Geração de padrões de cortes bidimensionais guilhotinados restritos via programação dinâmica e busca em grafo-e/ou

Reinaldo Morabito
Vitória Pureza
UFSCar

Resumo
Um método heurístico para geração de padrões de cortes bidimensionais guilhotinados restritos, baseado no método exato de Christofides e Hadjiconstantinou (1995) foi proposto em Silveira e Morabito (2002). O método combina uma relaxação do espaço de estados de uma formulação de programação dinâmica, um procedimento do tipo otimização do subgradiente e uma heurística de factibilização. Neste trabalho, o método de Silveira e Morabito é modificado com a utilização de uma heurística de factibilização mais efetiva que a anterior, e com uma abordagem de busca em grafo-e/ou para geração de boas soluções iniciais. Resultados computacionais de exemplos da literatura e gerados aleatoriamente indicam que o método refinado tem desempenho bem superior ao anterior, e é competitivo diante de outros métodos propostos na literatura.

Palavras-chave
Problemas de corte, padrões de cortes bidimensionais guilhotinados restritos, programação dinâmica, heurísticas.

Generation of constrained two-dimensional guillotine cutting patterns via dynamic programming and and/or-graph search

Abstract
A heuristic method for generating constrained two-dimensional guillotine cutting patterns based on the exact method by Christofides and Hadjiconstantinou (1995) was presented in Silveira and Morabito (2002). The method combines a state space relaxation of a dynamic programming formulation, a subgradient optimization procedure and an inner heuristic that turn infeasible solutions provided in each step of the optimization procedure into feasible solutions. In this work, the method of Silveira and Morabito is modified by using a more effective inner heuristic, and an and/or-graph search approach in order to generate good initial solutions. Results for benchmark and randomly generated instances indicate that the refined method's performance is superior to the previous one, and it is competitive face to other methods proposed in the literature.

Key words
Cutting problems, constrained two-dimensional guillotine cutting patterns, dynamic programming, heuristics.

INTRODUÇÃO

Problemas de corte e empacotamento aparecem em uma grande diversidade de classes e são, em geral, de difícil solução exata (DOWSLAND; DOWSLAND, 1992; DYCKHOFF; FINKE, 1992; LODI *et al.*, 2002; e SICUP, 2005). Existem vários estudos na literatura tratando estes problemas, como mostram os artigos de revisão e edições especiais em Dyckhoff e Waescher (1990), Sweeney e Paternoster (1992), Morabito e Arenales (1992), Martello (1994a, 1994b), Bischoff e Waescher (1995), Mukhacheva (1997), Dyckhoff *et al.* (1997), Arenales *et al.* (1999), Wang e Waescher (2002), Hifi (2002), Oliveira e Waescher (2006) e SICUP (2005). Neste trabalho estudamos a geração de padrões de cortes bidimensionais guilhotinados restritos. Nossa motivação é que, além de sua inerente dificuldade de resolução (problema NP-difícil), ele é importante em diversos processos industriais de corte de chapas retangulares, como por exemplo no corte de lâminas de vidros planos, placas de madeira na indústria de móveis e chapas de fibra de vidro para produção de placas de circuito impresso encomendadas por clientes (SILVEIRA; MORABITO, 2002). O padrão é dito bidimensional porque envolve duas dimensões relevantes para a solução (comprimento e largura das chapas e das peças recortadas); é guilhotinado porque as restrições impostas pelos equipamentos requerem que os cortes aplicados em um retângulo produzam dois retângulos; e é restrito porque há limitações para o número máximo de vezes que um tipo de peça poderá ser cortado a partir de uma chapa.

Alguns autores propõem métodos exatos para gerar padrões guilhotinados restritos, entre eles, Christofides e Whitlock (1977), Viswanathan e Bagchi (1993), Hifi (1997), Cung *et al.* (2000) e Christofides e Hadjiconstantinou (1995). Neste último, os autores apresentam um algoritmo de busca em árvore utilizando limitantes derivados de uma relaxação do espaço de estados de uma formulação de programação dinâmica, e um procedimento do tipo otimização do subgradiente (ascensão no espaço de estados). Os autores não chegaram a aplicar este método para resolver exemplos reais de processos de corte, mas apenas para resolver alguns exemplos da literatura e exemplos gerados aleatoriamente, todos de tamanho moderado.

Devido às dificuldades com métodos exatos, diversos autores apresentam métodos heurísticos para geração de padrões restritos, tais como Wang (1983), Vasko (1989), Oliveira e Ferreira (1990), Daza *et al.* (1995), Morabito e Arenales (1996), Mornar e Khoshnevis (1997), Fayard *et al.* (1998), Parada *et al.* (1998), Alvarez-Valdes *et al.* (2002) e Silveira e Morabito (2002). Neste último, os autores apresentam uma variação do método de Christofides e Hadjiconstantinou em que, em vez de aplicar o algoritmo de busca em árvore, aplicam uma heurística de factibilização simples que, a partir da solução relaxada de programação dinâmica gerada a cada iteração do procedimento de otimização do subgradiente, tenta melhorar a melhor solução factível obtida até então. O resultado é um método sem garantia de otimalidade, porém, bem mais rápido e capaz de fornecer soluções para problemas de maior porte do que o método exato de Christofides e Hadjiconstantinou.

No presente artigo, refinamos o método de Silveira e Morabito (2002) por meio de: (i) uma heurística de factibilização mais elaborada, capaz de gerar resultados melhores do que a anterior e demandando (em média) menores tempos computacionais, e (ii) uma solução inicial bem mais efetiva, gerada com a abordagem de busca em grafo-e/ou de Morabito e Arenales (1996). Uma das vantagens do presente método sobre outros métodos heurísticos disponíveis na literatura é que, na maioria dos casos em que se obtém uma solução ótima, é também fornecido um certificado de otimalidade para esta solução. Além disso, nos casos em que a solução obtida é subótima, o método é capaz de produzir um bom limitante superior para o valor da solução ótima, o que permite uma boa estimativa do *gap* de otimalidade. Convém observar que o método tem garantia de otimalidade para gerar padrões guilhotinados irrestritos (relaxados). O desempenho computacional do método é analisado resolvendo-se exemplos da literatura e exemplos gerados aleatoriamente, e comparando-se as soluções obtidas com as soluções do método de Silveira e Morabito (2002) e de outros métodos da literatura.

Este artigo está organizado da seguinte maneira: inicialmente é apresentada a formulação de programação dinâmica para o problema e alguns detalhes do método de solução baseado na relaxação de espaço de estados utilizado em Silveira e Morabito (2002). Em seguida, é proposta uma nova heurística de factibilização a ser utilizada no procedimento de otimização do subgradiente. Continuando, a abordagem de busca em grafo-/ou para geração de soluções iniciais de alta qualidade é descrita de forma resumida. A seguir são apresentados os resultados computacionais do método e, finalmente, as conclusões e perspectivas de pesquisa futura.

FORMULAÇÃO DE PROGRAMAÇÃO DINÂMICA

Assim como em Silveira e Morabito (2002), considere uma chapa de comprimento e largura L x W, e um conjunto de n retângulos menores (tipos de peças) com comprimentos e larguras l_1, x w_1, l_2, x w_2, ..., l_n x w_n, demandas b_1, b_2, ..., b_n, e valores v_1, v_2, ..., v_n. Admitimos que os comprimentos e larguras da chapa e das peças são números inteiros. De-

sejamos gerar o padrão de corte guilhotinado mais valioso, usando não mais do que b_i peças de cada tipo i = 1,2, ..., n. Se todos os valores v_i forem iguais às áreas $l_i w_i$, então o problema é equivalente a determinar o padrão de menor perda de material.

Para desenvolver uma fórmula recursiva de programação dinâmica, Christofides e Hadjiconstantinou (1995) admitem que os cortes sejam produzidos em estágios. A Figura 1 ilustra um padrão de corte em três estágios para um retângulo (x, y): no primeiro estágio, um corte horizontal divide o retângulo (x, y) em dois retângulos (superior e inferior); no segundo estágio, um corte vertical divide o retângulo superior em dois retângulos (superior esquerdo e superior direito) e dois cortes verticais dividem o retângulo inferior em três retângulos (inferior esquerdo, inferior intermediário e inferior direito); e, no terceiro estágio, um corte horizontal divide o retângulo superior esquerdo em dois retângulos e dois cortes horizontais dividem o retângulo inferior direito em três retângulos.

Sejam s_1, s_2, ..., s_n as quantidades de peças dos tipos 1, 2, ...,n que podem ser usadas para produzir um padrão de corte factível para o retângulo (x, y). Representando essas peças pela seqüência ordenada S_{xy} = {s_1, s_2,..., s_n}, Christofides e Hadjiconstantinou (1995) definem $F_k(x, y, S_{xy})$ como o valor do padrão de corte ótimo em até k-estágios, para um retângulo de tamanho (x, y), usando uma combinação factível de retângulos do conjunto S_{xy}, com cortes no primeiro estágio paralelos ao eixo y. Similarmente, seja $G_k(x, y, S_{xy})$ o valor do padrão de corte ótimo em até k-estágios, para um retângulo de tamanho (x, y), usando uma combinação factível de retângulos do conjunto S_{xy}, com cortes no primeiro estágio paralelos ao eixo x. Definindo X = {1, 2, ... , L} e Y = {1, 2, ..., W}, as funções recursivas de programação dinâmica $F_k(x, y, S_{xy})$ e $G_k(x, y, S_{xy})$, para todo $k \geq 1$, $x \in X$, $y \in Y$ e $S_{xy} \subseteq S_{LW}$, são dadas por:

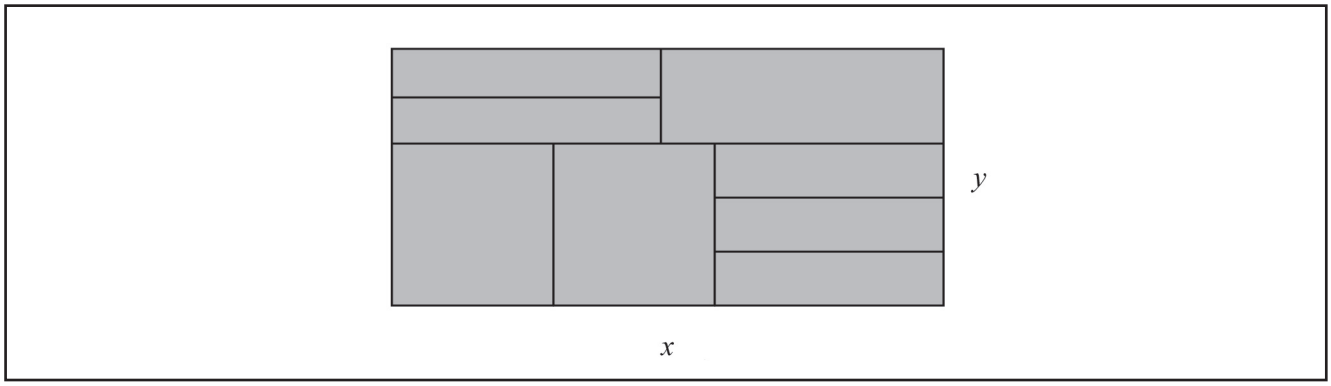
$$F_k(x,y,S_{xy}) = \max\left[G_{k-1}(x,y,S_{xy}), \max_{x'<x, x'\in X, S'\subset S_{xy}}\left[F_k(x',y,S') + G_{k-1}(x-x',y,S_{xy}-S')\right]\right] \quad (1)$$

$$G_k(x,y,S_{xy}) = \max\left[F_{k-1}(x,y,S_{xy}), \max_{y'<y, y'\in Y, S'\subset S_{xy}}\left[G_k(x,y',S') + F_{k-1}(x,y-y',S_{xy}-S')\right]\right] \quad (2)$$

O primeiro termo do colchete da fórmula (1) corresponde ao caso em que, no primeiro estágio do padrão ótimo em até k estágios de (x, y), não há cortes em (x, y) paralelos ao eixo y, o que implica que este padrão pode ser considerado como um padrão em até k-1 estágios, com os cortes do primeiro estágio paralelos ao eixo x. O segundo termo do colchete da fórmula (1) corresponde ao caso em que, no primeiro estágio do padrão ótimo em até k estágios de (x, y), existe pelo menos um corte em (x, y) paralelo ao eixo y, digamos um corte vertical $x' \in X$, $x'<x$, produzindo dois retângulos (x', y) e (x-x', y). Neste caso, temos dois padrões de corte associados com estes retângulos. No primeiro, as peças pertencem ao conjunto S', os cortes no primeiro estágio do padrão ótimo de (x', y) são paralelos ao eixo y, e ainda há k estágios no total (dado que pode haver outros cortes verticais $x''<x'$ em (x', y), que não implicam em outro estágio em (x, y)) . No segundo, as peças pertencem ao conjunto complementar S_{xy} - S', os cortes no primeiro estágio do padrão ótimo de (x-x', y) são paralelos ao eixo x, e há somente k-1 estágios no total (dado que cortes horizontais em (x-x', y) implicam em outro estágio em (x, y)). Estas considerações são aplicáveis similarmente para a fórmula recursiva (2).

Um valor para k = 0 em $F_k(x, y, S_{xy})$ ou $G_k(x, y, S_{xy})$ (fórmulas (1) e (2)) corresponde a uma alocação da maior peça (i.e., mais valiosa) do conjunto S_{xy} para o retângulo (x, y).

Figura 1: Exemplo de padrão de corte para (x, y) em três estágios.



Os dois cortes para produzir esta peça a partir de (x, y) não são considerados como estágios adicionais de corte. Assim, condições iniciais são fornecidas, para todo x, y e S_{xy}, por:

$$F_0(x, y, S_{xy}) = \max\left[0, \max_{i \in S_{xy}}\{v_i \mid l_i \le x, w_i \le y\}\right], \quad (3)$$

$$G_0(x, y, S_{xy}) = F_0(x, y, S_{xy}). \quad (4)$$

Note que o máximo entre $F_k(x, y, S_{xy})$ e $G_k(x, y, S_{xy})$ em (1)-(4) corresponde ao padrão ótimo guilhotinado restrito com até k-estágios do retângulo (x, y), gerado pelo uso das peças em S_{xy}, quando a direção dos cortes do primeiro estágio não é especificada. Em particular, $\max[F_k(L, W, S_{LW}), G_k(L, W, S_{LW})]$ corresponde ao padrão ótimo k-estágios da chapa (L, W). Se não for especificado um número máximo de estágios, devemos considerar o valor de k como sendo aquele onde: $F_k(L, W, S_{LW}) = Fk_{+1}(L, W, S_{LW})$ e $G_k(L, W, S_{LW}) = G_{k+1}(L, W, S_{LW})$. Sem perda de generalidade, as dimensões dos conjuntos X e Y nestas fórmulas ainda podem ser substancialmente reduzidas aplicando a restrição dos padrões canônicos ou normais (Christofides e Whitlock, 1977; Beasley, 1985). Uma conseqüência disso é que os conjuntos X e Y podem ser redefinidos por:

$$X = \left\{x \mid x = \sum_{i=1}^{n} \theta_i l_i, \quad 1 \le x \le L,\ 0 \le \theta_i \le b_i \ \text{ e } \ \theta_i \text{ inteiro}, i = 1,\ldots,n\right\}$$

$$Y = \left\{y \mid y = \sum_{i=1}^{n} \tau_i w_i, \quad 1 \le y \le W,\ 0 \le \tau_i \le b_i \ \text{ e } \ \tau_i \text{ inteiro}, i = 1,\ldots,n\right\}$$

Note na formulação (1)-(4) que para gerar um padrão guilhotinado irrestrito basta relaxar a variável de estado S_{LW} e as fórmulas se reduzem às fórmulas apresentadas em Beasley (1985). De fato, a principal dificuldade da formulação (1)-(4) é a dimensão do espaço de estados associada a S_{LW}. Christofides e Hadjiconstantinou (1995) propõem uma relaxação do espaço de estados de (1)-(4), combinada com um procedimento do tipo otimização do subgradiente (ascensão no espaço de estados), que produz bons limitantes para o problema original. Para maiores detalhes sobre este procedimento, veja Christofides e Hadjiconstantinou (1995) e Silveira e Morabito (2002).

Lucena (2004) propôs algumas modificações do tipo *relax and cut* em procedimentos de otimização do subgradiente. Em uma destas modificações, sugere que uma boa prática para convergência do procedimento de otimização do subgradiente seja ajustar os subgradientes antes do cálculo do tamanho do passo (veja expressão para t em Silveira e Morabito, 2002), usando $s_i = 0$ sempre que $s_i \ge 0$ e $q_i = 0$, em que $s_i = b_i - \gamma_i$ é o subgradiente associado à restrição de disponibilidade da peça i, q_i é o multiplicador (peso) da peça i e γ_i é o número de vezes que a peça i foi usada na solução (padrão) (veja expressões para q_i e s_i em Silveira e Morabito, 2002). Ou seja, se a restrição i não está violada (isto é, $s_i \ge 0$) na iteração corrente, e possui multiplicador associado nulo (isto é, $q_i = 0$), então seu gradiente é desconsiderado, fixando-o em $s_i = 0$. Com isso, ele não contribui no cálculo do tamanho do passo t. Esta estratégia de ajuste dos multiplicadores não foi utilizada em Silveira e Morabito (2002), porém é aplicada no presente trabalho.

HEURÍSTICAS DE FACTIBILIZAÇÃO

A heurística de factibilização proposta em Silveira e Morabito (2002) é bastante simples e pode ser resumida nos seguintes passos:

1. Para cada peça do tipo i em excesso faça:
   1.1 Retire aleatoriamente uma unidade da peça i, obtendo um espaço vazio S_i correspondente.
   1.2 Para cada espaço S_i resultante do passo anterior, determine a solução homogênea H_i^* (arranjo com peças do mesmo tipo) mais valiosa dentre os tipos de peças que ainda não atenderam a demanda.
2. Preencha o espaço S_i^* (com arranjo H_i^*) que resulta no maior valor para o padrão de corte. Atualize a demanda das peças utilizadas.
3. Retorne ao passo 1 enquanto houver excesso de peças no padrão.

Note que o preenchimento de cada espaço S_i, ocupado por uma única peça do tipo i, por um arranjo de peças do mesmo tipo (solução homogênea) dentro de S_i, mantém o padrão da chapa guilhotinado. Esta heurística é executada em cada iteração do algoritmo de otimização do subgradiente no referido trabalho. Se em alguma destas iterações for obtida uma solução factível, então esta solução é ótima. Caso contrário, o método retorna a melhor solução produzida pela heurística de factibilização ao longo de todas as iterações, juntamente com o limitante inferior fornecido pelo algoritmo para o valor do padrão ótimo. Convém ressaltar que Christofides e Hadjiconstantinou (1995) não exploram heurísticas de factibilização, mas aplicam os limitantes obtidos em um procedimento de busca em árvore para garantir a obtenção de um padrão ótimo para o problema.

A seguir, é investigada uma nova variação de heurísticas de factibilização mais elaborada que a descrita acima. Assim como a heurística anterior, computa e seleciona a cada iteração o espaço mais valioso que resulta da retirada de peças em excesso. Entretanto, no caso de peças que apresentam um número em excesso maior que um, verifica-se se peças desse tipo estão adjacentes e formam um bloco retangular

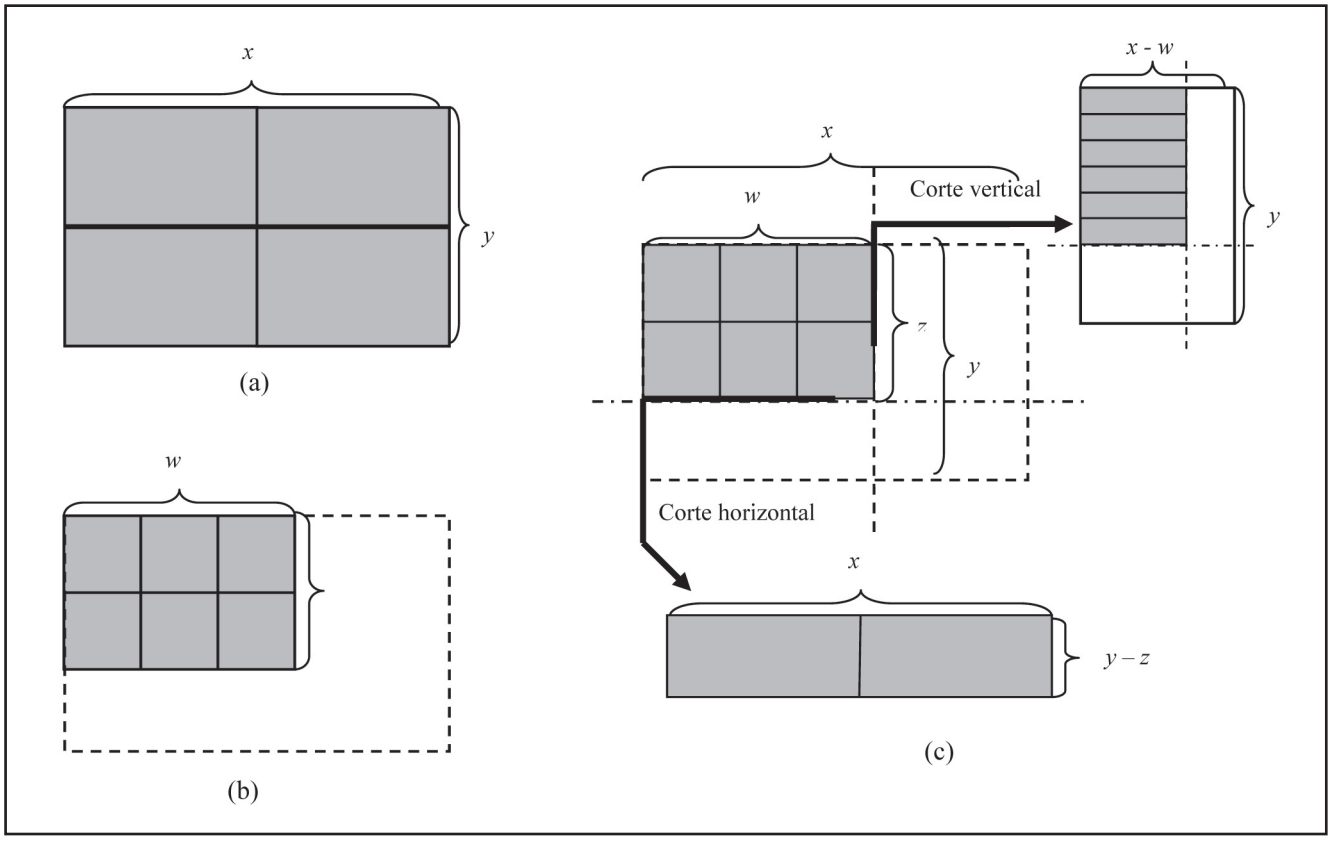
no padrão corrente. Desta forma, ao invés de retirar cada peça em excesso e preencher o espaço resultante com outras peças, é retirado em uma única etapa um bloco de peças adjacentes em um número preferencialmente igual ao número em excesso. Alguns cuidados são tomados para que o padrão resultante de chapa continue guilhotinado, conforme discutido adiante.

A retirada deste bloco define um espaço retangular correspondente à casca convexa das peças selecionadas com uma área maior que a de uma única peça, surgindo assim oportunidades mais interessantes de preenchimento. Note que no caso da peça com menor área, a retirada uma a uma do número em excesso não resulta em espaços aproveitáveis. Entretanto, a retirada de duas ou mais peças adjacentes pode permitir a inclusão de outras peças e a obtenção de soluções de melhor qualidade. Caso a peça seja excedida em uma única unidade e o espaço resultante não permita a inclusão de nenhuma outra peça, duas unidades adjacentes (se houver) são retiradas. A fim de maximizar ainda mais o aproveitamento do espaço, são incorporados à área do bloco de peças retirado espaços existentes não aproveitados entre este bloco e peças adjacentes do padrão que não foram afetadas pelo procedimento. Estes espaços precisam corresponder a tiras retangulares abaixo, acima, à direita e à esquerda do bloco para que a área para inclusão de novas peças mantenha seu formato retangular.

Outra importante diferença desta heurística em relação à anterior diz respeito à forma com que os espaços resultantes da retirada de peças são aproveitados. Enquanto a versão original preenche cada espaço com um padrão homogêneo de um único tipo de peça, a versão proposta permite a reaplicação do procedimento a espaços remanescentes após o preenchimento do espaço original através de um procedimento recursivo. Informações dos tipos e número de peças utilizadas no preenchimento são armazenadas e consideradas na escolha das peças em estágios posteriores do procedimento. A Figura 2 ilustra este procedimento.

Note que o espaço remanescente após a inclusão do padrão homogêneo da Figura 2b pode ser aproveitado a partir da aplicação de um corte guilhotinado vertical ou de um cor-

Figura 2: Aproveitamento de espaços de peças retiradas. (a) bloco retangular de peças adjacentes de um mesmo tipo a serem retiradas; (b) inclusão de padrão homogêneo de peças de tipo diverso no espaço do bloco; (c) aproveitamento dos espaços remanescentes com dimensões (*x- w, y*), resultantes de corte guilhotinado vertical, e (*x, y - z*), resultante de corte guilhotinado horizontal.



te guilhotinado horizontal (Figura 2c). Cada corte define uma possível área retangular a ser preenchida. O corte horizontal possibilita a inclusão de um padrão homogêneo de duas peças, que preenche totalmente o espaço disponibilizado. O corte vertical possibilita a inclusão de um padrão homogêneo de seis peças; após este corte, permanece ainda uma área não utilizada no espaço disponibilizado, a qual pode, a princípio, ser aproveitada através de um outro corte vertical ou horizontal. Como todos estes cortes são guilhotinados, o padrão resultante no bloco (x, y) da Figura 2 é guilhotinado.

Uma das vantagens do presente método sobre outros métodos heurísticos disponíveis na literatura é que, na maioria dos casos em que se obtém uma solução ótima, é também fornecido um certificado de otimalidade para esta solução.

Para garantir que o padrão resultante na chapa (L, W) continue guilhotinado, a heurística adota uma regra conservadora para seleção de blocos candidatos. Especificamente, são considerados somente os blocos que não envolvam cortes no seu interior (para gerar as peças do bloco) coincidentes com cortes fora do bloco (para gerar as peças adjacentes ao bloco). Dizemos que uma peça é adjacente ao bloco se ela não pertence ao bloco mas contém uma face (ou parte dela) adjacente a uma face do bloco. Por exemplo, suponha que o bloco com quatro peças da Figura 2a seja definido pelas coordenadas inicial e final (p_1,q_1) e (p_2,q_2). Suponha também que o bloco envolva um corte vertical interior (para separar suas quatro peças em dois conjuntos de duas peças), digamos na abscissa p $(p_1 < p < p_2)$, e um corte horizontal interior (para produzir as peças de cada conjunto de duas peças), digamos na ordenada q $(q_1 < q < q_2)$. Este bloco é considerado pela heurística somente se não houver peças adjacentes às faces superior ou inferior do bloco cujas abscissas iniciais ou finais sejam iguais a p (i.e., coincidentes com o corte vertical interior em p), e peças adjacentes às faces esquerda ou direita do bloco cujas ordenadas iniciais ou finais sejam iguais a q (i.e., coincidentes com o corte horizontal interior em q). Note que por meio desta regra, a escolha do padrão de corte dentro do bloco torna-se independente do padrão definido fora do bloco.

A Figura 3 ilustra a aplicação da regra, onde o bloco de quatro caixas da Figura 2a é parte do padrão de uma chapa. As linhas grossas representam alguns cortes guilhotinados C_i que foram aplicados para produção do padrão da chapa, onde i indica a ordem de aplicação do corte. Note na Figura 3a que o bloco seria desconsiderado para seleção, uma vez que a abscissa p do corte vertical interior do bloco coincide com a abscissa inicial da peça adjacente A, localizada em sua face superior. De fato, e como ilustrado na Figura 3b, a seleção do bloco e aplicação dos dois cortes verticais consecutivos da Figura 2c incluiria um padrão de 12 caixas, que não poderia ser obtido por nenhuma seqüência de cortes guilhotinados na placa (padrão resultante não guilhotinado). Na Figura 3c, o mesmo bloco seria candidato para seleção, dado que não existem peças adjacentes em suas faces superior ou inferior com coordenadas inicial ou final p, ou em suas faces esquerda ou direita com coordenadas inicial ou final q; a inclusão do padrão de 12 caixas resultaria, portanto, em um padrão guilhotinado que poderia ser obtido após a seqüência de cortes guilhotinados C_1 a C_4 (Figura 3d).

É importante enfatizar que a severidade da regra adotada pode eliminar opções viáveis. Se, por exemplo, a peça adjacente A da Figura 3c fosse substituída por um padrão homogêneo de duas peças, o bloco seria desconsiderado, uma vez que a abscissa p de seu corte vertical interior coincidiria com a abscissa inicial de uma das peças. Note, entretanto, que a seleção do bloco e inclusão do padrão manteria o padrão guilhotinado da chapa. Por razões de simplicidade de implementação e aplicação, a regra foi mantida na heurística nos experimentos realizados.

Os passos gerais da heurística proposta são apresentados a seguir.

Passos da heurística de factibilização proposta

1. Para cada peça em excesso i faça:
   - 1.1 Se a peça apresenta excesso de uma unidade, verifique se é possível preencher a área desta peça com outra peça cuja demanda não foi atendida. Neste caso, faça o número de peças a serem retiradas igual a 1. Caso a área seja insuficiente e haja mais de uma peça deste tipo no padrão corrente, faça o número de peças a serem retiradas igual a 2.
   - 1.2 Se o número de peças a serem retiradas for maior que 1, verifique no padrão a existência de subconjuntos de peças deste tipo que estejam adjacentes, formando cascas convexas retangulares. Caso existam espaços vazios entre o subconjunto de peças e peças adjacentes do padrão que não foram afetadas pelo procedimento, adicione-os à casca convexa, desde que esta se mantenha retangular. Selecione o bloco que mais se aproxima do número de peças a serem retiradas,

o qual corresponde ao espaço S_i mais promissor para aproveitamento.

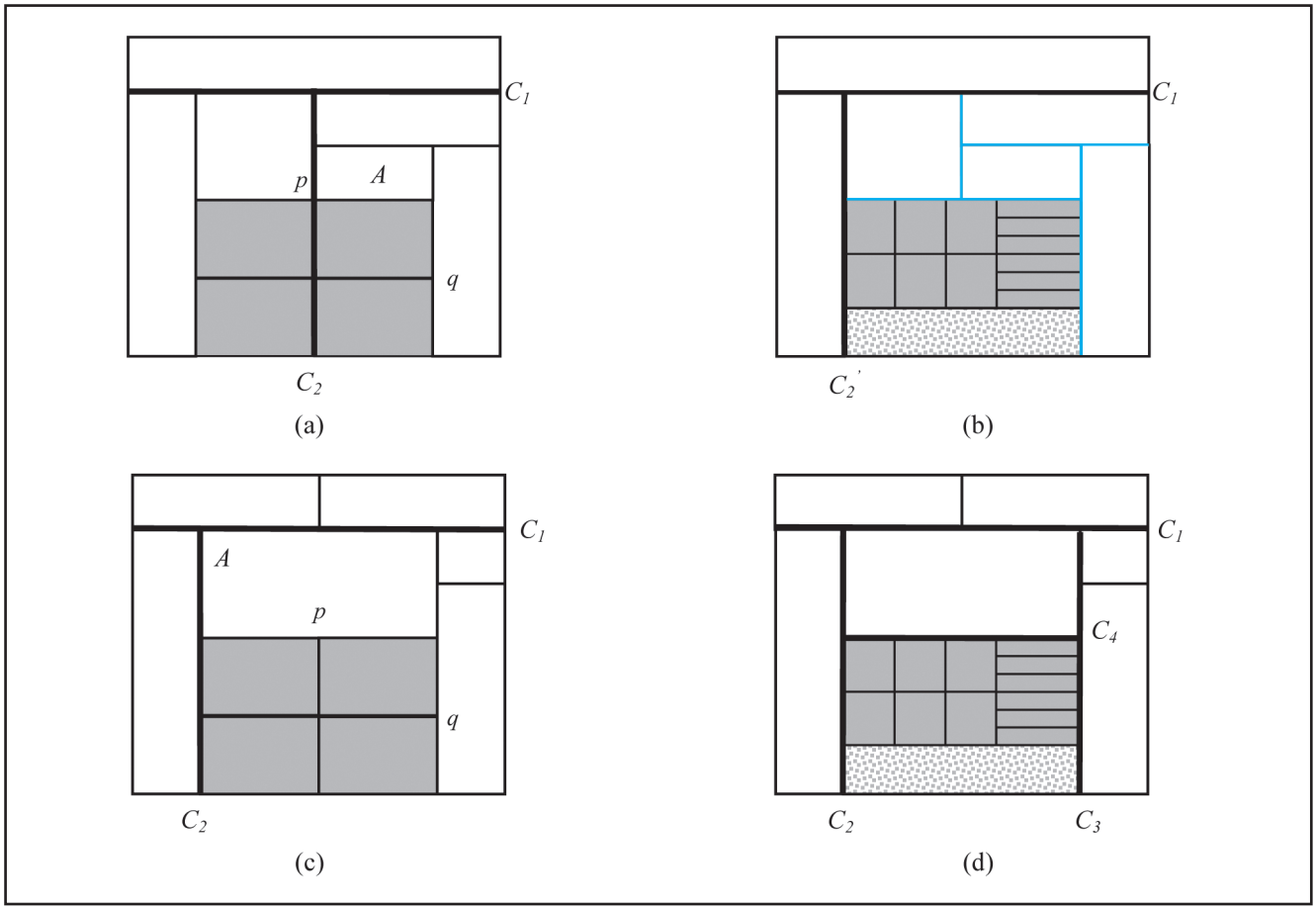
1.3 Retire do padrão o bloco de peças selecionado.

2. Para cada espaço S_i resultante do passo anterior:
   2.1 Faça $A_i = \varnothing$ (conjunto de arranjos homogêneos já selecionados) e $S_{corrente} = S_i$.
   2.2 Para $S_{corrente}$, determine a solução homogênea mais valiosa H^* dentre os tipos de peças que ainda não atenderam a demanda, e faça $A_i = A_i \cup H^*$. Caso o arranjo não tenha preenchido completamente $S_{corrente}$, obtenha até dois espaços retangulares correspondentes à área não utilizada em $S_{corrente}$ através de um corte guilhotinado horizontal e de um corte guilhotinado vertical (Figura 2).
   2.3 Para o espaço não utilizado resultante de cada tipo de corte, verifique se sua área é suficientemente grande para ser preenchida com peças cuja demanda não tenha sido atendida, considerando as decisões de arranjos armazenadas em A_i. Neste caso, faça este novo espaço igual a $S_{corrente}$ e retorne ao passo 2.2, armazenando o arranjo na seqüência da árvore de decisões associada.
   2.4 Selecione a seqüência de decisões de arranjo que resultem no maior valor de padrão de corte do espaço S_i e a faça igual a A^*_i.
3. Preencha o espaço S_i (com o arranjo A^*_i) que resulta no maior valor para o padrão de corte. Atualize a demanda das peças utilizadas.
4. Retorne ao passo 1 enquanto houver excesso de peças no padrão.

ABORDAGEM DE BUSCA EM GRAFO-E/OU

Uma solução factível inicial trivial para o método descrito nas seções "Formação de programação dinâmica" e "Heurísticas de factibilização" consiste simplesmente em não cortar

Figura 3: Aplicação da regra de seleção de blocos candidatos. (a) padrão de peças com bloco cuja abscissa de corte vertical interno coincide com a abscissa inicial da peça adjacente *A*; (b) padrão não guilhotinado resultante da seleção do bloco; (c) padrão de peças com bloco com coordenadas de corte interno não coincidentes com coordenadas iniciais de peças adjacentes; (d) padrão guilhotinado resultante da seleção do bloco.



nenhuma peça (i.e., solução com valor nulo). Outra solução um pouco mais elaborada é a solução homogênea (composta de peças do mesmo tipo), mais valiosa, não excedendo a demanda deste tipo (conforme a expressão (17) em MORABITO e ARENALES, 1996). Silveira e Morabito (2002) utilizam o algoritmo de Wang (1983) para fornecer uma boa solução factível inicial (primeiro limitante inferior). Note que outros algoritmos poderiam ter sido aplicados com este propósito.

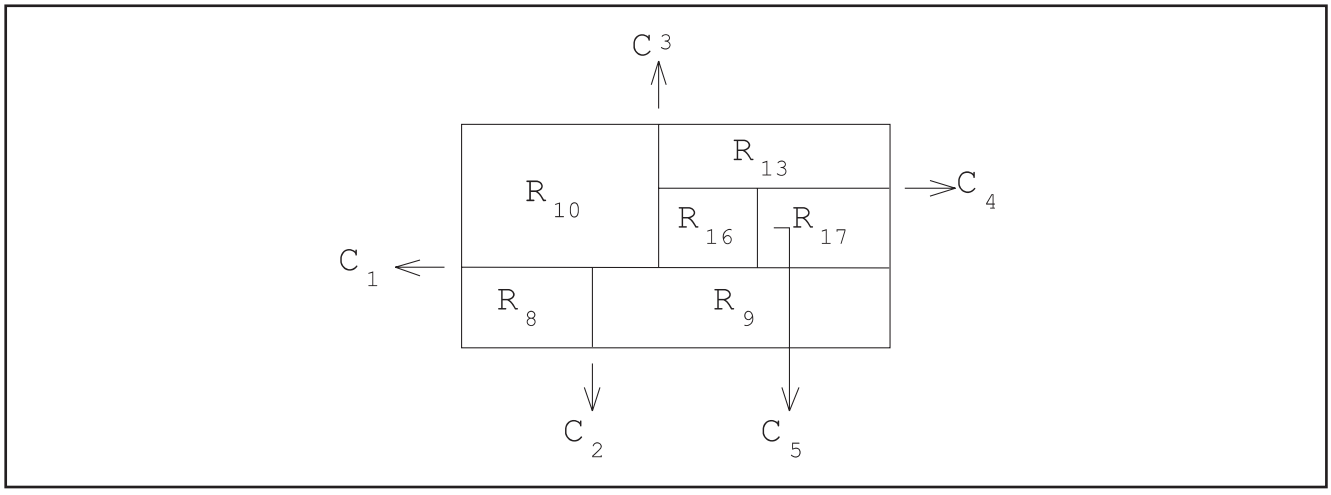
No presente artigo, propomos utilizar a abordagem de busca em grafo-e/ou (MORABITO; ARENALES, 1996). Esta abordagem (aqui denotada AOG) representa basicamente cada possível padrão de corte bidimensional guilhotinado restrito como um caminho completo em um grafo-e/ou em que: (i) os nós do grafo correspondem ao retângulo inicial (chapa), aos retângulos intermediários obtidos durante o processo de corte, aos retângulos encomendados (peças) ou à perda, e (ii) os arcos-e do grafo correspondem aos cortes sobre os retângulos (ligando um nó aos outros nós). A Figura 4 ilustra uma seqüência de cinco cortes guilhotinados C1, C2, ..., C5, produzindo um padrão de corte contendo os retângulos (peças) denominados R8, R9, R10, R13, R16 e R17. A Figura 5 apresenta o caminho completo correspondente do grafo-e/ou com 17 nós (retângulos R1, R2, ..., R17) e 5 arcos-e (cortes C1, C2, ..., C5). Note que o corte horizontal C1 sobre o retângulo R1 (nó inicial) produz os retângulos intermediários R2 e R3, o corte vertical C2 sobre o retângulo R2 produz os retângulos R4 e R5 (que não podem mais ser cortados e resultam nas peças R8 e R9 após um 0-corte, i.e. um corte que representa a opção de não cortar mais o retângulo, tornando-o um nó final), o corte vertical C3 sobre o retângulo R3 produz os retângulos intermediários R6 e R7, e assim por diante.

Cada caminho completo no grafo-e/ou corresponde a um padrão de corte. A abordagem de busca AOG é, portanto, um método de enumeração implícita. Para reduzir o tamanho da busca, AOG utiliza os conjuntos *X* e *Y* relativos aos padrões normais definidos na seção "Formação de programação dinâmica". Além disso, para evitar que diferentes caminhos no grafo-e/ou resultem em padrões de corte equivalentes (i.e., contendo as mesmas peças), ela aplica regras de simetria, exclusão e ordenação de cortes. Com vistas a melhorias no desempenho da busca, a abordagem utiliza estratégias de busca híbridas (combinação *backtracking* com *hill-climbing*), limitantes inferiores e superiores, as heurísticas H1-H4, entre outras. A heurística H1 desconsidera um caminho a partir de um nó se o limitante superior do valor deste caminho não for λ_1% maior que o valor da melhor solução conhecida até então para este nó. A heurística H2 desconsidera um caminho a partir de um nó se o limitante inferior do valor deste caminho não for pelo menos λ_2% do valor da melhor solução conhecida até então para este nó. A heurística H3 utiliza uma regra de simetria para reduzir o tamanho da busca no grafo. A heurística H4 define um limite *M* para o número máximo de elementos dos conjuntos *X* e *Y*. Mais detalhes do algoritmo completo podem ser obtidos em Morabito e Arenales (1996) e Vianna (2000).

EXPERIMENTOS E RESULTADOS COMPUTACIONAIS

Nesta seção apresentamos os experimentos e resultados computacionais a fim de ilustrar o desempenho do método aqui proposto. As implementações foram codificadas em linguagem Pascal (compilador Delphi 6 – Borland) e executadas em um microcomputador Pentium IV (2,99 Ghz – 2,0 GBytes de RAM). Utilizamos os códigos do algoritmo de otimização do subgradiente desenvolvidos em Silveira e

Figura 4: Seqüência de cortes guilhotinados produzindo um padrão de corte.



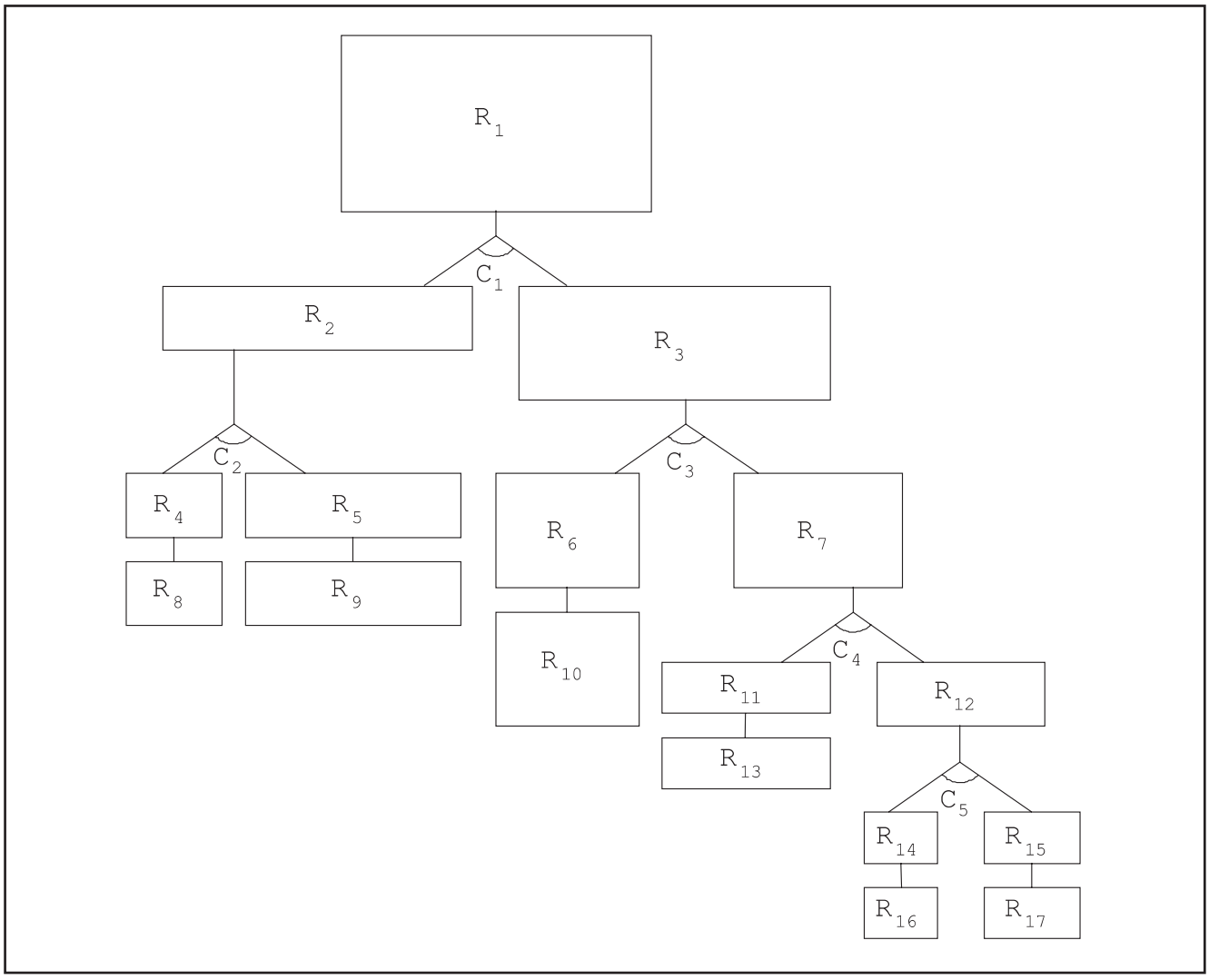
Morabito (2002), e os códigos da abordagem em grafo-e/ou desenvolvidos em Morabito e Arenales (1996). Após alguns experimentos computacionais preliminares, os parâmetros da otimização do subgradiente escolhidos foram: $\pi = 2$, $\varepsilon = 0{,}05$ (limite de 18 iterações) e estratégia de ajuste dos multiplicadores ativada. Os parâmetros da abordagem AOG selecionados foram: limite de profundidade da busca $DB = 6$, $\lambda_1 = 0$ (heurística H1 desligada) e $\lambda_2 = 0{,}9$ (heurística H2 ligada). A regra de simetria foi ativada (heurística H3 ligada) e os conjuntos X e Y foram completamente gerados para cada exemplo (heurística H4 desligada). Para maiores detalhes sobre estes parâmetros, veja Morabito e Arenales (1996).

Como primeira etapa de nossos experimentos, investigamos o impacto das duas heurísticas de factibilização descritas na seção "Heurísticas de factibilização". Especificamente, foram analisados a qualidade das soluções geradas e o esforço computacional requerido pelo algoritmo com a heurística de factibilização proposta (aqui denotado por algoritmo MP) em relação aos resultados obtidos pelo algoritmo com a heurística de factibilização empregada em Silveira e Morabito (aqui denotado por SM02). Diferentemente do que foi empregado naquele trabalho, ambos os algoritmos não utilizam o algoritmo de Wang (1983) para inicializar o limitante inferior do padrão ótimo. A solução inicial corresponde a uma simples solução homogênea (expressão (7) em MORABITO; ARENALES, 1996), permitindo assim uma verificação mais precisa da robustez de cada algoritmo.

Comparação entre heurísticas de factibilização — algoritmos SM02 e MP

Nesta primeira etapa, utilizamos 28 exemplos, 26 dos quais analisados em Silveira e Morabito (2002) para com-

Figura 5: Caminho completo do grafo-e/ou representando o padrão de corte da Figura 4.



parar as duas heurísticas de factibilização. Tomamos inicialmente oito exemplos amplamente utilizados na literatura com soluções ótimas conhecidas: os exemplos *CW1*, *CW2* e *CW3* em Christofides e Whitlock (1977), *OF1* e *OF2* em Oliveira e Ferreira (1990), e *WANG1*, *WANG2* e *WANG3* em Wang (1983).

Além disso, nos casos em que a solução obtida é subótima, o método é capaz de produzir um bom limitante superior para o valor da solução ótima, o que permite uma boa estimativa do gap de otimalidade.

A Tabela 1 resume os dados dos exemplos e os resultados computacionais obtidos com o método baseado na heurística de factibilização anterior (SM02) e pela heurística proposta neste trabalho (MP). Em cada exemplo da tabela, as colunas 2 a 5 (*Dados*) informam algumas de suas características; a segunda coluna indica as dimensões (L,W) da chapa, a terceira coluna apresenta o número n de tipos de peças, e a quarta e quinta colunas representam, respectivamente, a cardinalidade dos conjuntos normais X e Y de coordenadas no eixo x e no eixo y para posicionamento do canto inferior esquerdo das peças. A coluna z^- fornece o melhor limitante superior gerado pelo algoritmo de otimização do subgradiente, cujo valor para cada exemplo foi idêntico para as duas heurísticas. Para cada heurística, é apresentado o valor da melhor solução factível obtida, seguido dos tempos computacionais (em segundos) para a melhor solução e o total decorrido na execução.

Note que, para estes exemplos, a heurística MP obteve as mesmas ou melhores soluções do que as providas pela heurística SM02; dentre os exemplos, apenas *CW2* não foi resolvido otimamente. O método de otimização do subgradiente proveu certificados de otimalidade em todos os exemplos para os quais foi atingida a solução ótima, exceto no exemplo *OF2*. Observe também que em três dos oito exemplos, MP requereu menores tempos computacionais para obtenção da melhor solução e para finalização da execução. Convém salientar que o método exato de Christofides e Hadjiconstantinou (1995) foi capaz de encontrar as soluções ótimas de todos os exemplos da Tabela 1; no entanto, os tempos computacionais foram bem elevados, atingindo a ordem de horas em um microcomputador IBM 486 (veja Tabela 6 adiante). Ao compararmos os resultados de MP com outros métodos heurísticos da literatura, por exemplo, o método de Fayard *et al.* (1998), este encontrou soluções piores para os exemplos

Tabela 1: Resultados para oito exemplos da literatura.

EXEMPLO	DADOS				z^-	SM02	MP
CW1	(15,10)	7	12	10	244	**244**◆ 0,4 0,4	**244**◆ 0,6 0,6
CW2	(40,70)	10	29	56	2892	2533 234,1 395,7	**2768** 64,3 242,5
CW3	(40,70)	20	25	55	1860	**1860**◆ 129,1 129,1	**1860**◆ 93,8 93,8
OF1	(70,40)	10	48	32	2737	**2737**◆ 44,6 44,6	**2737**◆ 106,5 106,5
OF2	(70,40)	10	41	32	2717	2506 31,9 154,8	**2690**❖ 5,0 28,5
WANG1	(69,33)	20	54	18	2277	**2277**◆ 0,1 0,1	**2277**◆ 0,1 0,1
WANG2	(70,39)	20	55	24	2694	**2694**◆ 0,1 0,2	**2694**◆ 0,2 0,2
WANG3	(70,40)	20	55	25	2721	**2721**◆ 2,8 2,8	**2721**◆ 2,8 2,8

❖Solução ótima sem certificado de otimalidade.
◆ Solução ótima com certificado de otimalidade.

CW2 (2731), *CW3* (1740), *OF1* (2713) e *OF2* (2586), com tempos computacionais da ordem de poucos segundos em uma estação de trabalho Sparc-Server20.

A seguir comparamos o desempenho das heurísticas SM02 e MP nos outros 20 exemplos analisados em Silveira e Morabito (2002). Estes exemplos foram gerados aleatoriamente com chapas de dimensões $(L, W) = (100, 100)$ e $n = 10$ peças, divididos em duas classes. Na classe 1, as dimensões das peças l_i, w_i, $i = 1,2, \ldots, n$, foram sorteadas (e em seguida arredondadas) de uma distribuição uniforme nos intervalos $[0{,}25L, 0{,}75L]$, $[0{,}25W, 0{,}75W]$, respectivamente. Na classe 2, as dimensões l_i, w_i, $i = 1,2, \ldots, n$, foram sorteadas (e em seguida arredondadas) de uma distribuição uniforme nos intervalos $[0{,}20L, 0{,}50L]$, $[0{,}20W, 0{,}50W]$, respectivamente. Esses intervalos têm sido utilizados por outros autores na literatura, como em Beasley (1985), Morabito e Arenales (1996) e Hifi (1997). Ressaltamos que o desempenho computacional do algoritmo com as heurísticas SM02 e MP depende da cardinalidade dos conjuntos X e Y; assim, à medida que as peças tendem a ficar menores em relação à chapa, os conjuntos X e Y tendem a conter mais elementos e o esforço computacional demandado pelo algoritmo tende a crescer. Nestes exemplos, os valores de utilidade v_i, $i = 1,2,\ldots n$, foram considerados iguais às áreas $l_i w_i$ das peças, e as demandas d_i foram sorteadas de uma distribuição uniforme no intervalo [1, 4]. Como estes exemplos foram gerados aleatoriamente, a solução ótima não é conhecida *a priori*; só há garantia de otimalidade quando a solução coincide com o limitante superior do algoritmo de otimização do subgradiente.

Assim como na Tabela 1, a coluna z^- das Tabelas 2 e 3 fornece o melhor limitante superior gerado pelo algoritmo de otimização do subgradiente, cujo valor para cada exemplo foi idêntico para as heurísticas SM02 e MP. Para cada heurística, é então apresentado o valor da melhor solução factível obtida, seguido dos tempos computacionais (em segundos) para a melhor solução e o total decorrido na execução.

Como podemos observar, para os exemplos da classe 1 (dimensões das peças com 25% a 75% das dimensões da chapa), as soluções obtidas pelas heurísticas SM02 e MP são quase sempre ótimas e acompanhadas de um certificado de otimalidade (Tabela 2). Por outro lado, para os exemplos da classe 2 (dimensões das peças com 20% a 50% das dimensões da chapa), as soluções obtidas nem sempre são acompanhadas de um certificado de otimalidade (Tabela 3), o que indica que as heurísticas de factibilização são mais efetivas

Tabela 2: Resultados para exemplos da classe 1 — $(L,W) = (100,100)$, $n = 10$.

EXEMPLO	DADOS		z^-	SM02	MP
1	24	30	8828	**8828**◆ 3,1 3,1	**8828**◆ 5,7 5,7
2	26	13	8244	**8244**◆ 0,1 0,1	**8244**◆ 0,1 0,1
3	35	24	8180	**8180**◆ 0,9 0,9	**8180**◆ 0,2 0,3
4	33	22	8799	**8799**◆ 4,0 4,0	**8799**◆ 1,6 1,6
5	32	37	9507	**9194** 0,1 12,0	**9194** 0,1 12,1
6	40	25	8464	**8464**◆ 1,3 1,3	**8464**◆ 0,8 0,8
7	27	27	8751	**8751**◆ *0,1 0,1*	**8751**◆ 0,1 0,1
8	31	35	9715	**9715**◆ 0,1 0,1	**9715**◆ 0,1 0,1
9	40	26	8919	**8919**◆ 0,1 0,1	**8919**◆ 0,1 0,1
10	35	31	9611	8571 0,1 25,2	**9168** 0,1 13,1
Tempos médios				1,0 4,7	0,9 3,4

◆ Solução com certificado de otimalidade.

em problemas com peças maiores. Isso se deve ao fato de que nos exemplos com peças menores cabem relativamente mais peças dentro da chapa, aumentando assim as chances de que o algoritmo de otimização de subgradiente gere soluções infactíveis (i.e., cortando mais peças no padrão do que a demanda requerida). Note que nos exemplos 10 de ambos os conjuntos, a heurística MP obteve soluções melhores em relação à heurística SM02.

No tocante aos tempos computacionais requeridos, observe que para alguns dos exemplos da classe 2 (Tabela 3), os tempos de obtenção da melhor solução e o tempo total da execução para MP são maiores que os para SM02. Entretanto, se considerarmos os casos para os quais MP apresenta tempos menores de obtenção da melhor solução (exemplos 6 e 8), os ganhos em velocidade de convergência compensam em larga escala as perdas nos demais exemplos. Para os exemplos da classe 1 (Tabela 2), as diferenças nos tempos entre os algoritmos se reduzem. Conforme apontado na última linha de ambas as tabelas, a heurística MP consome tempos computacionais médios inferiores aos da heurística SM02 para estes dois conjuntos de exemplos. Conclui-se, portanto, que a utilização da heurística de factibilização aqui proposta resulta em um algoritmo superior a SM02 tanto em termos de qualidade de solução como de esforço computacional.

Uma outra questão importante investigada neste trabalho é como o uso de um método para geração do primeiro limitante inferior pode agilizar e melhorar os resultados até então obtidos com o método MP. Nosso segundo conjunto de experimentos contempla a utilização da abordagem em grafo-e/ou (AOG). Nas análises que se seguem, a versão de MP inicializada com AOG é denotada por MP_AOG.

Comparação entre MP, MP_AOG e algoritmos da literatura

Além dos 28 exemplos já apresentados, foram incluídos os resultados de outras 22 instâncias bem conhecidas da literatura (FAYARD *et al.*, 1998; ALVAREZ-VALDES *et al.*, 2002). O desempenho do algoritmo MP é comparado com sua versão MP_AOG e com métodos competitivos já reportados na literatura.

Exemplos gerados aleatoriamente

As Tabelas 4 e 5 apresentam os resultados de MP e MP_AOG para os 20 exemplos gerados aleatoriamente utilizados em Silveira e Morabito (2002) e descritos no item que trata da comparação entre heurísticas de factibilização. Para o conjunto de exemplos da classe 1 (Tabela 4), observa-se que a utilização da procedimento de busca em grafo-e/ou

Tabela 3: Resultados para exemplos da classe 2 — (*L*,*W*) = (100,100), *n* = 10.

EXEMPLO	DADOS		Z^-	SM02	MP
1	51	43	9446	**9238** 23,2 222,9	**9238** 30,0 135,7
2	42	43	9308	**9216** 3,5 17,1	**9216** 3,6 17,2
3	45	41	9356	**8932** 1,8 102,5	**8932** 2,2 126,4
4	58	39	9682	**9682**◆ 0,2 0,3	**9682**◆ 0,2 0,3
5	55	46	9381	**9381**◆ 1,3 1,3	**9381**◆ 1,3 1,3
6	53	42	9148	**8914** 15,4 62,9	**8914** 9,9 75,5
7	34	50	9656	**9656**◆ 0,2 0,2	**9656**◆ 0,2 0,2
8	50	49	9891	**9533** 191,7 339,8	**9533** 80,8 134,3
9	41	31	8964	**8964**◆ 0,1 0,2	**8964**◆ 0,1 0,2
10	35	53	9477	7119 0,2 135,8	**9199** 0,2 16,0
Tempos médios				23,8 88,3	12,9 50,7

◆ Solução com certificado de otimalidade.

resulta em soluções de melhor qualidade nos exemplos 5 e 10. Para o conjunto de exemplos da classe 2 (Tabela 5), melhores soluções são obtidas nos exemplos 1, 3, 6 e 8. Note também a redução dos tempos computacionais com a aplicação do procedimento, em particular no tocante aos tempos de obtenção da melhor solução, todos menores que 0,1 segundo.

O algoritmo MP_AOG gerou a melhor solução em todos os exemplos; nos casos em que a solução ótima foi obtida, coube ao procedimento de otimização do subgradiente combinado à heurística de factibilização certificar sua otimalidade. A Figura 6 ilustra padrões ótimos obtidos para alguns exemplos deste conjunto.

Exemplos da literatura

A seguir, são apresentados os resultados da aplicação de MP e MP_AOG a três conjuntos de exemplos da literatura. O 1º conjunto é composto pelos oito exemplos utilizados na comparação entre heurísticas de factibilização, caracterizados por até 20 tipos de peças (Tabela 6), enquanto os exemplos do 2º e 3º conjuntos (HIFI, 1997; FAYARD *et al.*, 1998) apresentam de 25 a 50 tipos de peças. Nos 11 exemplos do 2º conjunto (Tabela 7), o valor da peça é igual a sua área, enquanto que 11 exemplos do 3º conjunto (Tabela 8) o valor da peça não é igual à área. Em cada tabela, *n* indica o número de tipos de peças de cada exemplo.

Como estes exemplos são amplamente conhecidos, foram incluídos os valores das soluções de algoritmos competitivos que os trataram, assim como os tempos computacionais requeridos (se disponibilizados). Os algoritmos selecionados são referenciados como se segue:

- **CW77** (CHRISTOFIDES e WHITLOCK, 1977): Algoritmo de busca em árvore com programação dinâmica e uma rotina de transporte – computador CDC 7600.
- **W83** (WANG, 1983): Algoritmo 1 com $\beta = 0{,}01$, 0,02 e 0,03, respectivamente – computador Univac 1100.
- **OF90** (OLIVEIRA e FERREIRA, 1990): Variação do algoritmo 1 de Wang (1983) – microcomputador 80286/7.
- **VB93** (VISWANATHAN e BAGCHI, 1993): Algoritmo de busca *best-first* – computador VAX 11.
- **CH95** (CHRISTOFIDES e HADJICONSTANTINOU, 1995): Algoritmo de busca em árvore com relaxação do espaço de estados de uma formulação de programação dinâmica – microcomputador IBM 486.
- **H97** (HIFI, 1997): Variação do algoritmo de Viswanathan e Bagchi – computador Data General AV 8000.
- **FHZ98** (FAYARD *et al.*, 1998): Algoritmo geral de corte da melhor faixa, explorando a abordagem em dois estágios de Gilmore and Gomory – computador Sparc 20.
- **APT02** (ALVAREZ-VALDÉS *et al.*, 2002): Algoritmo de busca tabu – microcomputador Pentium II.

Tabela 4: Resultados para exemplos da classe 1 — (*L*,*W*) = (100,100), *n* = 10.

EXEMPLO	DADOS		Z^-	MP	MP_AOG
1	24	30	8828	**8828**◆ 5,7 5,7	**8828**◆ < 0,1 2,4
2	26	13	8244	**8244**◆ 0,1 0,1	**8244**◆ < 0,1 0,1
3	35	24	8180	**8180**◆ 0,2 0,3	**8180**◆ < 0,1 0,2
4	33	22	8799	**8799**◆ 1,6 1,6	**8799**◆ < 0,1 1
5	32	37	9507	9194 0.1 12,1	**9397** < 0,1 9,1
6	40	25	8464	**8464**◆ 0,8 0,8	**8464**◆ < 0,1 0,6
7	27	27	8751	**8751**◆ 0,1 0,1	**8751**◆ < 0,1 0,1
8	31	35	9715	**9715**◆ 0,1 0,1	**9715**◆ < 0,1 0,1
9	40	26	8919	**8919**◆ 0,1 0,1	**8919**◆ < 0,1 0,1
10	35	31	9611	9168 0,1 13,1	**9399** < 0,1 10,7
Tempos médios				0,9 3,4	< 0,1 2,4

◆ Solução com certificado de otimalidade.

É importante ressaltar que dentre estes, CW77, VB93, CH95 e H97 oferecem certificado de otimalidade. Comparações no tocante à qualidade das soluções reportadas envolvem todos os algoritmos, enquanto aquelas que dizem respeito aos tempos computacionais foram restritas a MP e MP_AOG. Isso se deve tanto à ausência de dados nos artigos referidos (e.g. algoritmo APT02) como às diferenças nos computadores e ambientes computacionais (sistemas operacionais, linguagens, compiladores, etc.) utilizados. O desempenho computacional com base em informações disponibilizadas em sítios especializados da Internet (e.g., www.netlib.org/benchmark/linpackjava, www.spec.org) também não foi aqui incluído, ou por não terem sido encontradas quaisquer informações (computadores mais antigos) ou pela exigência de dados adicionais relativos a suas configurações e ambientes computacionais, por sua vez não disponibilizados nos artigos originais.

A utilização do método AOG forneceu a solução ótima a todos os exemplos do 1º conjunto, o que permitiu que MP_AOG resolvesse otimamente uma instância a mais do que MP (Tabela 6). Os algoritmos da literatura proveram soluções ótimas a todas as instâncias a que foram aplicados, cujo número é infelizmente muito limitado para uma comparação geral apropriada. A Figura 7 apresenta padrões ótimos obtidos para alguns exemplos deste conjunto.

Além do melhor desempenho em termos de qualidade de solução, AOG gerou limitantes iniciais apertados que permitiram a obtenção do certificado de otimalidade para a maioria das instâncias em um tempo computacional substancialmente menor do que o requerido quando o método não é empregado. Note, entretanto, que para MP_AOG, o certificado de otimalidade não foi conferido a *OF1* e *OF2*, enquanto que MP foi capaz de certificar a otimalidade de *OF1*.

Assim como verificado para o 1º conjunto de exemplos, o método AOG forneceu a solução ótima para todas as instâncias do 2º conjunto (Tabela 7). Como resultado, MP_AOG resolveu otimamente todos os exemplos, enquanto MP obteve soluções ótimas em oito exemplos (padrões ótimos obtidos para alguns exemplos deste conjunto são ilustrados na Figura 8). Observe também que os algoritmos da literatura FHZ98 e APT02 obtiveram, respectivamente, soluções ótimas em 2 e 6 exemplos deste conjunto.

Cabe ressaltar que os exemplos com maior número de peças deste conjunto, em geral, requereram um tempo computacional substancialmente maior em cada iteração do subgradiente. Por esta razão, os algoritmos MP e MP_AOG

Tabela 5: Resultados para exemplos da classe 2 — (*L*,*W*) = (100,100), *n* = 10.

EXEMPLO	DADOS		z^-	MP	MP_AOG
1	51	43	9446	9238 30,0 135,7	**9307** < 0,1 103,9
2	42	43	9308	**9216** 3,6 17,2	**9216** < 0,1 9,9
3	45	41	9356	8932 2,2 126,4	**9088** < 0,1 26,7
4	58	39	9682	**9682♦** 0,2 0,3	**9682♦** < 0,1 0,3
5	55	46	9381	**9381♦** 1,3 1,3	**9381♦** < 0,1 1,0
6	53	42	9148	8914 9,9 75,5	**8918** < 0,1 25,8
7	34	50	9656	**9656♦** 0,2 0,2	**9656♦** < 0,1 0,2
8	50	49	9891	9533 80,8 134,3	**9550** < 0,1 133,9
9	41	31	8964	**8964♦** 0,1 0,2	**8964♦** < 0,1 0,2
10	35	53	9477	**9199** 0,2 16,0	**9199** < 0,1 19,9
Tempos médios				12,9 50,7	< 0,1 32,2

♦ Solução com certificado de otimalidade.

Figura 6: Padrões ótimos de exemplos gerados aleatoriamente: (a) exemplo 5, (b) exemplo 7 e (c) exemplo 9 da Tabela 5; (d) exemplo 1, (e) exemplo 6 e (f) exemplo 8 da Tabela 4.

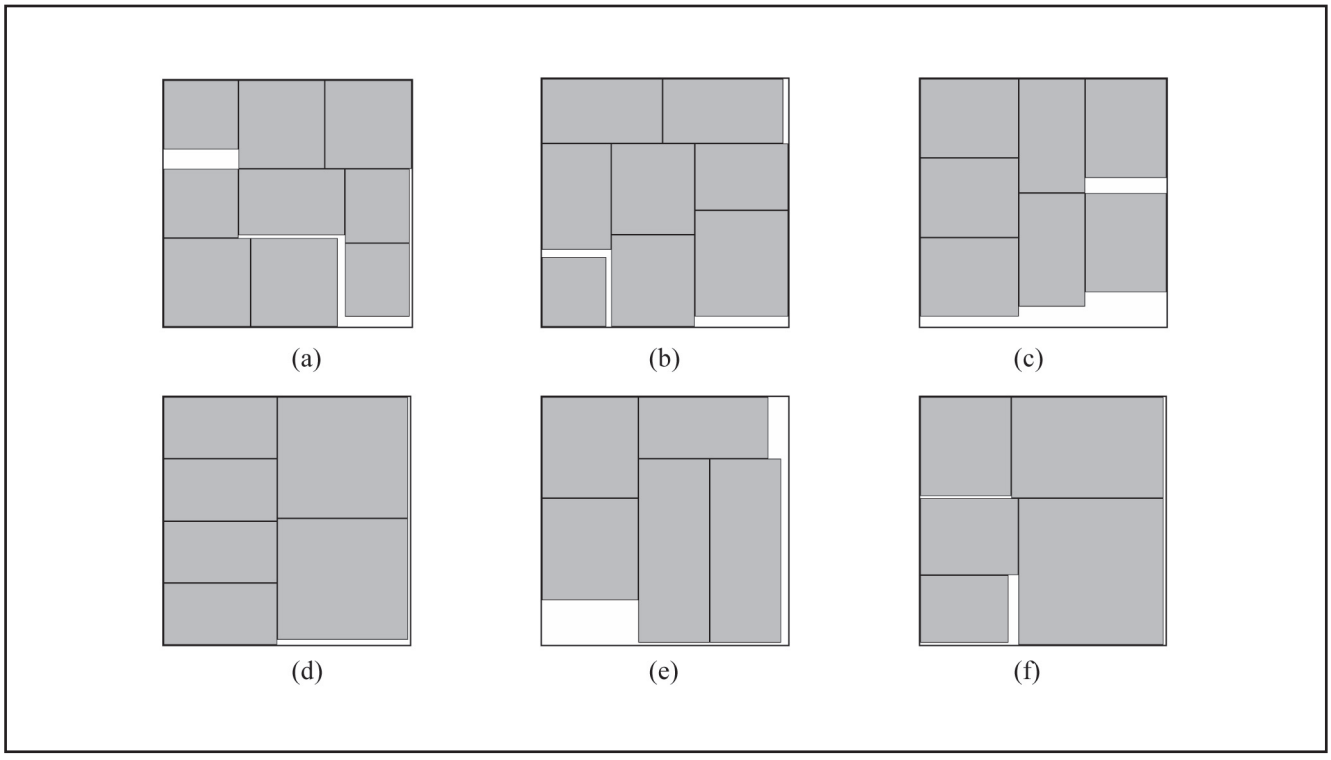


Tabela 6: Resultados — 1º conjunto de exemplos da literatura.

EXEMPLO	N	ALGORITMO								
		CW77	W83	OF90	VB93	CH95	H97	APT02	MP	MP_AOG
CW1	7	**244**◆ 2,5	-	-	**244**◆ 1,6	**244**◆ 120	**244**◆ < 1	-	**244**◆ 0,6 0,6	**244**◆ < 0,1 0,1
CW2	10	**2892**◆ 24,1	-	-	**2892**◆ 12	**2892**◆ 4236	**2892**◆ 7,1	**2892**❖ N/D	**2768** 64,3 242,5	**2892**◆ 0,8 38,4
CW3	20	**1860**◆ 66,1	-	-	**1860**◆ 69,8	**1860**◆ 3912	**1860**◆ 29,8	**1860**❖ N/D	**1860**◆ 93,8 93,8	**1860**◆ 0,1 19,9
OF1	10	-	-	**2737**❖ N/D	-	-	-	-	**2737**◆ 106,5 106,5	**2737**❖ 0,1 11,2
OF2	10	-	-	**2690**❖ N/D	-	-	-	-	**2690**❖ 5,0 28,5	**2690**❖ 0,1 23,6
WANG1	20	-	**2277**❖ 23	-	-	-	-	-	**2277**◆ 0,1 0,1	**2277**◆ < 0,1 0,1
WANG2	20	-	**2694**❖ 34	-		-	-	-	**2694**◆ 0,2 0,2	**2694**◆ < 0,1 0,2
WANG3	20	-	**2721**❖ 73	-	**2721**◆ 180	-	-		**2721**◆ 2,8 2,8	**2721**◆ < 0,1 0,6

❖Solução ótima sem certificado de otimalidade.
◆ Solução ótima com certificado de otimalidade.
N/D: não disponibilizado na fonte.

foram modificados de forma a limitar o tempo de execução a um máximo de 1800 segundos. Neste caso, é retornada a melhor solução factível obtida até o momento. Para ambos, MP e MP_AOG, a interrupção do algoritmo foi verificada nos exemplos *CU7*, *CU10* e *CU11*.

Os limitantes fornecidos por AOG permitiram a obtenção do certificado de otimalidade para todos os exemplos (exceto, naturalmente, *CU7*, *CU10* e *CU11*) em um tempo computacional menor do que o requerido quando o método não é empregado. O desempenho do método MP_AOG em

Figura 7: Padrões ótimos de exemplos da literatura — conjunto 1: (a) CW1; (b) OF1; (c) OF2; (d) WANG2.

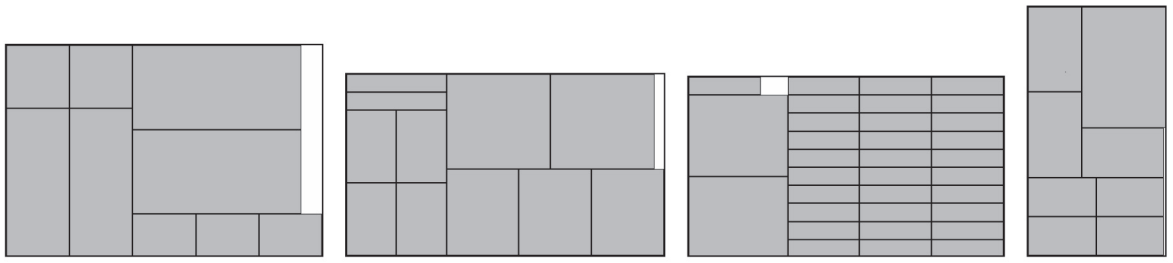


Tabela 7: Resultados - 2º conjunto de exemplos da literatura.

EXEMPLO	N	ALGORITMO			
		FHZ98	APT02	MP	MP_AOG
CU1	25	12312 7,1	**12330**❖ N/D	**12330**◆ 0,4 0,4	**12330**◆ 0,1 0,5
CU2	35	25806 19,8	**26100**❖ N/D	**26100**◆ 1,1 1,2	**26100**◆ 0,2 1,4
CU3	45	16608 30,5	16679 N/D	**16723**◆ 45,8 45,9	**16723**◆ 0,3 5,8
CU4	45	98190 55,8	99366 N/D	**99945**◆ 105,9 105,9	**99945**◆ 3,8 35,0
CU5	50	171651 89,6	**173364**❖ N/D	**173364**◆ 768,2 768,2	**173364**◆ 2,7 873,2
CU6	45	**158572**❖ 29,8	**158572**❖ N/D	**158572**◆ 10,2 10,3	**158572**◆ 1,3 11,8
CU7	45	246860 26,7	**247150**❖ N/D	240409 23,6 1800,0	**247150**❖ 0,1 1800,0
CU8	35	432198 68,8	432714 N/D	**433331**◆ 20,6 20,6	**433331**◆ 0,5 26,8
CU9	25	**657055**❖ 47,2	**657055**❖ N/D	**657055**◆ 19,6 19,7	**657055**◆ 0,1 19,7
CU10	40	764696 123,9	773485 N/D	766724 250,2 1800,0	**773772**❖ 69,2 1800,0
CU11	50	913387 219,6	922161 N/D	917107 854,7 1800,0	**924696**❖ 333,6 1293,1
Tempos médios				190,9 579,3	37,4 533,4

❖Solução ótima sem certificado de otimalidade.
◆ Solução ótima com certificado de otimalidade.
N/D: não disponibilizado na fonte.

termos de qualidade das soluções é bom, quando comparado ao dos métodos FHZ98 e APT02.

O melhor desempenho do método proposto inicializado com AOG é mais uma vez verificado nos exemplos do 3º conjunto (Tabela 8). Enquanto MP obteve soluções ótimas em oito dos 11 exemplos, MP_AOG resolveu otimamente todas as 11 instâncias, e apenas para *CW2* não se obteve o certificado de otimalidade (a Figura 9 ilustra padrões ótimos obtidos para alguns exemplos deste conjunto). Os algoritmos da literatura FHZ98 e APT02 obtiveram, respectivamente,

Figura 8: Padrões ótimos de exemplos da literatura — conjunto 2: (a) CU4; (b) CU7; (c) CU8; (d) CU11.

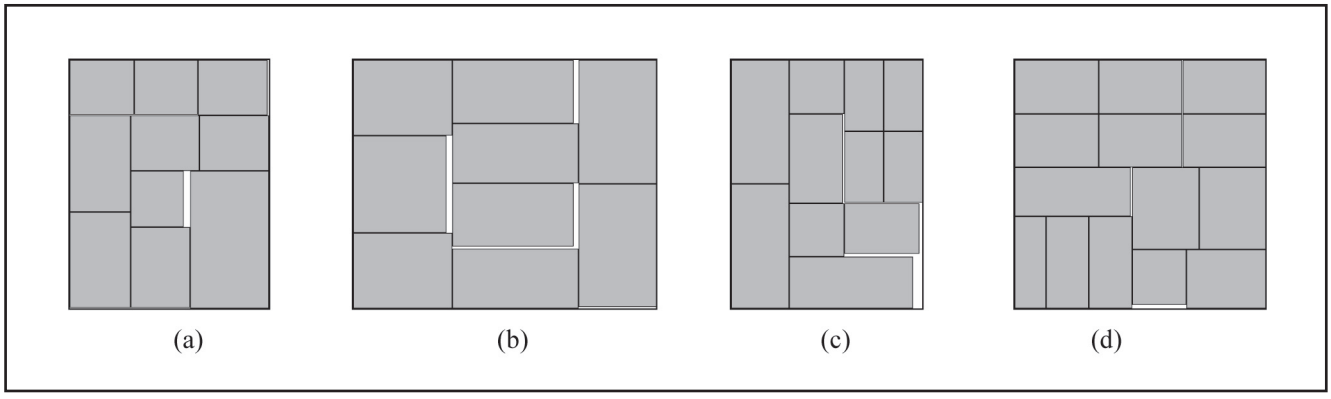


Tabela 8: Resultados — 3º conjunto de exemplos da literatura.

EXEMPLO	N	ALGORITMO			
		FHZ98	APT02	MP	MP_AOG
CW1	25	**6402**❖ 5,3	**6402**❖ N/D	**6402**◆ 0,6 24,7	**6402**◆ 1,1 25,6
CW2	35	**5354**❖ 6,8	**5354**❖ N/D	5281 1,1 970,7	**5354**❖ 0,7 1118,8
CW3	45	5148 17,3	**5689**❖ N/D	**5689**◆ 27,7 27,7	**5689**◆ 0,5 26,9
CW4	45	6168 30,8	6170 N/D	6170 10,1 1800	**6175**◆ 1,3 46,8
CW5	50	11550 19,8	11644 N/D	**11659**◆ 1431,5 1431,5	**11659**◆ 0,8 1268,5
CW6	45	12403 79,0	**12923**❖ N/D	**12923**◆ 920,6 920,7	**12923**◆ 33,3 890,6
CW7	45	9484 61,4	**9898**❖ N/D	**9898**◆ 8,8 8,8	**9898**◆ 4,1 10,7
CW8	35	4504 107,4	**4605**❖ N/D	**4605**◆ 517,2 517,2	**4605**◆ 20,8 340,5
CW9	25	**10748**❖ 125,3	**10748**❖ N/D	**10748**◆ 8,0 104,4	**10748**◆ 7,0 121,3
CW10	40	6116 281,1	**6515**❖ N/D	**6515**◆ 1012,5 1012,5	**6515**◆ 2,8 182,2
CW11	50	6084 197,5	**6321**❖ N/D	5897 430,8 1800	**6321**◆ 2,0 1375,0
Tempos Médios				397,2 783,5	6,8 491,5

❖Solução ótima sem certificado de otimalidade.
◆ Solução ótima com certificado de otimalidade.
N/D: não disponibilizado na fonte.

soluções ótimas em três e nove exemplos deste conjunto.

É fácil observar a interrupção do algoritmo MP nos exemplos *CW4* e *CW11* em função da limitação imposta no tempo computacional, o que, por sua vez, não ocorreu com MP_AOG dada sua maior velocidade de convergência.

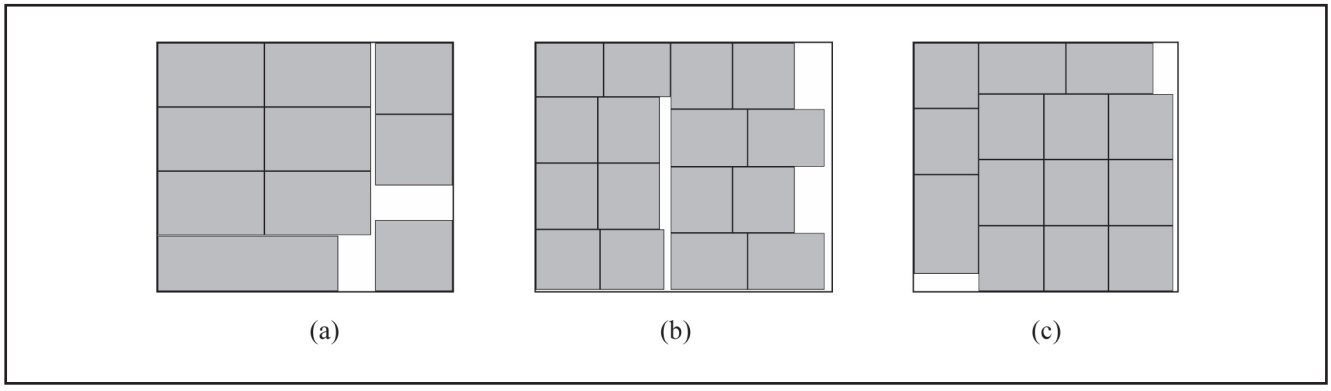
CONCLUSÕES

Neste trabalho foi proposta uma extensão de um método heurístico para geração de padrões de corte bidimensionais guilhotinados restritos, baseado em programação dinâmica. Esta extensão foi analisada em duas variantes (MP e MP_AOG) que têm garantia de gerar padrões ótimos irrestritos (relaxados). A primeira variante (MP) pode ser vista como um refinamento do método SM02 em Silveira e Morabito (2002), utilizando uma heurística de factibilização mais efetiva dentro do procedimento de otimização do subgradiente. Ao ser aplicada para resolver os mesmos exemplos analisados em Silveira e Morabito (2002), MP produziu resultados melhores do que SM02 tanto em termos de qualidade de solução como de esforço computacional.

A segunda variante (MP_AOG) corresponde ao método MP utilizando soluções iniciais de melhor qualidade geradas pela abordagem de busca em grafo-e/ou de Morabito e Arenales (1996). Comparado ao método MP, este refinamento resultou no incremento da qualidade das soluções resultantes e na redução nos tempos médios computacionais. Sua aplicação em vários exemplos da literatura e exemplos gerados aleatoriamente forneceu, na maior parte dos casos, uma solução ótima para o problema, acompanhada de um certificado de garantia de otimalidade. Nos exemplos em que não encontrou soluções comprovadamente ótimas, o método MP_AOG foi, em geral, capaz de produzir um limitante superior relativamente próximo do valor do padrão ótimo restrito, o que permite uma boa estimativa do *gap* de otimalidade. Ao comparar seu desempenho computacional com o de outros métodos da literatura, ele se mostrou bem competitivo para os exemplos analisados neste trabalho.

Como próximos passos desta pesquisa, pretende-se aplicar MP_AOG a outras bases de dados e verificar seu desempenho em relação a outros métodos da literatura. Além disso, serão investigadas novas heurísticas de factibilização que permitam uma melhor utilização dos espaços vazios resultantes da retirada de peças em excesso; acreditamos que o aproveitamento de espaços, ainda que pequenos, pode trazer uma grande diferença na qualidade da solução final. Finalmente, serão também investigadas alternativas para melhorar a convergência da otimização do subgradiente.

Figura 9: Padrões ótimos de exemplos da literatura — conjunto 3: (a) CW4; (b) CW6; (c) CW8.



Artigo recebido em 06/02/2006
Aprovado para publicação em 01/09/2006

Referências

ALVAREZ-VALDÉS, R.; PARAJÓN, A.; TAMARIT, J. A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems. *Computers and Operations Research,* 29, p. 925-947, 2002.

ARENALES, M.; MORABITO, R.; YANASSE, H. Cutting and packing problems. *Pesquisa Operacional,* 19(2), 107-299, 1999.

BEASLEY, J. E. Algorithms for unconstrained two-dimensional guilhotine cutting. *Journal of the Operational Research Society,* 36(4), 297-306, 1985.

BISCHOFF, E.; WAESCHER, G. Cutting and packing. *European Journal of Operational Research*, n. 84, v. 3, 1995.

CHRISTOFIDES, N.; HADJICONSTANTINOU, E. (1995). An Exact Algorithm for Orthogonal 2-D Cutting Problems Using Guillotine Cuts. *European Journal of Operational Research* 83, 21-38.

Referências

CHRISTOFIDES, N.; WHITLOCK, C. An algorithm for two-dimensional cutting problems. *Operations Research,* v. 25, n. 1, p. 30-44, 1977.

CUNG, V., HIFI. M.; LE CUN, B. Constrained two-dimensional guillotine cutting stock problems: A best-first branch-and-bound algorithm. *International Transactions in Operational Research,* v. 7, p. 185-201, 2000.

DAZA, V. P., ALVARENGA, A. G.; DIEGO, J. Exact solutions for constrained two-dimensional cutting problems. *European Journal of Operational Research,* v. 84, p. 633-644, 1995.

DOWSLAND, K.; DOWSLAND, W. Packing problems. *European Journal of Operational Research*, v. 56, p. 2-14, 1992.

DYCKHOFF, H.; FINKE, U. *Cutting and Packing in Production and Distribution: Typology and Bibliography.* Heidelberg: Springler-Verlag, 1992.

DYCKHOFF, H.; SCHEITHAUER, G.; TERNO, J. (1997). *Cutting and Packing*. In: AMICO, M.; MAFFIOLI, F.; MARTELLO, S. (Ed.) *Annoted Bibliographies in Combinatorial Optimization*. New York: John Wiley & Sons, p. 393-414, 1997.

DYCKHOFF, H.; WAESCHER, G. Cutting and packing. *European Journal of Operational Research*, v. 44, n. 2, 1990.

FAYARD, D.; HIFI, M.; ZISSIMOPOULOS, V. An efficient approach for large-scale two-dimensional guillotine cutting stock problems. *Journal of the Operational Research Society,* v. 49, p. 1270-1277, 1998.

HIFI, M. An improvement of Viswanathan and Bagchi's exact algorithm for constrained two-dimensional cutting stock. *Computers and Operations Research,* v. 24, n. 8, p. 727-736, 1997.

HIFI, M. Special issue: Cutting and packing problems. *Studia Informatica Universalis*, v. 2, n. 1, p. 1-161, 2002.

LODI, A.; MARTELLO, S.; MONACI, M. Two-dimensional packing problems: a survey. *European Journal of Operational Research*, v. 141, p. 241-252, 2002.

LUCENA, A. Relax and Cut Algorithms (submetido para publicação), 2004.

MARTELLO, S. Special issue: Knapsack, packing and cutting, Part I: One dimensional knapsack problems. *INFOR*, v. 32, n. 3, 1994a.

MARTELLO, S. Special issue: Knapsack, packing and cutting, Part II: Multidimensional knapsack and cutting stock problems. *INFOR*, v. 32, n. 4, 1994b.

MORABITO, R.; ARENALES, M. Um exame dos problemas de corte e empacotamento. *Pesquisa Operacional,* v. 12, n. 1, p. 1-20, 1992.

MORABITO, R.; ARENALES, M. Staged and constrained two-dimensional guilhotine cutting problems: An AND/OR-graph approach. *European Journal of Operational Research,* v. 94, p. 548-560, 1996.

MORNAR, V.; KHOSHNEVIS, B. A cutting stock procedure for printed circuit board production. *Computers and Industrial Engineering,* v. 32, n. 1, p. 57-66, 1997.

MUKHACHEVA, E. A. *Decision Making under Conditions of Uncertainty: Cutting –packing Problems*. The International Scientific Collection, Ufa, Russia, 1997.

OLIVEIRA, J. F.; FERREIRA, J. S. An improved version of Wang's algorithm for two-dimensional cutting problems. *European Journal of Operational Research,* v. 44, p. 256-266, 1990.

OLIVEIRA, J. F.; WAESCHER, G. Special Issue on Cutting and Packing. *European Journal of Operational Research* (a aparecer), 2006.

PARADA, V.; SEPULVEDA, M.; SOLAR, M.; GOMES, A. Solution for the constrained guillotine cutting problem by simulated annealing. *Computers and Operations Research,* v. 25, n. 1, p. 37-47, 1998.

SICUP *Special Interest Group on Cutting and Packing*. Available in: <http://www.apdio.pt/sicup/>, 2005.

SILVEIRA, R.; MORABITO, R. Um método heurístico baseado em programação dinâmica para o problema de corte bidimensional guilhotinado restrito. *Gestão & Produção,* v. 9, n. 1, p. 78-92, 2002.

SWEENEY, P.; PATERNOSTER, E. Cutting and packing problems: a categorized, application-oriented research bibliography. *Journal of the Operational Research Society*, v. 43, p. 691-706, 1992.

VASKO, F. J. A computational improvement to Wang's two-dimensional cutting stock algorithm. *Computers and Industrial Engineering,* v. 16, n. 1, p. 109-115, 1989.

VIANNA, A. Extensões da abordagem em grafo-e/ou para problemas de corte e empacotamento, *Tese de doutorado*, ICMSC-USP, São Carlos, 2000.

VISWANATHAN, K. V.; BAGCHI, A. Best-first search methods for constrained two-dimensional cutting stock problems. *Operations Research,* v. 41, n. 4, p. 768-776, 1993.

WANG, P. Y. Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems. *Operations Research,* v. 31, p. 573-586, 1983.

WANG, P.Y.; WAESCHER, G. Cutting and packing. *European Journal of Operational Research*, v. 141, n. 2, p. 239-469, 2002.

Agradecimentos

Os autores agradecem aos dois revisores anônimos pelos úteis comentários e sugestões. Esta pesquisa contou com apoio da FAPESP (processo 95/9522-0) e CNPq (processos 522973/95-7 e 307665/2003-8).

Sobre os autores

Reinaldo Morabito
Departamento de Engenharia de Produção
Universidade Federal de São Carlos
End.: Caixa Postal 676 – 13565-905 – São Carlos – SP
Tel.: (16) 3351-8237, r. 216, Fax: (16) 3351-8240
E-mail: morabito@power.ufscar.br

Vitória Pureza
Departamento de Engenharia de Produção
Universidade Federal de São Carlos
End.: Caixa Postal 676 – 13565-905 – São Carlos – SP
Tel.: (16) 3351-8237, r. 277, Fax: (16) 3351-8240
E-mail: vpureza@dep.ufscar.br