PRODUÇÃO

Um refinamento do algoritmo tabu de Dowsland para o problema de carregamento de paletes do produtor

Cintia A. Yamassaki
Vitória Pureza[1]
Departamento de Engenharia de Produção - Universidade Federal de São Carlos
E-mail: vpureza@power.ufscar.br

Resumo

O Problema de Carregamento de Paletes do Produtor consiste em arranjar, ortogonalmente e sem sobreposição, o máximo número de caixas retangulares idênticas de dimensões (l,w) sobre um palete (L,W). Este problema vem sendo tratado com sucesso por heurísticas de blocos, as quais constroem padrões de carregamento com um ou mais blocos, cujas caixas possuem a mesma orientação. Os arranjos gerados por estes métodos estão limitados aos chamados padrões não-guilhotinados de primeira ordem. Neste trabalho, foi elaborada uma implementação baseada no algoritmo de busca tabu de Dowsland, que, ao contrário das heurísticas de blocos, provê arranjos não limitados a padrões particulares de empacotamento. Experimentos computacionais com um conjunto de 34 exemplos extraídos da literatura e de contextos reais indicam que a abordagem é capaz de resolver otimamente a maioria dos exemplos; para os exemplos não resolvidos, é proposto um procedimento adicional simples, cuja aplicação resultou na obtenção de padrões ótimos.

Palavras-chave

Problema de carregamento de paletes, busca tabu, otimização combinatória.

A refinement of Dowsland´s tabu search algorithm for the manufacturer´s pallet loading problem

Abstract

The Manufacturer´s Pallet Loading Problem (MPLP) consists in arranging, orthogonally and without overlapping, the maximum number of rectangular and identical pieces of sizes (l, w) onto a pallet (L, W). The MPLP has been successfully handled by block heuristics, which generate loading patterns with one or more blocks where the pieces have the same orientation. The resulting patterns produced by these methods are limited to the so-called 1st order non-guillotine patterns. In this work we present a tabu search implementation based on the algorithm proposed by Dowsland to solve the MPLP; differently than block heuristics, the solutions are not limited to particular loading patterns. Computational experiments with a set of 34 instances extracted from the literature as well as from practical contexts indicate that this approach is capable to solve most of the examples. For the unsolved instances, we propose a simple additional procedure, which resulted in optimal patterns.

Key words

Pallet loading problem, tabu search, combinatorial optimization.

INTRODUÇÃO

A crescente preocupação com a racionalização das atividades logísticas tem despertado a atenção para o chamado *problema de carregamento de paletes* (*pallet loading problem*) e o interesse na aplicação de técnicas de Pesquisa Operacional para sua resolução. O problema de carregamento de paletes consiste em arranjar, ortogonalmente e sem sobreposição, o máximo número de caixas retangulares com dimensões (l_i,w_i,h_i) sobre um palete de dimensões (L,W,H), onde H é a altura máxima tolerável de carregamento. Este problema surge em várias atividades logísticas tais como o transporte e distribuição de produtos embalados em caixas. Ao se otimizar a utilização de cada palete, mesmo um pequeno número de caixas adicionais pode resultar em economias substanciais.

O problema de carregamento de paletes é visto como uma classe especial dos problemas de Corte e Empacotamento (DYCKHOFF, 1990). Problemas de corte e problemas de empacotamento guardam uma estrutura lógica similar, uma vez que empacotar itens pequenos (caixas) em objetos maiores (paletes) é análogo a cortar o espaço vazio dos objetos maiores (chapas) em partes de espaços vazios, alguns dos quais ocupados pelos itens.

O problema de carregamento de paletes pode ser subdividido em dois tipos: o *problema de carregamento de paletes do produtor* (*manufacturer's pallet loading problem*) e o *problema de carregamento de paletes do distribuidor* (*distributor's pallet loading problem*) (Hodgson, 1982). O problema do distribuidor abrange situações onde as caixas a serem carregadas têm dimensões diferentes (l_i,w_i,h_i), $i=1,..,m$, podendo ser visto como um *problema de carregamento de contêineres* (BISCHOFF; RATCLIFF, 1995; MORABITO; ARENALES, 1994). Em particular, se uma orientação vertical for fixada, por exemplo, se as caixas devem ser carregadas sob a face l_iw_i, então o problema do distribuidor reduz-se ao problema bidimensional de encontrar o arranjo ortogonal de retângulos (l_i,w_i) e (w_i,l_i), $i=1,..,m$, dentro do retângulo (L,W) que maximize a utilização de (L,W).

O problema do produtor, por outro lado, trata do carregamento de produtos embalados em caixas idênticas ($l_i=l$, $w_i=w$ e $h_i=h$, i), as quais estão disponíveis em grandes quantidades. Em cada camada horizontal de carregamento, uma orientação vertical das caixas é fixada. Se nenhuma orientação for fixada *a priori*, isto é, caso as caixas possam ser posicionadas em quaisquer de suas faces (l,w), (l,h) ou (w,h), o problema do produtor pode ser decomposto em dois subproblemas: (i) para cada face das caixas, o problema bidimensional de arranjar ortogonalmente o máximo número de retângulos (i.e., faces das caixas) dentro do retângulo (L,W) (superfície do palete), e (ii) o problema unidimensional de arranjar o máximo número de camadas ao longo da altura H do palete. Note que este último é um *problema da mochila* (*knapsack problem*) com capacidade H. Compilações dos problemas do produtor e distribuidor podem ser encontradas nos artigos e edições especiais de Dowsland; Dowsland (1992), Sweeney; Parternoster (1992), Dyckhoff; Finke (1992), Balasubramanian (1992), Martello (1994), Bischoff; Waescher (1995), Dyckhoff *et al.* (1997), Arenales *et al.* (1999), Wang; Waescher (2002), Hifi (2002) e Martins (2002).

Este trabalho aborda o problema do produtor, especificamente o subproblema bidimensional (i) de se obter um arranjo com o número máximo de retângulos (l,w) e (w,l) no retângulo maior (L,W), resultando em uma camada de altura h. A literatura aponta algumas abordagens exatas para sua resolução (veja, por exemplo, Dowsland, 1987; Tsai *et al*., 1993; Hadjiconstantinou; Christofides, 1995 e Bhattacharya *et al*., 1998). Beasley (1985) trata o caso geral do problema de corte bidimensional com placas de múltiplos tipos, resolvendo a relaxação lagrangiana do problema de programação inteira 0-1 e utilizando-o como limitante em um procedimento de busca em árvore. Mais recentemente, Lins *et al*. (2003) apresentaram uma abordagem capaz de resolver exemplos considerados difíceis por outros métodos; os autores conjecturam que tal abordagem tenha garantia de otimalidade. Pesquisas com limitantes superiores para o problema também podem ser encontradas em Nelissen (1995), Letchford; Amaral (2001) e Morabito; Farago (2002).

Embora existam algoritmos polinomiais para a versão guilhotinada do problema do produtor (TARNOWSKI *et al.*, 1994), o caso mais geral parece ser de difícil resolução do ponto de vista da teoria de complexidade (DOWSLAND, 1987, NELISSEN, 1995, LETCHFORD; AMARAL, 2001). De fato, abordagens exatas são geralmente intratáveis computacionalmente em se tratando de situações práticas. Nesses casos, soluções sem garantia de otimalidade são obtidas através da aplicação de métodos heurísticos.

Nas heurísticas de bloco, por exemplo, são construídos padrões de empacotamento compostos por um ou mais blocos cujas caixas possuem a mesma orientação. Exemplos desses métodos construtivos podem ser encontrados em Steudel (1979), Smith; De Cani (1980), Bischoff; Dowsland (1982), Nelissen (1995), Scheithauer; Terno (1996) e Morabito; Morales (1998). Apesar da alta qualidade das soluções geradas, a aplicação de heurísticas de bloco invariavelmente resulta em um tipo particular de padrão, o chamado padrão não-guilhotinado de 1ª ordem (veja em Arenales; Morabito, 1995, a descrição de tipos

de padrões de carregamento). Portanto, problemas cuja solução ótima não corresponde a padrões não-guilhotinados de 1ª ordem (padrões de ordem superior) não podem ser resolvidos otimamente com heurísticas de bloco. Farago; Morabito (2000) apresentam um método heurístico baseado em relaxação lagrangiana e *surrogate* que por não estar sujeito à limitação de tipos particulares de padrão, mostrou-se capaz de resolver otimamente problemas da literatura e extraídos de contextos práticos.

Nos últimos 30 anos, a abundância de problemas de otimização de difícil resolução encontrados em contextos práticos motivou o desenvolvimento das chamadas meta-heurísticas. Estas técnicas resultam da adaptação de idéias de outras áreas do conhecimento de forma que a definição de sua estrutura deriva de conexões muitas vezes inesperadas. *Simulated annealing* (KIRKPATRICK *et al.*, 1983), por exemplo, é baseada no processo físico de recozimento de sólidos; algoritmos genéticos (HOLLAND, 1975) procuram emular aspectos da reprodução evolutiva; GRASP (FEO; RESENDE, 1995) realiza amostragens inteligentes do espaço de solução, e busca tabu (GLOVER; LAGUNA, 1997) incorpora conceitos selecionados de inteligência artificial. Qualquer que seja a técnica utilizada, o propósito destes métodos é o de prover à busca heurística elementos que permitam a superação da otimalidade local e a obtenção de soluções de qualidade superior.

Apesar do enorme sucesso que meta-heurísticas têm alcançado em uma variedade de problemas, no que diz respeito ao problema de carregamento de paletes do produtor, poucas aplicações são reportadas na literatura. Dowsland (1993) apresenta alguns experimentos com uma implementação de *simulated annealing*, onde são investigados a efetividade do método, a estrutura de vizinhança mais apropriada e os melhores parâmetros a serem utilizados. Herbert; Dowsland (1996) apresentam uma família de algoritmos genéticos para o problema, onde o espaço de soluções inclui tanto soluções factíveis como infactíveis. Pureza; Morabito (2003) propõem uma implementação de busca tabu simples em heurísticas de bloco. Partindo-se de uma solução inicial, são realizados movimentos de aumento de blocos, que resultam na diminuição, eliminação e criação de outros blocos. Resultados computacionais indicam que a abordagem é capaz de gerar rapidamente padrões ótimos de ordem superior em problemas de difícil resolução para outros métodos da literatura.

Este estudo baseia-se no algoritmo de busca tabu *thresholding* proposto por Dowsland (1996) para resolução do problema do produtor. Partindo-se de uma solução com caixas sobrepostas, porém em número ótimo, são realizados movimentos destas caixas para novas posições com vistas à obtenção de sobreposição zero (padrão ótimo). Foi inicialmente implementada uma versão do algoritmo original, e para os problemas em que o algoritmo básico não encontrou a solução ótima, propôs-se um procedimento adicional simples para factibilização que busca melhorar o empacotamento das caixas em áreas de maior sobreposição.

A TÉCNICA TABU THRESHOLDING

Heurísticas de busca em vizinhança são métodos que, partindo de uma solução inicial x, investigam modificações na estrutura da solução e que resultam em novas soluções. O conjunto de soluções que pode ser obtido pela aplicação destas modificações ou movimentos é chamado de vizinhança $N(x)$. Heurísticas de busca em vizinhança selecionam a cada iteração o movimento que resulta em uma solução vizinha com maior qualidade que a solução anterior. Um exemplo clássico é o problema de programação de n tarefas em 1 máquina. Cada tarefa j está associada a um tempo de processamento p_j e data de entrega d_j; o objetivo é obter o seqüenciamento que minimize a função $T = \Sigma_j \text{ Max}(C_j - d_j, 0)$, onde C_j é o instante em que a tarefa j é completada. Neste caso, uma opção de estrutura de vizinhança consiste em trocas de pares de tarefas de suas posições originais. A cada iteração, computam-se as $n(n\text{-}1)/2$ trocas e seleciona-se aquela que resulta na permutação de tarefas com o maior ganho na função objetivo. Invariavelmente, este processo produz uma solução que não pode ser melhorada pela aplicação da estrutura de vizinhança selecionada. Ou seja, atinge-se um ótimo local.

A busca tabu (GLOVER; LAGUNA, 1997) é uma técnica que se superpõe a heurística de busca com o propósito de guiar o processo além da otimalidade local. Sua principal característica consiste na armazenagem e utilização de atributos das soluções mais recentemente visitadas, de forma a permitir que escolhas estratégicas sejam realizadas ao longo do processo. A proibição de movimentos que resultem em soluções com tais atributos pode não só impedir que as mesmas sejam revisitadas (ciclagem), contribuindo assim para a continuidade das explorações, como também evitar que as próximas soluções geradas possuam estes atributos. Tais restrições são mantidas por um número limitado de iterações durante o qual os atributos são ditos tabu-ativos. No exemplo de programação de tarefas, um atributo possível é a posição k que a tarefa i trocada ocupava na permutação antes da realização do movimento. Ao se proibir, por um número t de iterações, movimentos cujos atributos impliquem no retorno da tarefa i à posição k, impede-se que soluções com i em k sejam geradas e, no caso particular, que a

permutação anterior seja novamente obtida. É possível também estimular tais características se as mesmas estão associadas a soluções de alta qualidade (intensificação), ou seja, favorecer movimentos que mantenham certas tarefas em determinadas posições. Pode-se também desestimular atributos muito freqüentemente encontrados a fim de promover a busca em outras regiões do espaço (diversificação).

A utilização destas estratégias resulta na necessidade de definição prévia dos atributos a serem restringidos/estimulados e das estruturas ou funções de memória onde são armazenados os valores destes atributos. Segundo Dowsland (1996), em problemas como o problema de carregamento de paletes do produtor, a definição destes elementos não é trivial, o que resultaria em funções de memória bastante complexas. Por esta razão, nesse trabalho foi empregada a técnica de tabu *thresholding*, uma variante sem memória da busca tabu, mas que procura emular seus efeitos.

A técnica de tabu *thresholding* foi proposta visando à exploração agressiva do espaço da busca de modo não monotônico. Ou seja, alternam-se fases onde apenas movimentos que resultem na melhoria do valor da função objetivo da solução corrente podem ser realizados (*fases de melhoria*) com fases onde movimentos de degradação da qualidade da solução também são permitidos (*fases de mix*). Efeitos de busca não monotônica podem ser realizados deterministicamente ou probabilisticamente; no caso de tabu *thresholding*, utilizam-se elementos de busca tabu probabilísticos. Especificamente, utiliza-se aleatorização controlada para cumprir certas funções originalmente desempenhadas pela memória, como o controle da seleção de movimentos a serem investigados. Este processo provoca resultados semelhantes aos das restrições de busca tabu.

Um procedimento tabu *thresholding* de âmbito geral é apresentado a seguir.

Fase de Melhoria

1.1 Gere um subconjunto S dentre os movimentos disponíveis e faça S^* igual ao conjunto de melhores movimentos de melhoria em S. Se $S^* = \phi$ e S não consiste de todos os movimentos disponíveis, aumente S adicionando novos subconjuntos de movimentos até que $S^* \neq \phi$ ou todos os movimentos sejam incluídos em S.

1.2 Se $S^* \neq \phi$, escolha o *melhor movimento probabilístico* (definido a seguir) de S^* para gerar uma nova solução e retorne ao passo 1.1. Caso contrário, vá para a fase de *mix*.

Fase de *Mix*

2.1 Selecione aleatoriamente um número inteiro t entre t_{min} e $t_{máx}$.

2.2 Gere um subconjunto S dentre os movimentos disponíveis e selecione o melhor movimento probabilístico em S, gerando uma nova solução.

2.3 Repita o passo 2.2. por t iterações ou até que um critério de aspiração seja satisfeito, e retorne para a fase de melhoria.

O conjunto S nestas fases pode consistir de todos os movimentos disponíveis em problemas pequenos ou onde os movimentos podem ser gerados e avaliados com pequeno esforço computacional. A escolha dos movimentos utiliza estratégias de lista de candidatos (definida a seguir), que isola subconjuntos de movimentos a serem examinados a cada iteração, e também utiliza o critério de melhor probabilístico. O critério do melhor probabilístico consiste em extrair de S ou S^* os k melhores movimentos aos quais são atribuídas probabilidades de seleção de acordo com a qualidade da solução resultante.

O efeito do critério de escolha é ainda influenciado pelo parâmetro t, que determina o número de iterações na fase de *mix* e é análogo a armazenar um período em que o *status* tabu é mantido. Quanto maior o valor de t, maior é a diversificação imposta ao processo. O controle sobre o parâmetro t é obtido selecionando-se limitantes inferiores e superiores para este valor, e este parâmetro pode variar aleatoriamente entre estes limites, ou de acordo com alguma distribuição selecionada. Critérios de aspiração são situações que interrompem este processo. Como a melhor solução obtida é atualizada ao longo da busca, é natural utilizar um critério que provoca o retorno à fase de melhoria quando é gerada uma solução de qualidade superior à melhor já obtida.

Admite-se que a solução inicial é gerada através de uma heurística ou aleatoriamente. O término do procedimento ocorre depois de um número predeterminado de iterações ou por outro critério de parada (por exemplo, número de iterações sem melhoria).

Lista de Candidatos

A base de uma lista de candidatos consiste em subdividir os movimentos a serem examinados em subconjuntos. Seja $MS(x)$ o conjunto de todos os movimentos associados a uma dada solução x, isto é, o conjunto de movimentos que podem transformar x em $x' \in N(x)$. Cada movimento tem um dado valor, onde o mais alto valor corresponde ao movimento mais atraente.

Para criar a lista de candidatos, divide-se $MS(x)$ em subconjuntos indexados $MS(1,x)$, $MS(2,x)$, ..., $MS(m, x)$. Para um dado problema, $MS(i,x)$ identifica todos os movimentos associados ao subconjunto indexado pelo índice i ($\forall i = 1...m$).

A lista de candidatos é cíclica e pode ser investigada ordenando-se aleatoriamente blocos de movimentos. O conjunto *M* é dividido em blocos sucessivos, cada um contendo um pequeno número de índices relativos a *m* subconjuntos de movimentos. Quando um dado bloco é tomado para investigação, reordenam-se seus elementos aleatoriamente antes de examiná-los, mudando-se assim a seqüência desta porção de *M*.

A estratégia da lista de candidatos opera da seguinte maneira:

1. Inicie com o conjunto *R* (conjunto de movimentos rejeitados) = ϕ e faça *i*=0.
2. Incremente *i* fazendo *i*=*i*+1 (se *i*>*m*, faça *i*=1) e selecione, do subconjunto *MS*(*i*,*x*), o movimento candidato por uma regra de seleção predefinida.
3. Se o movimento candidato for aceitável, execute-o gerando uma nova solução. Faça *R* = ϕ e retorne ao passo 2.
4. Se o movimento candidato é inaceitável, adicione *i* a *R*. Se *R*=*M* pare, pois todos os subconjuntos foram examinados e nenhum movimento candidato foi aceito. Caso contrário, retorne ao passo 2.

A Figura 1 ilustra uma situação hipotética onde o conjunto de movimentos foi dividido em *m*=5 subconjuntos e cujo tamanho adotado de blocos é *k*=2. O primeiro bloco (Figura 1.a) a ser analisado é formado pelos subconjuntos 1 e 2, que após o embaralhamento serão investigados na ordem 2-1. A seguir (Figura 1.b), é tomado o bloco formado pelos subconjuntos 3 e 4 (ordem 3-4). Na Figura 1.c, temos o bloco formado por 5 e 2, que será investigado na ordem 2-5.

O índice *i* na lista de busca identifica a porção de *M* analisada, isto é, o bloco que está sendo investigado. A menos que o número de elementos em um bloco seja divisível por *m*, a utilização de blocos de busca permite que os elementos gradualmente migrem ao longo da lista devido à alteração da seqüência de investigação. Observe na Figura 1.c que o subconjunto 2 migrou para um bloco diferente do que pertencia no início do processo. Entretanto, em quaisquer duas investigações sucessivas de *M*, um dado elemento será investigado em aproximadamente m movimentos após a investigação anterior. Esta estratégia procura emular, portanto, o efeito de proibições de atributos de movimentos em sistemas que utilizam estruturas de memória.

O ALGORITMO DE DOWSLAND (1996)

O problema foi tratado como o de dispor um número predeterminado de caixas sobre o palete. Em todos os experimentos reportados é utilizado o número ótimo de caixas (obtidos a partir dos resultados de Dowsland, 1984). Em outras palavras, existe a garantia de que as caixas podem ser arranjadas no palete sem sobreposição. Em situações mais gerais, em que não é conhecido o número de caixas ótimo, um limitante superior (como o apresentado por Nelissen, 1995) pode ser utilizado com uma pequena modificação no procedimento, a ser descrita no fim desta seção.

O conjunto de soluções factíveis é definido como o conjunto de padrões que contém tal número de caixas. Especificamente, sobreposições são permitidas. A vizinhança de uma dada solução *S* é definida como o conjunto de soluções obtidas ao se mover uma única caixa para uma nova posição. A cada iteração é computada a vizinhança (total ou parcial) e selecionado o movimento que resulta na maior redução da sobreposição total. O custo *f*(*S*) da solução *S* é definido como a somatória das sobreposições entre cada par de caixas. O objetivo é reduzir ao valor mínimo de zero, ou seja, sobreposição nula. A solução inicial foi gerada aleatoriamente.

Em seus experimentos, Dowsland propôs e utilizou uma redução do número de posições da coordenada inferior esquerda (x_1,y_1) das caixas, sem perda de otimalidade. Também foram investigadas algumas possibilidades de indexação dos subconjuntos e variações na amostragem dos movimentos. Este trabalho está limitado à variação que utiliza indexação dos subconjuntos por caixas por ter apresentado melhores resultados. Isso significa que a vizinhança foi particionada em *n* subconjuntos, cada um correspondendo aos possíveis reposicionamentos de cada caixa. Também foram permitidos apenas movimentos de caixas com sobreposição durante a fase de *mix*, já que desta forma não se perde tempo movendo-se caixas de uma posição para outra e depois movendo-as de volta para as posições originais em um novo ciclo.

No trabalho de Dowsland, adotou-se *t*=5 e uma série de experimentos iniciais resultou na definição dos valores dos seguintes parâmetros: número de blocos *k*=5 (se *n* não for divisível por 5), 7 (se *n*=30) e 6 (outros);

Figura 1: Ordem de investigação dos subconjuntos de movimentos.

amostragem (durante a fase de *mix*) p =10, 50, 90 e 100%.

A amostragem parcial durante a fase de *mix* permite que se reduza a chance de ciclagem. Quanto menor p, maior a chance da busca fazer movimentos maiores de subida nesta fase. Portanto, p determina a facilidade com a qual a busca é capaz de subir vales no espaço de solução, emergindo assim dos ótimos locais.

No trabalho de Dowsland, dois conjuntos de 100 problemas aleatórios foram gerados, o primeiro na faixa de 10-19 caixas e o segundo na faixa de 20-30 caixas. Em média, o melhor valor de p obtido foi de 50% em ambos os casos e permitiu que com 1000 movimentos realizados (número máximo de iterações), 100% dos problemas com 10-20 caixas e 91% dos problemas com 20-30 caixas fossem resolvidos. Ou seja, apenas 9 problemas não foram resolvidos. A seguir, é apresentado o pseudocódigo do algoritmo.

Passo 1: Inicialização

1.1 Dados de entrada e definição de variáveis:

(L,W): dimensões do palete;	it_{max}: nº máximo de iterações;
(l,w): dimensões das caixas;	*cont*: nº de iterações consecutivas sem melhoria;
n: nº ótimo de caixas;	*cont_mix*: nº de iterações na fase de *mix*;
L_t: lista cíclica com n posições;	$f(.)$: valor da sobreposição de qualquer solução;
k: tamanho do bloco de busca;	S_o: solução inicial;
fase: melhoria ou *mix*;	S: solução corrente;
it: nº de iterações;	S^*: melhor solução candidata, do subconjunto investigado;
t: nº de iterações da fase de *mix*;	*incumbente*: solução com menor sobreposição encontrada até o momento.

1.2. Gere as posições factíveis para as caixas no palete.
1.3. Gere aleatoriamente uma solução inicial S_o. Faça $S = S_o$, *it*=0 e *incumbente*= S_o.
1.4. Crie a lista cíclica L_t e associe cada caixa i à posição i na lista L_t.
1.5. Agrupe as caixas da lista L_t em blocos de tamanho k.
1.6. Faça *fase* = melhoria e vá para o passo 2.

Passo 2: Fase de melhoria

2.1 Repita
  2.1.1 Embaralhe o bloco corrente e obtenha a seqüência de investigação dos subconjuntos.
  2.1.2 Tome primeiro subconjunto não analisado da seqüência e obtenha S^*.
  2.1.3 Se f(S^*)<f (*incumbente*) então execute o movimento, faça *incumbente*=S^* e *it*=*it*+1. Caso contrário, faça *cont*=*cont*+1 e vá para o passo 2.1.2.

Até que *cont* ≥ n ou *it* ≥ it_{max}.
2.2. Se *it* ≥ it_{max} ou se *incumbente*=0 vá para o passo 4.
2.3. Se *cont* ≥ n faça *fase*=*mix*, *cont*=0 e vá para o passo 3.

Passo 3: Fase de *mix*

3.1. Repita
  3.1.1. Execute o movimento, faça S=S^*, *cont_mix* = *cont_mix* +1 e *it*=*it*+1;

Até que *cont_mix* ≥ t ou *it* ≥ it_{max}.
3.2. Se *it* ≥ it_{max} ou se *incumbente*=0 vá para o passo 4.
3.3. Se *cont_mix* ≥ t faça *fase*=melhoria, *cont_mix*=0 e vá para o passo 2.

Passo 4: Final do procedimento

4.1. Se *incumbente*=0 uma solução ótima foi encontrada. Caso contrário, a solução ótima não foi encontrada e a solução final é infactível quanto à sobreposição.

Caso a sobreposição não seja reduzida a zero, uma possível ação é a retirada de uma caixa e a reexecução do algoritmo até que uma solução sem sobreposição seja encontrada. Nos exemplos tratados neste trabalho, o número ótimo de caixas era conhecido, de forma que a solução resultante seria necessariamente subótima. No caso em que se dispõe de um limitante superior para o número de caixas, a garantia de otimalidade da solução é perdida.

EXPERIMENTOS E RESULTADOS COMPUTACIONAIS

O algoritmo foi implementado em Borland Delphi 4.0 e executado em um microcomputador Pentium II, 333 MHZ, 64 Mbytes de RAM. Foram considerados como critérios de parada, soluções com custo zero (sobreposição nula), ou um número máximo de 1000 iterações.

Primeira bateria de testes — Experimentos com a implementação do algoritmo de Dowsland

O trabalho de Dowsland tem como objetivo principal fazer comparações metodológicas com o algoritmo de *simulated annealing* (DOWSLAND, 1993) de maneira

que muitas das decisões sobre o espaço de solução como estrutura de vizinhança, solução inicial e função custo foram mantidas. Além disso, o valor de t=5 seguiu a orientação de outros autores. Por esta razão, no presente trabalho decidiu-se investigar a utilização de outros valores de parâmetros, em especial dos parâmetros p e t.

Trinta exemplos até 30 caixas foram selecionados na primeira bateria de experimentos. Destes, foram selecionados 8 exemplos-teste, sendo 4 deles da faixa de 10 a 19 caixas e 4 de 20 a 30 caixas. Estes exemplos foram testados para 10 diferentes valores do parâmetro p (porcentagem de vizinhança analisada) com o objetivo de definir as melhores combinações para os demais problemas.

Experimentos reportados pela autora, e outros realizados preliminarmente neste trabalho, sugeriram a utilização do parâmetro t dentre os valores 3, 5, 10 e 15. Para todos os exemplos, o algoritmo foi executado a partir de diferentes soluções iniciais geradas aleatoriamente e pós-processadas, resultando em um total de 800 execuções com diferentes combinações de parâmetros e soluções iniciais. Soluções iniciais geradas aleatoriamente são construídas colocando-se as caixas aleatoriamente no palete; já as pós-processadas consistem em mover cada caixa para a melhor posição no palete (gerando menor sobreposição total). Analisando-se o desempenho destes problemas, os parâmetros t, p e o tipo de solução inicial foram escolhidos e fixados em t=3 e t=5, p variando de 10% a 60% e solução inicial pós-processada.

Tendo fixado a faixa de valores de parâmetros e o tipo de solução inicial, foram realizados experimentos nos demais problemas selecionados. Metade destes problemas tinha a solução ótima entre 10 e 19 caixas, e a outra metade entre 20 e 30 caixas. Todos foram executados até que uma solução ótima fosse encontrada, ou então até que todos as variações de parâmetros fossem executadas sem que se encontrasse a solução do problema.

Os problemas foram coletados em Dowsland (1996), em Morabito; Morales (1998) e em Farago; Morabito (2000). Dentre estes, encontram-se 9 problemas descritos por Dowsland como de difícil resolução, já que a pesquisadora não conseguiu resolvê-los com nenhuma das variações paramétricas adotadas. Os problemas cujo palete apresenta dimensão de 120x100cm (PBR) foram coletados de Farago; Morabito (2000) e consistem de problemas reais registrados no Centro de Distribuição da Transportadora Americana em Campinas, SP. A Tabela 1 apresenta as características dos problemas com número ótimo na faixa de 20 a 30 caixas.

O algoritmo foi aplicado a estes problemas, e a partir dos resultados, foram agrupados em duas categorias: problemas *fáceis* e *outros*. Os problemas foram considerados fáceis quando a solução era encontrada logo na primeira execução. Já *outros* engloba os 9 citados como difíceis por Dowsland (1996) e os demais onde foi necessário um maior número de execuções para se encontrar a solução. Dos 30 problemas, todos os entre 10-19 caixas foram classificados como *fáceis*, assim como 5 dos problemas entre 20-30 caixas. Estes 5 problemas foram

Tabela 1: Problemas com número ótimo entre 20 e 30 caixas.

PROBLEMA	(L,W)	(L,W)	# DE CAIXAS	FONTE
E16	(120,100)	(26,15)	20	Farago; Morabito (2000)
E17	(120,100)	(28,17)	23	Farago; Morabito (2000)
E18	(120,100)	(35,12)	26	Farago; Morabito (2000)
E19	(120,100)	(24,19)	26	Farago; Morabito (2000)
E20	(19,18)	(5,3)	22	Dowsland (1996)
E21	(120,100)	(32,18)	20	Farago; Morabito (2000)
E22	(30,16)	(8,3)	20	Dowsland (1996)
E23	(19,16)	(5,3)	20	Dowsland (1996)
E24	(29,16)	(7,3)	22	Dowsland (1996)
E25	(22,16)	(5,3)	23	Morabito; Morales (1998)
E26	(31,19)	(8,3)	24	Dowsland (1996)
E27	(20,17)	(7,2)	24	Dowsland (1996)
E28	(36,22)	(11,3)	24	Dowsland (1996)
E29	(33,26)	(11,3)	26	Dowsland (1996)
E30	(27,25)	(8,3)	28	Dowsland (1996)

provavelmente resolvidos pelo fato de os experimentos envolverem um número maior de combinações paramétricas relevantes que aquelas utilizadas por Dowsland. Em termos de tempos computacionais, a obtenção do padrão ótimo requereu em média 15 segundos de CPU.

Na categoria *outros*, 10 problemas entre 20-30 caixas não foram resolvidos na primeira execução. Os problemas desta categoria foram então tratados com todas as variações de parâmetros (e não apenas com os valores sugeridos a partir dos problemas-teste). Para os parâmetros selecionados, o algoritmo não encontrou a solução em apenas um dos problemas (E22). Neste caso específico, reexecutamos o algoritmo com t=7, obtendo-se a solução ótima após 31 execuções correspondentes às variações das soluções iniciais e demais parâmetros. Em termos de tempos computacionais, esta classe de problemas requereu em média cerca de 1,7 hora de CPU, considerando todas execuções até a obtenção do ótimo global.

Foram então selecionados 4 problemas adicionais, desta vez com mais de 30 caixas e descritos na Tabela 2. Devido ao maior porte destes problemas, há um maior número de movimentos disponíveis na vizinhança, o que sugeriu que uma maior diversificação deveria ser aplicada para evitar que soluções sejam revisitadas. Optou-se, portanto, por variar o valor de *t* entre 3, 5 e 7.

Os problemas foram executados até que uma solução ótima fosse encontrada. Para os problemas com 31 e 32 caixas, foi encontrada a solução ótima, requerendo-se em média 8 horas de CPU. Para os problemas de 41 e 42 caixas não foi possível encontrar a solução ótima utilizando-se todas as variantes previamente selecionadas.

Segunda bateria de testes — Experimentos com a inclusão do procedimento para factibilização

Dada a dificuldade do algoritmo obter padrões ótimos, particularmente em exemplos com maior número ótimo de caixas, foi desenvolvido um procedimento para factibilização das soluções. O procedimento consiste nos seguintes passos:

1. Identificar a caixa com maior sobreposição (caixa i).
2. Retirar a caixa i e todas as caixas adjacentes a ela.
3. Recolocar as caixas retiradas em posições de menor sobreposição total.
4. Prosseguir com a execução do algoritmo na fase de melhoria a partir deste ponto da busca.

Após um determinado número de iterações sem que a melhor solução seja atualizada, este procedimento é chamado. A primeira execução do procedimento é aplicada a partir da melhor solução obtida até o momento. Nas demais chamadas do procedimento, a melhor solução é recuperada apenas se ela foi atualizada após a última chamada do procedimento para factibilização. Nos casos em que a melhor solução não foi atualizada desde a última tentativa para factibilização, o procedimento é aplicado à solução corrente. A recuperação da melhor solução tem como ponto a favor a melhor qualidade da solução, mas experimentos preliminares indicaram que a sua recuperação a cada chamada do procedimento pode levar à ciclagem se a mesma não tiver sido atualizada.

O *design* deste procedimento envolve ainda a definição do intervalo (número de iterações) entre duas aplicações do procedimento e se o número de execuções deste procedimento deve ou não ser limitado. Aplicações muito freqüentes podem ser desvantajosas na medida que impedem a estabilização da busca, enquanto que aplicações pouco freqüentes podem não trazer os efeitos desejados. Após experimentos preliminares, adotou-se que o procedimento para factibilização fosse aplicado a cada 100 iterações sem que a melhor solução fosse atualizada. Além disso, não se limitou o número de aplicações.

Outra questão diz respeito à recolocação das caixas. Em cada reposicionamento, é selecionada a posição para a coordenada inferior esquerda (x_1,y_1) da caixa com a *menor* ou a *maior* distância da coordenada de origem (0,0) do palete (ou seja, onde $d=((x_1)^2+(y_1)^2)^{1/2}$ seja máximo ou mínimo). A seleção do critério menor ou maior distância é feita aleatoriamente. A principal motivação

Tabela 2: Problemas com número ótimo acima de 30 caixas.

PROBLEMA	(*L*, *W*)	(*l*, *w*)	# DE CAIXAS	FONTE
E31	(120,100)	(22,17)	31	Farago; Morabito (2000)
E32	(120,100)	(25,15)	32	Farago; Morabito (2000)
E33	(57,44)	(35,12)	41	Morabito; Morales (1998)
E34	(86,82)	(15,11)	42	Morabito; Morales (1998)

para a proposição deste procedimento deve-se ao fato de a heurística de Dowsland posicionar caixas em coordenadas de menor sobreposição, porém sem considerar explicitamente a qualidade do empacotamento resultante. Assim, a retirada de caixas de áreas consideradas mais críticas, seguida de uma recolocação que procura utilizar mais eficientemente o espaço vazio, pode permitir que a factibilização seja alcançada.

Após a recolocação de todas as caixas, prossegue-se com a execução do algoritmo na fase de melhoria a partir deste ponto da busca. A seguir, é apresentado o procedimento para factibilização proposto.

Procedimento para factibilização

Passo 1: Retirada de caixas

1. Após um número previamente especificado de iterações sem atualização da melhor solução:
   1.1 Recupere a melhor solução caso a mesma tenha sido obtida após a última chamada do procedimento para factibilização. Caso contrário, mantenha a solução corrente.
   1.2 Da solução selecionada no passo 1.1, selecione a caixa com maior sobreposição.
   1.3 Retire a caixa selecionada e todas as caixas adjacentes a ela.

Passo 2: Recolocação de caixas

2. Para cada caixa retirada no passo 1.3:
   2.1 Obtenha as posições de reposicionamento da caixa que resultem na menor sobreposição.
   2.2 Se houver mais de uma posição com este valor de sobreposição, ordene as posições da menor para a maior distância da origem (0,0) do palete.
   2.3 Escolha aleatoriamente o critério de posicionamento (maior ou menor distância da origem), obtenha a posição que atende a este critério e reposicione a caixa.

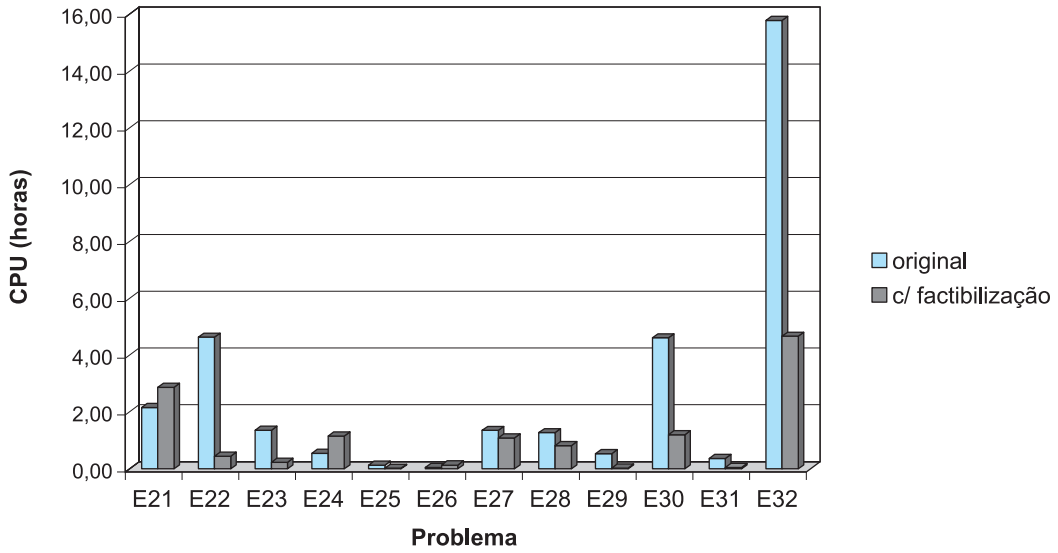
Passo 3: Continuidade

3. Faça *fase*=melhoria e prossiga a partir deste ponto da busca.

Experimentos preliminares foram realizados aplicando-se o procedimento para factibilização aos problemas da categoria *outros*, já resolvidos anteriormente (E21 a E30). Estes experimentos foram feitos com vistas a analisar sua eficácia e, em caso positivo, comparar o desempenho do algoritmo antes e depois de sua incorporação. O procedimento para factibilização não foi aplicado aos problemas da categoria fáceis, já que os tempos de resolução com o algoritmo original foram considerados pequenos. A Figura 2 apresenta os tempos *da execução* que resultou na solução ótima antes e depois da aplicação do procedimento.

Todos os problemas desta classe (E21 a E30) foram resolvidos. À primeira vista, a introdução do procedimento parece ter atrasado a obtenção da solução ótima, já que foi computado um aumento médio de 19% nos tempos de CPU em relação aos resultados do algoritmo original. Entretanto, se considerarmos o tempo total

Figura 2: Tempo da execução que resultou na solução ótima.



dispendido nas variações paramétricas até a obtenção destas soluções, observamos melhorias consideráveis no número total de execuções. Em média, este número foi reduzido para 47% do número de execuções anteriormente necessário. Ou seja, o procedimento para factibilização de fato reduziu a sensibilidade do algoritmo original à solução inicial e aos valores dos parâmetros tabu (veja Figura 3).

Para os problemas de 31 e 32 caixas (veja na Figura 3, problemas E31 e E32), já resolvidos pelo algoritmo original, o número de execuções com o procedimento para factibilização foi reduzido para 44% do requerido anteriormente. Assim como no caso anterior, todos os problemas foram resolvidos.

Para os problemas não resolvidos anteriormente (E33 e E34), os resultados foram promissores. Anteriormente, com todas as variações de parâmetros selecionadas e após um total de 180 execuções para cada problema (resultando em um tempo médio de 144 horas), nenhuma solução ótima foi encontrada Com a incorporação da heurística para factibilização, estes problemas foram resolvidos em 1954 e 207 segundos, respectivamente. Os tempos totais (considerando todas as variações paramétricas anteriores) foram em média 34 horas. Grande parte desse esforço computacional deve-se também ao cálculo da vizinhança. Isso sugere que, além de melhorias metodológicas, a incorporação de estruturas computacionais mais eficientes e mecanismos de redução da vizinhança são cruciais para sua aplicabilidade.

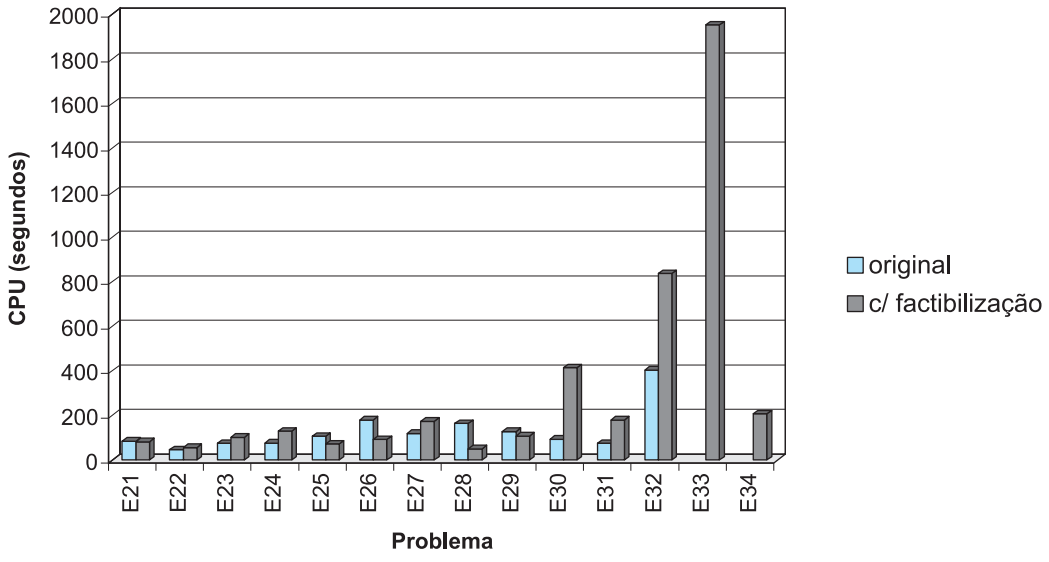
CONCLUSÕES E PERSPECTIVAS

Neste trabalho foi investigada a utilização da metaheurística tabu na resolução do problema de carregamento de paletes do produtor. Em especial, uma versão do algoritmo tabu *thresholding,* proposto por Dowsland (1996) foi implementada e testada com diferentes valores de parâmetros e soluções iniciais para um conjunto de 34 exemplos com um número ótimo de 10 a 42 caixas. Para os exemplos em que o algoritmo original não encontrou um padrão ótimo, foi proposto um procedimento adicional para factibilização das soluções.

No que diz respeito aos resultados obtidos, observa-se que os tempos computacionais para problemas com maior número de caixas foram bastante altos, uma vez que um conjunto de variações paramétricas foi testado até que se atingisse um ótimo global. Neste sentido, é importante ressaltar que o algoritmo tabu de Dowsland foi utilizado com o objetivo de fazer comparações de cunho metodológico com um algoritmo de *simulated annealing*. Ou seja, não houve a intenção de apresentar um algoritmo realmente competitivo. O grande esforço computacional encontrado inviabilizou o tratamento de problemas considerados difíceis na literatura, cujo número de caixas é ainda maior que os aqui reportados (MORABITO; MORALES, 1998).

Não obstante, é importante considerar que o método apresentado possui um grande atrativo. Nenhuma estrutura para geração do padrão de empacotamento de caixas (como as encontradas em heurísticas de blocos)

Figura 3: Tempo total de execução.



é necessária, o que permite, teoricamente, que problemas cujo padrão ótimo é não-guilhotinado de ordem superior sejam resolvidos. Tal flexibilidade sugere que investigações metodológicas adicionais com vistas à melhoria da eficiência do método sejam bastante bem-vindas.

Neste sentido, foi aqui proposto um artifício simples para factibilização, envolvendo a retirada da caixa com maior sobreposição e suas adjacentes, seu posterior rearranjo no palete e a reexecução do algoritmo. Sua utilização permitiu resolver problemas que não tinham sido resolvidos por nenhuma das variantes do algoritmo original. Este mesmo procedimento foi empregado na resolução dos problemas já resolvidos, porém com tempo computacional alto, apresentando uma melhoria considerável no número de execuções necessárias. Em média, o número de execuções caiu para cerca de 46% da média anterior.

Possíveis caminhos de pesquisa envolvem a investigação de procedimentos alternativos para factibilização e de redução inteligente da vizinhança. Além disso, o algoritmo pode ser facilmente estendido para o tratamento do Problema de Corte Bidimensional Restrito, no qual placas de diferentes dimensões (l_i,w_i) e demandas restritas devem ser cortadas a partir de chapas de dimensões (L,W). Como o algoritmo utiliza indexação por caixas (ou placas), é empregada uma estrutura de dados em que são armazenados os atributos das caixas/placas tais como orientação e coordenadas de posicionamento no palete/chapa. Portanto, a adição dos atributos (l_i,w_i) a esta estrutura pode ser facilmente incorporada.

Artigo recebido em 15/04/2002
Aprovado para publicação em 22/05/2003

Referências Bibliográficas

ARENALES, M.; MORABITO, R. An and/or-graph approach to the solution of two-dimensional non-guillotine cutting problems. *European Journal of Operational Research*, vol. 84, p. 599-617, 1995.

ARENALES, M.; MORABITO, R.; YANASSE, H. Special issue: Cutting and packing problems. *Pesquisa Operacional*, vol. 19, n. 2, p. 109-284, 1999.

BEASLEY, J. E. An exact two-dimensional non-guillotine cutting tree search procedure. *Operations Research*, vol. 33, n. 1, pp. 49-64, 1985.

BALASUBRAMANIAN, R. The pallet loading problem: A survey. *International Journal of Production Economics*, vol. 28, p. 217-225, 1992.

BHATTACHARYA, S.; ROY, R.; BHATTACHARYA, S. An exact depth-first algorithm for the pallet loading problem. *European Journal of Operational Research*, vol. 110, n. 3, p. 612-625, 1998.

BISCHOFF, E.; DOWSLAND, W. B. An application of the micro product design and distribution. *Journal of Operational Research*, vol. 84, n. 3, p. 271-280, 1982.

BISCHOFF, E.; RATCLIFF, M. Loading multiple pallets. *Journal of the Operational Research Society*, vol. 46, p. 1322-1336, 1995.

BISCHOFF, E.; WAESCHER, G. Special issue on cutting and packing. *European Journal of Operational Research*, vol. 84, n. 3, p. 503-712, 1995.

DYCKHOFF, H. A typology of cutting and packing problems. *European Journal of Operational Research*, vol. 44, p. 145-159, 1990.

DYCKHOFF, H.; FINKE, U. Cutting and packing in production and distribution: Typology and bibliography. Springler-Verlag Co, Heildelberg, 1992.

DYCKHOFF, H.; SCHEITHAUER, G.; TERNO, J. Cutting and packing. In: M. AMICO; F. MAFFIOLI; S. MARTELLO: *Annotated bibliographies in combinatorial optimisation*, John Wiley & Sons, New York, NY, p. 393-414, 1997.

DOWSLAND, K. The three-dimensional pallet chart: an analysis of the factors affecting the set of feasible layouts for a class of two-dimensional packing problems. *Journal of the Operational Research Society*, vol. 35, p. 895-905, 1984

DOWSLAND, K. An exact algorithm for the pallet loading problem. *European Journal of Operational Research*, vol. 31, p. 78-84, 1987.

DOWSLAND, K. Some experiments with simulated annealing techniques for packing problems. *European Journal of Operational Research*, vol. 68, p. 389-399, 1993.

DOWSLAND, K. Simple tabu thresholding and the pallet loading problem. In: I. H. OSMAN, I. H.; KELLY J. P: Metaheuristics*: Theory and Applications*, Kluwer Academic Publishers, p. 379-406, 1996.

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

FARAGO, R.; MORABITO, R. Um método heurístico baseado em relaxação Lagrangiana para resolver o problema de carregamento de paletes do produtor. *Pesquisa Operacional*, vol. 12, n. 2, p. 197-212, 2000.

FEO, T.; RESENDE M. G. C. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, vol. 6, p. 109-133, 1995.

GLOVER, F.; LAGUNA, M. *Tabu Search*, Kluwer Academic Pub., 1997.

HADJICONSTANTINOU, E.; CHRISTOFIDES, N. An exact algorithm for general, orthogonal, two-dimensional knapsack problems. *European Journal of Operational Research*, vol. 83, p. 39-56, 1995.

HERBERT E.; DOWSLAND K. A family of genetic algorithms for the pallet loading problem. *Annals of Operations Research*, vol. *63 – Metaheuristics in Combinatorial Optimization*, 1996.

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

HODGSON, T. A combined approach to the pallet loading problem. *IIE Transactions*, vol. 14, n. 3, p. 176-182, 1982.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. Ann Arbor, University of Michigan Press, 1975.

KIRKPATRICK, S.; GELLAT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science*, vol. 220, p. 671-680, 1983.

LETCHFORD, A.; AMARAL, A. Analysis of upper bounds for the pallet loading problem. *European Journal of Operational Research*, n. 132, p. 582-593, 2001.

LINS, L.; LINS, S.; MORABITO, R. An *L*-approach for packing (*l*, *w*)-rectangles into rectangular and *L*-shaped pieces. Aceito para publicação em *Journal of the Operational Research Society,* 2003.

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

MARTINS, G. H. *Packing in two and three dimensions*. Ph.D. Dissertation. Naval Postgraduate School, Monterey, CA, EUA, 2002.

MORABITO, R.; ARENALES, M. An and/or-graph approach to the container loading problem. *International Transactions in Operational Research*, vol. 1, n. 1, p. 59-73, 1994.

MORABITO, R.; FARAGO, R. A tight lagrangean relaxation bound for the manufacturer's pallet loading problem. *Studia Informatica Universalis*, vol. 2, n. 1, p. 57-76, 2002.

MORABITO, R.; MORALES, S. R. A simple and effective heuristic to solve the manufacturer's pallet loading problem. *Journal of the Operational Research Society*, n. 48, p. 819-828, 1998.

NELISSEN, J. How to use structural constraints to compute an upper bound for the pallet loading problems. *European Journal of Operational Research*, n. 84, p. 662-680, 1995.

PUREZA, V.; MORABITO, R. Uma heurística de busca tabu simples para o problema de carregamento de paletes do produtor. *Pesquisa Operacional*, vol. 23, n. 2, p. 359-378, 2003.

SCHEITHAUER, G.; TERNO, J. The G4 heuristic for the pallet loading problem. *Journal of the Operational Research Society*, n. 47, p. 511-522, 1996.

SMITH, A.; DE CANI, P. An algorithm to optimize the layout of boxes in pallets. *Journal of the Operational Research Society*, n. 31, p. 573-578, 1980.

STEUDEL, H. Generating pallet loading patterns: a special case of the two-dimensional cutting stock problem. *Management Science*, n. 10, p. 997-1004, 1979.

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

TARNOWSKI, A.; TERNO, J.; SCHEITHAUER, G. A polynomial-time algorithm for the guillotine pallet loading problem. *INFOR*, vol. 32, n. 4, p. 275-287, 1994.

TSAI, R.; MALSTROM, E.; KUO, W. Three-dimensional palletization of mixed box sizes. *IEE Transactions*, vol. 25, p. 64-75, 1993.

WANG, P.; WAESCHER, G. Special issue on cutting and packing. *European Journal of Operational Research,* vol. 141, n. 2, p. 239-469, 2002.

Agradecimentos

As autoras agradecem o apoio da FAPESP e as valiosas sugestões dos revisores anônimos.