante o número de padrão viáveis. Podemos então esperar que, com esta restrição, os problemas possam ser resolvidos mais rapidamente, com o ônus de uma queda considerável na qualidade das soluções encontradas. Veremos no entanto que, empiricamente, estas duas expectativas não se confirmam. A seguir citamos os algoritmos que devem ser modificados de modo a incorporar esta nova restrição.

m	Tamanho dos itens	Tempo (seg)			Ganho sobre $FFDe_\delta$ (%)			Desperdício (%)		
		$\delta = m$	$\delta = 5$	$\delta = 4$	$\delta = m$	$\delta = 5$	$\delta = 4$	$\delta = m$	$\delta = 5$	$\delta = 4$
10	0-100%	0,04	0,02	0,04	0,345	0,345	0,345	13,734	13,734	13,734
	0-75%	0,05	0,06	0,07	0,565	0,565	0,565	15,089	15,089	15,089
	0-50%	0,13	0,10	0,17	2,721	2,714	2,701	3,098	3,104	3,118
	0-25%	0,32	0,33	0,39	1,590	1,550	1,351	1,388	1,428	1,628
20	0-100%	0,17	0,14	0,18	0,273	0,273	0,273	9,873	9,873	9,873
	0-75%	0,24	0,23	0,28	0,639	0,639	0,639	7,452	7,452	7,452
	0-50%	0,70	0,89	0,95	2,319	2,315	2,267	0,698	0,701	0,756
	0-25%	1,05	1,31	2,52	0,864	0,818	0,645	0,665	0,732	0,964
30	0-100%	0,43	0,40	0,51	0,224	0,224	0,225	10,047	10,047	10,046
	0-75%	0,69	1,05	0,90	0,545	0,545	0,541	6,376	6,376	6,381
	0-50%	2,67	6,55	3,25	1,887	1,882	1,855	0,362	0,366	0,395
	0-25%	2,39	3,31	4,57	0,551	0,533	0,281	0,451	0,487	0,775
40	0-100%	0,94	0,95	0,93	0,197	0,198	0,197	9,905	9,904	9,905
	0-75%	1,79	1,96	1,94	0,631	0,631	0,629	4,304	4,304	4,306
	0-50%	6,63	7,08	8,10	1,453	1,449	1,412	0,210	0,213	0,253
	0-25%	4,12	5,83	9,29	0,435	0,438	0,255	0,348	0,374	0,607
50	0-100%	1,94	1,50	1,93	0,226	0,225	0,226	7,176	7,177	7,176
	0-75%	3,52	4,18	4,86	0,580	0,579	0,574	4,401	4,402	4,407
	0-50%	16,40	19,25	18,39	1,157	1,147	1,107	0,157	0,169	0,211
	0-25%	16,83	11,88	18,64	0,363	0,374	0,261	0,275	0,299	0,497

Tabela 8: Algoritmo HÍBRIDOm_δ aplicado às instâncias de W & G, com $\delta = m$, $\delta = 5$ e $\delta = 4$.

Basicamente precisamos evitar que padrões com mais do que δ itens distintos sejam gerados. Padrões são gerados no passo 1 do algoritmo SimplexGC, e nos algoritmos MOCHILA, FFDe e FFDWB. Este último algoritmo usa o algoritmo MOCHILAe como subrotina. No passo 1 do algoritmo simplexGC os padrões gerados possuem apenas um item. Precisamos então adaptar apenas os algoritmos MOCHILA,

MOCHILAe, FFDe e FFDWB, obtendo os algoritmos MOCHILA$_\delta$, MOCHILAe$_\delta$, FFDe$_\delta$ e FFDWB$_\delta$, respectivamente. As adaptações necessárias nos algoritmos SimplexGC, HÍBRIDO e HÍBRIDOm se resumem a utilizar os algoritmos MOCHILA$_\delta$, MOCHILAe$_\delta$, FFDe$_\delta$ e FFDWB$_\delta$.

Repetimos os mesmos testes descritos na seção 4, utilizando $\delta = 5$ e $\delta = 4$. Escolhemos utilizar sempre o algoritmo HÍBRIDOm_δ, visto que o algoritmo HÍBRIDOm mostrou-se mais eficiente, em termos de tempo, que o algoritmo HÍBRIDO, especialmente para instâncias maiores. Nestes testes, o nosso interesse foi centralizado na avaliação do tempo requerido para resolver os problemas e da qualidade das soluções encontradas. Dessa forma, nas tabelas desta seção, comparamos o tempo dispendido e a qualidade da solução encontrada.

Bitola	m médio	Tempo (seg)			Ganho sobre FFDe$_\delta$			Desperdício		
		$\delta = m$	$\delta = 5$	$\delta = 4$	$\delta = m$	$\delta = 5$	$\delta = 4$	$\delta = m$	$\delta = 5$	$\delta = 4$
5.0	7	0,23	0,31	0,31	0,945	0,945	0,945	0,324	0,324	0,324
6.3	11	0,30	0,33	0,44	1,868	1,849	1,830	1,438	1,453	1,472
8.0	12	0,35	0,33	0,42	1,128	1,128	1,128	6,281	6,281	6,281
10.0	20	0,94	0,89	1,06	0,825	0,825	0,825	7,843	7,843	7,843
12.5	14	0,53	0,33	0,56	2,207	2,207	2,207	5,467	5,467	5,467
16.0	13	0,35	0,23	0,23	1,247	1,247	1,247	20,675	20,675	20,675
20.0	8	0,12	0,15	0,19	0,894	0,894	0,894	12,023	12,023	12,023

Tabela 9: Algoritmo HÍBRIDOm_δ aplicado às instâncias práticas, com $\delta = m$, $\delta = 5$ e $\delta = 4$.

Nas tabelas 8, 9 e 10 comparamos o tempo médio, o ganho médio em relação às soluções encontradas pelo FFDe$_\delta$, e o desperdício médio verificado ao aplicar o algoritmo HÍBRIDOm_δ às instâncias de W & G, às instâncias práticas e às instâncias agrupadas, com $\delta = m$, $\delta = 5$ e $\delta = 4$. Os resultados obtidos mostraram que o desempenho dos algoritmos híbridos praticamente não é afetado, em termos de tempo de execução e qualidade da solução encontrada, mesmo quando δ assume valores pequenos, como 4 ou 5.

Bitola	m	Tempo (seg)			Ganho sobre FFDe$_\delta$			Desperdício		
		$\delta = m$	$\delta = 5$	$\delta = 4$	$\delta = m$	$\delta = 5$	$\delta = 4$	$\delta = m$	$\delta = 5$	$\delta = 4$
5.0	48	42	41	46	0,607	0,607	0,607	0,020	0,020	0,020
6.3	209	40	36	53	0,474	0,474	0,474	0,017	0,017	0,017
8.0	264	200	195	297	1,278	1,278	1,258	0,009	0,009	0,019
10.0	365	1601	1752	1586	1,365	1,365	1,365	0,061	0,061	0,061
12.5	301	484	658	840	1,481	1,481	1,481	0,268	0,268	0,268
16.0	218	169	165	176	1,673	1,673	1,673	4,995	4,995	4,995
20.0	105	32	25	28	0,489	0,489	0,489	2,507	2,507	2,507

Tabela 10: Algoritmo HÍBRIDOm_δ aplicado às instâncias agrupadas, com $\delta = m$, $\delta = 5$ e $\delta = 4$.

Considerações Finais

O algoritmo HÍBRIDO que desenvolvemos reúne algoritmos de diversas naturezas, como o SimplexGC, o FFDWB e o FFDe. Deste fato deriva o seu nome. Esta abordagem diversificada tem se mostrado promissora, e tem sido explorada em trabalhos recentes[55, 50, 63, 26]. Mostramos também que a diferença entre o valor da solução encontrada pelo algoritmo HÍBRIDO e o valor de uma solução inteira ótima é no máximo 1, se verdadeira a conjectura MIRUP. Dizemos então que o algoritmo HÍBRIDO é um algoritmo *quase-exato*.

Discorremos sobre resultados teóricos [48] e práticos [50, 62] que sugerem ser verdadeira a conjectura MIRUP. Os testes que realizamos com instâncias aleatórias e práticas vieram a fortalecer esta conjectura.

Esta conjectura nos fornece um excelente limitante para o valor de uma solução inteira ótima. Utilizando este limitante, propusemos o algoritmo exato FFDWB, que mostrou-se eficiente ao resolver instâncias pequenas do PCE_1.

Repetimos os testes realizados por Wäscher e Gau [63] com 4000 instâncias aleatórias e constatamos que o algoritmo HÍBRIDO obteve desempenho superior ao de todos os algoritmos testados por estes autores. O algoritmo HÍBRIDO encontrou uma solução inteira ótima para pelo menos 99,7% dessas instâncias e o tempo médio requerido para resolver cada uma das 4000 instâncias foi inferior a 4 segundos.

Como o algoritmo FFDWB é adequado apenas para instâncias pequenas, introduzimos uma modificação no algoritmo HÍBRIDO com o objetivo de reduzir o tamanho do problema residual a ser resolvido pelo FFDWB, obtendo assim o que chamamos de algoritmo HÍBRIDOm. Ao resolver as 4000 instâncias de W & G, o algoritmo HÍBRIDOm mostrou-se ainda mais rápido que o algoritmo HÍBRIDO, requerendo em média aproximadamente 3 segundos para cada instância. O algoritmo HÍBRIDOm também apresentou desempenho superior ao de todos os algoritmos testados por Wäscher e Gau, perdendo apenas para o algoritmo HÍBRIDO.

Resolvemos também 216 instâncias práticas, incluindo uma instância com 365 itens, sendo que o algoritmo HÍBRIDO encontrou soluções ótimas para todas essas instâncias. Os resultados obtidos, tanto nos testes realizados com instâncias geradas aleatoriamente, quanto nos testes com instâncias práticas, demonstraram a eficiência do algoritmo HÍBRIDO em termos de qualidade da solução e tempo de execução.

Motivados por restrições de ordem prática verificadas no processo de corte manual do aço utilizado na indústria de construção civil, investigamos o PCE_1 onde o número de itens distintos que podem ocorrer num padrão é limitado por uma constante inteira δ. Implementamos os algoritmos correspondentes para tratar desta variante por nós proposta. Repetimos os testes descritos na seção 4, com $\delta = 5$ e $\delta = 4$. Os resultados obtidos mostraram que o desempenho dos algoritmos híbridos praticamente não é afetado, em termos de tempo de execução e qualidade da solução encontrada, mesmo quando δ assume valores pequenos, como 4 ou 5.

Algoritmo	Classe	Tipo
FFDe	1/V/I/R	de aproximação
FFDWB	1/V/I/M	exato
HÍBRIDO	1/V/I/R	quase-exato
HÍBRIDOm	1/V/I/R	heurístico
MOCHILA$_\delta$	1/B/O	exato

Algoritmo	Classe	Tipo
MOCHILAe_δ	1/B/O	exato
FFDe_δ	1/V/I/R	de aproximação
FFDWB$_\delta$	1/V/I/M	exato
HÍBRIDO$_\delta$	1/V/I/R	quase-exato
HÍBRIDOm_δ	1/V/I/R	heurístico

Tabela 11: Algoritmos novos propostos na dissertação.

O esquema utilizado nos algoritmos híbridos propostos neste artigo, podem ser adaptados para o problema de corte bi- e tridimensional. Para isto é necessário dispor de métodos para achar uma solução inicial, gerar novas colunas e resolver o problema residual final. Tais métodos são encontrados na literatura. Um desdobramento natural deste trabalho de pesquisa seria integrar estes métodos de forma a obter algoritmos capazes de encontrar soluções inteiras para o problema de corte bi- e tridimensional.

Referências

[1] M. Arenales and R. Morábito, *An and/or-graph approach to the solution of two-dimensional non-guillotine cutting problems*, European Journal of Operations Research, 84 (1995), pp. 599–617.

[2] R. W. Ashford and R. C. Daniel, *Some lessons in solving practical integer programs*, Journal of the Operational Research Society, 43 (1992), pp. 425–433.

[3] B. S. Baker, D. J. Brown, and H. P. Katseff, *A $\frac{5}{4}$ algorithm for two-dimensional packing*, Journal of Algorithms, 2 (1981), pp. 348–368.

[4] S. Baum and L. E. T. Jr., *Integer rounding for polymatroid and branching optimization problems*, SIAM J. Alg. Disc. Meth., 2 (1981), pp. 416–425.

[5] A. Bortfeldt and H. Gehring, *Applying tabu search to container loading problems*, in Operations Research Proceedings, 1997, pp. 533–538.

[6] J. Bramel, W. T. Rhee, and D. Simchi-Levi, *Average-case analysis of the bin-packing problem with general cost structures*, Naval Res. Logist., 44 (1997), pp. 673–686.

[7] N. Christofides and C. Whitlock, *An algorithm for two dimensional cutting prblems*, Operations Research, 25 (1977), pp. 30–44.

[8] F. R. K. Chung, M. R. Garey, and D. S. Johnson, *On packing two-dimensional bins*, SIAM J. Algebraic Discrete Methods, 3 (1982), pp. 66–76.

[9] V. Chvátal, *Linear Programming*, W. H. Freeman and Company, New York, 1980.

[10] G. F. Cintra, *Algoritmos híbridos para problemas de corte unidimensional*, master's thesis, Instituto de Matemática e Estatística, São Paulo, 1998.

[11] G. F. Cintra and Y. Wakabayashi, *Um algoritmo híbrido para o problema de corte unidimensional*, in XXX Simpósio Brasileiro de Pesquisa Operacional, Curitiba, 1998.

[12] ——, *A hybrid algorithm for the unidimensional cutting stock problem*, (1999). Submitted.

[13] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson, *Approximation algorithms for bin packing - an updated survey*, in Algorithms design for computer system design, G. Ausiello, M. Lucertini, and P. Serafini, eds., Spring-Verlag, New York, 1984, pp. 49–106.

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

[15] CPLEX, *Using the CPLEX Callable Library and CPLEX Mixed Integer Library*, CPLEX Optimization, Inc, 1995.

[16] J. Csirik and G. J. Woeginger, *Shelf algorithms for on-line strip packing*, Information Processing Letters, 63 (1997), pp. 171–175.

[17] A. Diegel, *Integer lp solution for large trim problems*, Paper presented at the EURO/TIMS Conference, Paris, July 6-8, (1988).

[18] W. B. Dowsland, *Two and three dimensional packing problems and solution methods*, New Zealand Journal of Operational Research, 13 (1985), pp. 1–18.

[19] H. Dyckhoff, *A typology of cutting and packing problems*, European Journal of Operations Research, 44 (1990), pp. 145–159.

[20] H. Dyckhoff and U. Finke, *Cutting and Packing in Production and Distribution.*, Springer-Verlag, Heidelberg, 1992.

[21] E. Falkenauer, *A hybrid grouping genetic algorithm for bin packing*, Journal of Heuristics, 2 (1996), pp. 5–30.

[22] W. Fernandez de la Vega and G. S. Lueker, *Bin packing can be solved within $1+\epsilon$ in linear time*, Combinatorica, 1 (1981), pp. 349–355.

[23] M. Fieldhouse, *The duality gap in trim problems*, SICUP-bulletin, 5 (1990).

[24] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness*, W. H. Freeman and Co., San Fransisco, 1979.

[25] T. Gau, *Quasi-exact and heuristic algorithms for the standard one-dimensional cutting stock problem*, tech. report, Technische Universitaet Braunschweig, 1994.

[26] ——, *Solution methods for the standard one-dimensional cutting stock problem*, Physica, Heidelberg, 1997.

[27] T. Gau and G. Wäsher, *CUTGEN1: A problem generator for the standard one-dimensional cutting stock problem*, European Journal of Operations Research, 84 (1995), pp. 572–579.

[28] P. Gilmore and R. Gomory, *A linear programming approach to the cutting stock problem*, Operations Research, 9 (1961), pp. 849–859.

[29] ——, *A linear programming approach to the cutting stock problem - part II*, Operations Research, 11 (1963), pp. 863–888.

[30] B. L. Golden, *Approaches to the cutting stock problem*, AIIE Trans., 8 (1976), pp. 265–274.

[31] K. Heicken and W. König, *Integration eines heuristisch-optimierenden Verfahrens zur Lösung eines eindimensionalen Verschnittproblems in einem EDV-gestüzten Produktionsplanungs-und-steuerungssystems*, OR Spektrum, 1 (1980), pp. 251–259.

[32] J. C. Herz, *A recursive computational procedure for two-dimensional stock-cutting*, IBM Journal of Research Development, (1972), pp. 462–469.

[33] A. I. Hinxman, *The trim-loss and assortment problems: a survey*, European J. Oper. Res., 5 (1980), pp. 8–18.

[34] D. S. Johnson, *Near-optimal bin packing algorithms*, PhD thesis, Massachusetts Institute of Technology, Cambridge, Mass., 1973.

[35] D. S. Johnson, A. Demars, J. D. Ullman, M. R. Garey, and R. L. Graham, *Worst-case performance bounds for simple one-dimensional packing algorithms*, SIAM Journal on Computing, 3 (1974), pp. 299–325.

[36] N. Karmakar and R. M. Karp, *An efficient approximation scheme for the one dimensional bin packing problem*, in Proceedings, 23rd Ann. Symp. on Foundations of Computer Science, Los Angeles, 1982, IEEE Computer Society, pp. 312–320.

[37] C. C. Lee and D. T. Lee, *A simple on-line bin-packing algorithm*, J. Association Comput. Mach., 32 (1985), pp. 562–572.

[38] K. Li and K.-H. Cheng, *Heuristic algorithms for on-line packing in three dimensions*, Journal of Algorithms, 13 (1992), pp. 589–605.

[39] O. Marcotte, *The cutting stock problem and integer rounding*, Mathematical Programming, 33 (1985), pp. 82–92.

[40] ——, *An instance of the cutting stock problem for which the rounding property does not hold*, Oper. Res. Lett., 4 (1986), pp. 239–243.

[41] S. Martello and P. Toth, *Knapsack problems*, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons Ltd., Chichester, 1990. Algorithms and computer implementations.

[42] F. K. Miyazawa, *Algoritmos de aproximação para problemas de empacotamento*, PhD thesis, Instituto de Matemática e Estatística, São Paulo, 1997.

[43] F. K. Miyazawa and Y. Wakabayashi, *An algorithm for the three-dimensional packing problem with asymptotic performance analysis*, Algorithmica, 18 (1997), pp. 122–144.

[44] R. Morábito and M. N. Arenales, *Performance of two heuristics for solving large scale two-dimensional guillotine cutting problems*, INFOR, 33 (1995), pp. 145–155.

[45] K. Neumann and M. Morlock, *Operations Research*, Carl Hanser Verlag, Munich, 1993.

[46] J. F. Oliveira and J. S. Ferreira, *An improved version of Wang's algorithm for two-dimensional cutting problems*, European Journal of Operations Research, 44 (1990), pp. 256–266.

[47] J. F. Pierce, *Some large-scale production scheduling problems in the paper industry*, Prentice-Hall, Englewood Cliffs, 1964.

[48] G. Scheithauer and J. Terno, *About the gap between the optimal values of the integer and continuous relaxation one-dimensional cutting stock problem*, in Operations Research Proceedings, Berlin, 1992, Springer-Verlag.

[49] G. Scheithauer and J. Terno, *A branch & bound algorithm for solving one-dimensional cutting stock problems exactly*, Appl. Math. (Warsaw), 23 (1995), pp. 151–167.

[50] G. Scheithauer and J. Terno, *The modified integer round-up property of the one-dimensional cutting stock problem*, European Journal of Operations Research, 84 (1995), pp. 562–571.

[51] I. Schiermeyer, *Reverse-fit: A 2-Optimal algorithm for packing rectangles*, Lecture Notes in Computer Science, 855 (1994).

[52] A. Scholl, R. Klein, and C. Jürgens, *Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem*, Working Paper, Technische Hochschule Darmstadt, (1996).

[53] P. Schwerin and G. Wäscher, *The bin-packing problem: A problem generator and some numerical experiments with FFD packing and MTP*, International Transactions in Operational Research, 4 (1997), pp. 377–389.

[54] D. Simchi-Levi, *New worst-case results for the bin-packing problem*, Naval Res. Logist., 41 (1994), pp. 579–585.

[55] H. Stadtler, *A one-dimensional cutting stock problem in the aluminium industry and its solution*, European Journal of Operations Research, 44 (1990).

[56] A. Steinberg, *A strip-packing algorithm with absolute performance bound 2*, SIAM Journal on Computing, 26 (1997), pp. 401–409.

[57] P. E. SWEENY AND E. R. PATERNOSTER, *Cutting and packing problems: a categorized, application-oriented research bibliography*, Journal of the Operational Research Society, 43 (1992), pp. 691–706.

[58] P. H. VANCE, *Branch-and-price algoritms for the one-dimensional cutting stock problem*, Computational Optimization and Applications, 9 (1998), pp. 211–228.

[59] P. H. VANCE, C. BARNHART, E. L. JOHNSON, AND G. L. NEMHAUSER, *Solving binary cutting stock problems by column generation and branch and bound*, Computational Optimization and Applications, 3 (1994), pp. 111–130.

[60] F. VANDERBECK, *Computational study of a column generation algorithm for bin packing and cutting stock problems*, Research Papers in Management Studies, 14 (1996). Engineering Department and Judge Institute of Management Studies, University of Cambridge.

[61] P. Y. WANG, *Two algorithms for constrained two dimensional cutting stock problems*, Operations Research, 31 (1983), pp. 573–586.

[62] G. WÄSCHER AND T. GAU, *Two approaches to the cutting stock problem*, in IFORS'93 Conference, Lisbon, 1993.

[63] ——, *Heuristics for the integer one-dimensional cutting stock problem: a computational study*, OR Spektrum, 18 (1996), pp. 131–144.