Algoritmos de Aproximação para Problemas de Empacotamento

Flávio Keidi Miyazawa

Tese apresentada ao
Instituto de Matemática e Estatística da
Universidade de São Paulo para obtenção
do grau de Doutor em Ciências

Área de Concentração: Ciência da Computação
Orientadora: Profa. Dra. Yoshiko Wakabayashi

Durante a elaboração deste trabalho o autor recebeu apoio financeiro do CNPq

—São Paulo, novembro de 1997—

Algoritmos de Aproximação para Problemas de Empacotamento

Flávio Keidi Miyazawa

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida por Flávio Keidi Miyazawa e aprovada pela comissão julgadora.

Banca examinadora:

- Profa. Dra. Yoshiko Wakabayashi (Orientadora) - IME-USP
- Prof. Dr. Carlos E. Ferreira - IME-USP
- Prof. Dr. Cláudio L. Lucchesi - IC-UNICAMP
- Prof. Dr. Nelson Maculan - UFRJ
- Prof. Dr. Reinaldo Morábito - UFSCar

—São Paulo, novembro de 1997—

à minha irmã Miriam
e aos meus pais Marina e Takuo

Agradecimentos

Durante o período de elaboração desta tese recebi a colaboração de várias pessoas. Gostaria de agradecer a estas pessoas que tanto ajudaram para que esta tese chegasse à sua versão final.

Agradecimento especial merece a Profa. *Yoshiko Wakabayashi*, pela sua excelente orientação, dedicação, paciência e por ser além de tudo, uma grande amiga.

Agradeço aos demais membros da banca examinadora pela leitura, sugestões e correções: *Carlos E. Ferreira, Cláudio Lucchesi, João Meidanis, José Coelho, Nelson Maculan* e *Reinaldo Morábito*.

Ao *Carlinhos* agradeço também pelas leituras, comentários e sugestões das versões preliminares, que melhoraram a apresentação e organização desta tese.

Ao *Yoshi* agradeço pela sugestão da prova do Fato 4.4.15.

Gostaria de agradecer também àquelas pessoas que durante o período de pós-graduação no IME-USP tanto fizeram para que este período fosse tão agradável.

À *Cristina Fernandes, José Augusto, José Coelho* e *Siang W. Song*, por boas discussões de problemas acadêmicos e por toda ajuda na minha formação acadêmica.

Ao *Li Kuan, Marcelo Carvalho, Mario Harada* e *Ricardo Ueda* por seus conselhos, incentivo e amizade.

Ao *Cesar Bravo* pelos bons momentos de descontração e divertimento que passamos no CRUSP.

Aos amigos de mestrado e doutorado, *Alfredo, Ayumi, Cassio, Claus, Eliany, Eloi, Estela, Fabiana Soares, Fabio Kon, Fabio Martinez, Fran, Glauber, Henrique, Jair, Jefferson, Loparic, Lucia Ikemoto, Luis, Marcelo Coniglio, Marco Aurélio, Orlando Lee* e *Scalzitti* que me acompanharam durante a elaboração desta tese e fizeram os anos de pós-graduação no IME-USP um período muito agradável.

Ao *Arnaldo Mandel* e sua equipe (*Adriano N. Rodrigues, Douglas S. Vieira* e *Felix Almeida*) que ajudaram a resolver vários problemas computacionais e têm a dura tarefa de manter funcionando a rede de computadores do IME-USP (onde esta tese foi redigida).

À *Aurea Matsumura* pelo amor, carinho e todos os momentos felizes que passamos juntos.

Por fim gostaria de agradecer aos Meus Pais, *Marina* e *Takuo*, que sempre me deram muito apoio e incentivo para que eu conseguisse alcançar meus objetivos.

Algoritmos de Aproximação para Problemas de Empacotamento

Flávio Keidi Miyazawa

Resumo da tese apresentada ao IME-USP como parte dos
requisitos necessários para a obtenção do título de Doutor em Ciências.

Resumo

Problemas de empacotamento consistem em colocar, de uma forma econômica, uma coleção de objetos dentro de recipientes. Esses problemas podem diferir em duas linhas conforme o objetivo do problema de otimização em questão. Em uma destas linhas, o objetivo é minimizar o número de recipientes usados para empacotar os objetos. Em outra linha, os objetos devem ser empacotados em apenas um recipiente, sendo que este recipiente tem apenas uma dimensão ilimitada e todas as outras são limitadas. Neste caso, o objetivo é minimizar o 'tamanho' do empacotamento com relação à dimensão ilimitada do recipiente.

Nesta tese investigamos as versões em que os objetos e os recipientes são bi- ou tridimensionais, e de 'formas ortogonais'. Assim, os objetos são retângulos ou caixas retangulares e os recipientes são faixas, placas, caixas de altura ilimitada ou caixas de dimensões finitas (contêineres). Além disso, todos os empacotamentos devem ser ortogonais. Abordamos os seguintes problemas: o *problema de empacotamento em faixa*, o *problema de empacotamento em placas*, o *problema de empacotamento tridimensional* e o *problema de empacotamento em contêineres*. Esses problemas são $\mathcal{NP}$-difíceis, não aproximáveis — em termos absolutos— além de certas constantes. Apresentamos algoritmos de aproximação para esses problemas e estudamos o seu desempenho assintótico. Descrevemos algoritmos *on-line* e *off-line* tanto para o caso orientado como para o caso em que rotações ortogonais são permitidas.

Para o problema de empacotamento em faixa, apresentamos um algoritmo com limite de desempenho assintótico não maior que 1,62 e outro, *on-line*, cujo limite de desempenho assintótico é 1,75. Ambos para o caso onde rotações são permitidas. Para o problema de empacotamento em placas, apresentamos um algoritmo com limite de desempenho assintótico não maior que 2,64 e outro, *on-line*, com limite de desempenho assintótico não maior que 3,25. Ambos também para o caso onde rotações ortogonais são permitidas. Para o problema de empacotamento tridimensional para o caso orientado, apresentamos um algoritmo com limite de desempenho assintótico não maior que 2,67. Para o caso onde rotações em torno do eixo da altura são permitidas, apresentamos um algoritmo com um limite de desempenho assintótico 2,67 e outro, *on-line*, com um limite de desempenho assintótico 3,25. Para o problema de empacotamento em contêineres onde rotações são permitidas, apresentamos um algoritmo com um limite de desempenho assintótico 4,89 e para o caso *on-line*, um algoritmo com limite de desempenho assintótico não maior que 6,25.

Os limites acima ou são novos ou são melhores que os encontrados na literatura. Além disso, apresentamos resultados que relacionam a complexidade dos problemas da versão orientada com a versão em que rotações ortogonais são permitidas. Também apresentamos vários algoritmos de aproximação para casos particulares destes problemas: empacotamento de quadrados, de cubos e de objetos 'pequenos'. Para esses casos, os algoritmos que obtivemos têm limites de desempenho melhores que os acima mencionados.

Palavras-chave: problemas de empacotamento, algoritmos de aproximação, limite de desempenho assintótico.

Orientadora de tese: Yoshiko Wakabayashi.

Approximation Algorithms for Packing Problems

Flávio Keidi Miyazawa

Abstract of the thesis presented to IME-USP as partial
fulfillment for the requirements for the degree of Doctor of Science.

Abstract

Packing problems consist in placing a collection of objects inside recipients in an economical way. These problems can be of two types depending on the objective under consideration. In the first case, the objective is to minimize the number of recipients used to pack the objects. In the second case, the objects must be packed into only one recipient—whose dimensions are all bounded, except one—and the objective is to minimize the 'size' of the packing, measured with respect to the unbounded dimension of the recipient.

In this thesis we consider packing problems where the objects and recipients are 2- or 3-dimensional, and of 'orthogonal shape'. Thus, the objects are rectangles or rectangular boxes and the recipients are rectangular strips, rectangular bins, rectangular boxes of unbounded height or rectangular boxes of bounded dimensions (containers). Furthermore, all packings must be orthogonal. We consider the following problems: the *strip packing problem*, the *two-dimensional bin packing problem*, the *three dimensional packing problem* and the *container packing problem*. These problems are $\mathcal{NP}$-hard, not approximable—in the absolute sense—within certain constants. We present (polynomial) approximation algorithms for these problems and study their asymptotic performance. We describe on-line and off-line algorithms for the oriented case and for the case where orthogonal rotations are allowed.

For the strip packing problem, we present an algorithm with asymptotic performance bound at most 1.62 and an on-line algorithm with asymptotic performance bound 1.75. Both algorithms are for the case where orthogonal rotations are allowed. For the two-dimensional bin packing problem allowing orthogonal rotations, we present an algorithm with asymptotic performance bound at most 2.64 and an on-line algorithm with an asymptotic performance bound of 3.25. For the three-dimensional packing problem, we present an algorithm for the oriented case, with asymptotic performance bound at most 2.67. For the case where orthogonal rotations around the height axis are allowed, we present an algorithm whose asymptotic performance bound is at most 2.67 and another on-line algorithm with an asymptotic performance bound of 3.25. For the container packing problem where orthogonal rotations are allowed, we present an algorithm with asymptotic performance bound at most 4.89 and an on-line algorithm with asymptotic performance bound at most 6.25.

All the bounds above are either new or are improvements of the existing ones in the literature. Moreover, we present results relating the complexity of these problems in the oriented version with the version where orthogonal rotations are allowed. We also present several approximation algorithms for special cases of these problems: packing of squares, of cubes and of 'small' objects. For these cases, we describe algorithms whose asymptotic performance bounds are better than those mentioned above.

Keywords: packing problems, approximation algorithms, asymptotic performance bound.

Thesis supervisor: Yoshiko Wakabayashi.

Índice

Lista de Figuras

Lista de Tabelas

Introdução

Os problemas abordados nesta tese, como muitos de natureza combinatória, podem ser facilmente formulados e compreendidos, escondendo atrás de sua aparente simplicidade, a sua real complexidade. São problemas $\mathcal{NP}$-difíceis não aproximáveis —em termos absolutos— além de certas constantes. Esse caráter complexo em termos de aproximabilidade absoluta, justifica o estudo desses problemas quanto à sua aproximabilidade em termos assintóticos. Este é o tema central desta tese.

Consideramos aqui diversos problemas de empacotamento bi- e tridimensional, muitos deles já investigados sob essa abordagem, e descrevemos para esses problemas algoritmos de aproximação cujos limites de desempenho assintótico são melhores do que os encontrados na literatura.

Antes de descrevermos formalmente os problemas e mencionarmos a nossa contribuição, vamos introduzir o leitor a cada um dos problemas abordados nesta tese, ilustrando a ocorrência de vários deles no processo de construção de um prédio.

Primeiro, considere a estrutura de concreto armado de um prédio. Mais precisamente, considere as colunas de concreto contendo barras de aço em seu interior. Estas colunas podem ter diferentes comprimentos e conseqüentemente cada uma delas requer barras com certo comprimento. Em geral, a fábrica que produz estas barras somente as produz com um comprimento específico, digamos 12 metros. Assim, a construtora deve comprar estas barras grandes e cortá-las conforme sua necessidade. Claramente, para minimizar os custos, o seu objetivo é comprar o menor número de barras necessário na construção. Este é um problema muito comum que também aparece em problemas de corte de vigas, estruturas metálicas, canos, esquadrias, etc. Trata-se de um *problema de empacotamento unidimensional*.

Prosseguindo na construção do prédio, considere agora as janelas e divisórias de vidro retangulares, necessárias em quantidade e tamanhos diferentes. A construtora faz o pedido de compra a uma vidraçaria, especificando as dimensões e quantidades de cada item. A vidraçaria deve cortar estes pedaços a partir de placas de vidro de tamanhos fixos, digamos 3 metros por 3

metros, que são fornecidas pelo fabricante de vidro. Novamente, para minimizar os custos, o objetivo da vidraçaria é usar o menor número destas placas 3×3. Este problema também aparece no corte de chapas, divisórias, compensados, etc. Temos aqui um *problema de empacotamento (bidimensional) em placas.*

Considere agora o seguinte problema. A construtora, tendo terminada a construção do prédio, contrata uma empresa de publicidade para fazer bandeiras, faixas, bonés e camisetas com a propaganda do prédio. Para fazer todo este material publicitário, a empresa precisa comprar uma certa quantidade de tecido. As tecelagens produzem tecidos com uma largura fixa e com comprimento variável, sendo que seu custo é proporcional ao comprimento. Assim, para minimizar os seus gastos o objetivo da empresa de publicidade é comprar o menor comprimento de tecido necessário para cortar todos os itens requisitados. Este é um *problema de empacotamento (bidimensional) em faixa.*

Um outro problema que ocorre está relacionado ao transporte de produtos e cargas para o local de construção do prédio. Vários produtos (como vidros, pisos, azulejos) devem ser encaixotados e então transportados em furgões de dimensões fixas. Para minimizar os custos o objetivo aqui é fazer o menor número de viagens da loja de materiais para o local de construção. O problema consiste em empacotar as caixas no menor número de furgões e conseqüentemente, minimizar o número de viagens. Trata-se aqui de um *problema de empacotamento em contêineres.*

Uma vez que os produtos são transportados para o local da construção, estes devem ser armazenados em um galpão (de forma retangular), de forma a ficarem protegidos até seu uso. Surge aqui o problema de empacotar estas caixas dentro do galpão. O objetivo é encontrar um empacotamento cuja altura seja a menor possível.

Os problemas envolvidos nos exemplos acima são chamados de *problemas de corte e/ou problemas de empacotamento.* Nós os chamamos de *problemas de empacotamento.* Segundo a definição de *empacotamento* que apresentamos, os problemas de corte de barras, vidros, tecidos ou outros materiais podem ser vistos como problemas de empacotamento (de pedaços de barras em barras, de moldes em vidros ou tecidos).

De uma forma geral, os problemas de empacotamento que abordamos nesta tese consistem em empacotar uma lista de itens em recipientes. Dependendo da natureza dos itens sendo empacotados dizemos que os problemas são uni-, bi- ou tridimensionais. Não consideramos o caso em que os itens e recipientes têm forma arbitrária, mas apenas ângulos retos (ou seja, retângulos, paralelepípedos, etc). Os termos que usamos para os recipientes são *barras*, *faixas*, *placas*, *caixas* e *contêineres.* Os problemas abordados podem diferir em duas linhas conforme a função objetivo do problema de otimização em questão. Em uma destas linhas, o objetivo é minimizar o número de recipientes usados para empacotar os itens da lista. Em outra linha, os itens devem ser empacotados em apenas um recipiente, sendo que este recipiente tem apenas

uma dimensão ilimitada e todas as outras são limitadas (por exemplo, uma caixa com fundo limitado e altura ilimitada). Neste caso, o objetivo é minimizar o 'tamanho' do empacotamento com relação à dimensão ilimitada do recipiente. Nos referimos a este tamanho como sendo a altura do empacotamento.

Listamos a seguir os problemas de empacotamento que serão objeto de nosso estudo, colocando entre parênteses os respectivos exemplos mencionados anteriormente na construção de um prédio. Problema de Empacotamento Unidimensional (corte de barras de aço), Problema de Empacotamento em Faixa (corte de tecido), Problema de Empacotamento em Placas (corte de placas de vidro), Problema de Empacotamento Tridimensional (armazenamento em galpões) e Problema de Empacotamento em Contêineres (empacotamento em furgões).

Diferentes abordagens têm sido usadas no desenvolvimento de algoritmos para problemas de empacotamento. Uma destas abordagens é a *abordagem exata*. Nesta abordagem é feita a busca de uma solução ótima do problema. É em geral uma abordagem usada para instâncias particulares ou mesmo para instâncias pequenas, já que o esforço computacional exigido por essas abordagens torna inviável o tratamento de instâncias de médio porte. Devido a esta intratabilidade, uma outra abordagem consiste em desenvolver algoritmos polinomiais, não necessariamente buscando soluções ótimas.

Ao considerar o desenvolvimento de algoritmos polinomiais para estes problemas a principal preocupação está na qualidade da solução produzida pelo algoritmo, isto é, no *desempenho do algoritmo*. Surge então uma nova dificuldade: como comparar estes algoritmos. Uma maneira de fazer isso é executar vários algoritmos sobre diversas instâncias, vindas de casos práticos ou mesmo teóricos (*benchmark*), e comparar os resultados gerados por estes algoritmos: a *abordagem experimental*. É uma abordagem que pode ser boa quando fazemos o estudo de casos (particulares) práticos [19, 58]. Por outro lado, a sua principal desvantagem é que fica um tanto restrita aos exemplos testados. Além do mais, melhorias obtidas sobre um conjunto de instâncias não são garantias de que conduzem a melhorias sobre os casos gerais. Essa abordagem pode levar a algoritmos que geram resultados muito aquém do ótimo, sem nenhuma garantia quanto ao seu desempenho. A questão que surge é como fazer uma melhor análise dos desempenhos dos algoritmos.

Observamos que muitas heurísticas com bons desempenhos empíricos tiveram seu comportamento analisado mais formalmente e, não surpreendentemente, também apresentaram bons desempenhos sob outros tipos de análise. Como abordagens mais formais podemos citar a *abordagem de algoritmos de aproximação* e a *abordagem probabilística*.

Na *abordagem de algoritmos de aproximação*, busca-se desenvolver algoritmos para os quais são garantidos (para qualquer instância) um limitante superior para a razão entre o valor do empacotamento gerado pelo algoritmo e o valor de um empacotamento ótimo. Comparando

algoritmos desta forma, podemos dizer qual algoritmo tem um limitante melhor para esta razão. Como esse limite é válido para qualquer instância do problema, podem existir algoritmos com limites superiores não tão pequenos, mas que na prática apresentam desempenho melhor.

A *abordagem probabilística* consiste em se tomar uma distribuição mais ou menos realista dos dados de entrada e calcular a esperança do resultado gerado pelo algoritmo. Uma dificuldade desta abordagem está em achar esta distribuição realista, muitas vezes dependente das aplicações de cada problema. É em geral uma abordagem difícil, especialmente se considerada na análise de algoritmos mais eficientes e complicados (*cf.* Coffman e Shor [10]).

Optamos pela abordagem de algoritmos de aproximação, a qual referiremos apenas como *Algoritmos de Aproximação* (embora este termo seja também usado para a abordagem probabilística). Esta abordagem tem sido estudada antes mesmo da publicação dos artigos de Cook [11] e de Karp [36] que, no início dos anos 70 introduziram a teoria de complexidade computacional, dando um tratamento formal a questões referentes à eficiência de algoritmos, e classificação de problemas quanto à existência ou (provável) inexistência de algoritmos eficientes para se resolvê-los. O primeiro algoritmo de aproximação para um problema $\mathcal{NP}$-difícil foi desenvolvido por Graham [26], para um problema de escalonamento de tarefas em computadores, conhecido como *Multiprocessor Scheduling*. Este problema pode ser formulado da seguinte maneira.

São dados uma lista de n tarefas $T_1, \ldots, T_n$, e m máquinas idênticas, $M_1, \ldots, M_m$. Cada tarefa T_j deve ser processada sem interrupção, por um tempo t_j, por uma das m máquinas, cada uma das quais pode processar no máximo uma tarefa por vez. O objetivo consiste em achar uma atribuição das tarefas às máquinas, de modo a minimizar o tempo total para processar as n tarefas.

Os primeiros algoritmos de aproximação para problemas de empacotamento foram desenvolvidos no início da década de 70. Na tese de doutorado de David S. Johnson [30], de 1973, são apresentados vários algoritmos para o caso unidimensional. Antes disso, em 1972 o problema de empacotamento unidimensional foi estudado como um problema de alocação de memória por Garey, Graham e Ullman [23], que apresentaram as primeiras análises não triviais para os algoritmos de aproximação desenvolvidos.

Cabe aqui ressaltar que o desenvolvimento de algoritmos de aproximação para problemas de empacotamento está intimamente relacionado com a origem da teoria de algoritmos de aproximação. Notamos que até mesmo o problema investigado por Graham, enunciado acima, tem sua versão de decisão igual à versão de decisão do problema de empacotamento unidimensional.

Em um artigo de 1974, Johnson [31] introduziu o termo *approximation algorithm* usando-o para o problema de empacotamento unidimensional, o problema da mochila, o problema da satisfatibilidade máxima, o problema da cobertura de conjuntos, o problema da coloração de vértices e o problema do clique máximo (*cf.* Johnson [32]). Neste artigo, Johnson sugeriu que

a abordagem de algoritmos de aproximação anteriormente usada para o problema de empacotamento unidimensional fosse aplicada aos demais problemas citados. Vale lembrar também que os primeiros estudos sobre limitantes inferiores para os limites de desempenho assintóticos de algoritmos *on-line* foram feitos para problemas de empacotamento. Mais ainda, os primeiros algoritmos com esquema de aproximação assintótico para um problema fortemente $\mathcal{NP}$-completo *($\mathcal{NP}$-complete in strong sense)* foram desenvolvidos para o problema de empacotamento unidimensional. Vega e Luecker [16] apresentaram em 1981 um esquema PAAS (*Polynomial Time Asymptotic Approximation Scheme*) e Karmarkar e Karp apresentaram, em 1982, um esquema FPAAS (*Fully Polynomial Time Asymptotic Approximation Scheme*). Estes dois resultados causaram um choque na teoria de algoritmos de aproximação, pois muitos achavam que não existissem esquemas de aproximação assintóticos PAAS/FPAAS para problemas fortemente $\mathcal{NP}$-completos (*cf.* Motwani [51]).

Até a década de 80 as publicações sobre algoritmos de aproximação para problemas de empacotamento se concentraram nos casos unidimensional e bidimensional [30, 31, 34, 6, 1, 2]. Uma resenha sobre esses dois casos pode ser encontrada em [7]. Um texto recentemente publicado [28], que trata só de algoritmos de aproximação, traz um capítulo dedicado ao caso unidimensional [8], constituindo-se em uma resenha bem atualizada.

O objetivo principal desta tese é o de desenvolver algoritmos de aproximação para problemas de empacotamento e investigar aspectos relativos a este tipo de enfoque. Dedicamos cada capítulo a um problema distinto e analisamos o desempenho dos algoritmos propostos.

Muitos dos algoritmos que desenvolvemos usam (como subrotinas) outros algoritmos já mencionados na literatura. Alguns desses últimos são apresentados aqui, visando facilitar o entendimento dos nossos algoritmos ou visando tornar este trabalho mais auto-contido. Ressaltamos que, para o entendimento da análise do desempenho dos algoritmos que desenvolvemos, é suficiente que o leitor saiba (ou assuma como verdadeiro) o fato de que um certo algoritmo para um certo subproblema tem determinada garantia de desempenho. Assim, apenas apresentamos os principais resultados relacionados a esses algoritmos, mas não provamos esses resultados, já que podem ser encontrados na literatura.

No **Primeiro Capítulo**, *Preliminares*, estabelecemos a notação e definimos informalmente todos os problemas em questão (Problema de Empacotamento Unidimensional, Problema de Empacotamento em Faixa, Problema de Empacotamento em Placas, Problema de Empacotamento Tridimensional e Problema de Empacotamento em Contêineres). Mencionamos também as medidas de desempenho dos algoritmos. Além disso, discutimos e apresentamos resultados referentes às versões destes problemas quando rotações ortogonais podem ser feitas.

No **Segundo Capítulo** abordamos o *Problema de Empacotamento Unidimensional*. Apresentamos alguns algoritmos usados como subrotinas de outros, bem como alguns outros consider-

ados importantes na literatura. Estes algoritmos são importantes para que o leitor se familiarize com o modo como certos empacotamentos são construídos. Ao fim do capítulo, apresentamos brevemente algumas das técnicas de empacotamento (algumas desenvolvidas nesta tese para problemas bi- e tridimensionais) aplicadas ao problema de empacotamento unidimensional.

O **Terceiro Capítulo** é sobre o *Problema de Empacotamento em Faixa*. Neste capítulo apresentamos um algoritmo com limite de desempenho assintótico 1,613 e outro, *on-line*, com limite de desempenho assintótico 1,75, para o caso onde rotações ortogonais podem ser feitas. Além disso, introduzimos um resultado relacionado a um algoritmo usado como subrotina para o problema do empacotamento tridimensional.

No **Quarto Capítulo** estudamos o *Problema de Empacotamento em Placas*. Apresentamos dois algoritmos conhecidos para o caso orientado deste problema, e dois novos algoritmos — para a versão onde rotações são permitidas— um com limite de desempenho não maior que 2,64 e outro, *on-line*, com limite de desempenho assintótico 3,25. Também apresentamos outros algoritmos para casos particulares envolvendo restrições sobre o tamanho dos itens da instância, e restringindo o empacotamento a placas quadradas.

O *Problema de Empacotamento Tridimensional* é estudado no **Quinto Capítulo**. Abordamos duas versões para este problema. Uma onde todos os itens são orientados, e outra onde rotações em torno de um dos eixos é permitida. Apresentamos algoritmos para estas duas versões com limite de desempenho assintótico não maior que 2,67, e alguns algoritmos para casos especiais quando restrições quanto à forma do fundo das caixas são inseridas. Para o problema *on-line*, apresentamos um algoritmo com limite de desempenho assintótico que pode se tornar tão próximo de 3,25 quanto se queira.

O último problema abordado nesta tese, o *Problema de Empacotamento em Contêineres*, é investigado no **Sexto Capítulo**. Apresentamos algoritmos para o caso geral onde qualquer tipo de rotação ortogonal é permitida. Um dos algoritmos, *off-line*, com limite de desempenho assintótico não maior que 4,89 e outro, *on-line*, com limite de desempenho assintótico 6,25. Além disso, apresentamos algoritmos para casos particulares quando há restrições quanto ao tamanho dos itens a serem empacotados, e também quando todos os itens e contêineres são cubos.

No **Sétimo Capítulo** apresentamos algumas *Considerações Finais*. Comentamos brevemente nossos resultados, comentamos aspectos relativos à implementação e apresentamos algumas questões em aberto sobre os problemas de empacotamento.

Apresentamos também dois apêndices. No **Apêndice A**, apresentamos algumas *Tabelas* mostrando alguns dos melhores limites de desempenho assintóticos de algoritmos para os problemas estudados nesta tese. No **Apêndice B**, apresentamos várias *Aplicações* para os problemas de empacotamento discutidos nesta tese. Justamente por abordar uma variante pouco estudada para problemas de empacotamento —a possibilidade de rotações ortogonais— sentimos

necessidade de frisar esta importância ressaltando as aplicações onde rotações ortogonais são permitidas.

A organização dos capítulos de cada problema é feita do seguinte modo. Primeiramente apresentamos uma breve *introdução* onde descrevemos alguns resultados de aproximação conhecidos. Apresentamos algumas *definições*. Em seguida dividimos o capítulo em duas seções: abordamos primeiro o *caso orientado*, e depois o *caso com rotações*. Esta divisão só não é feita para o capítulo do problema de empacotamento unidimensional (onde rotações não fazem sentido); e neste caso, dividimos o capítulo em seções sobre *algoritmos on-line*, *algoritmos off-line* e *algumas técnicas*. Ao fim de cada um dos capítulos apresentamos uma tabela que inclui todos os algoritmos citados para o problema.

Quando o algoritmo apresentado em um capítulo é para o caso geral damos o mesmo nome do algoritmo para a seção correspondente. Para as seções onde apresentamos casos particulares do problema, o nome da seção correspondente descreve o caso particular considerado (e não o nome do algoritmo).

Capítulo 1

Preliminares

1.1 Introdução

Neste capítulo, estabelecemos a notação e definimos informalmente todos os problemas que são tratados nesta tese. Mencionamos a complexidade computacional destes problemas, bem como as medidas padrões usadas para especificar o desempenho dos algoritmos de aproximação.

Além disso, discutimos e apresentamos resultados referentes às versões desses problemas, nas quais rotações ortogonais podem ser feitas. Mostramos que essas versões, no caso geral, são tão difíceis quanto o caso com orientação fixa.

1.2 Problemas de Empacotamento

Nesta seção, descrevemos informalmente todos os problemas que serão objeto de nosso estudo. Lembramos que esses mesmos problemas serão formalmente definidos nos respectivos capítulos. A notação bem como a definição de alguns termos aqui utilizados são apresentados na seção seguinte.

O *Problema de Empacotamento Unidimensional* (PEU), também conhecido como *bin packing problem*, é o seguinte. Dados uma lista de barras $L = (e_1, \ldots, e_n)$, cada qual com comprimento $x(e_i)$ (se estivermos trabalhando sobre o eixo x); e barras B (recipientes) de comprimento C, o problema consiste em empacotar as barras de L em barras B de forma a usar o menor número possível de barras B.

O *Problema de Empacotamento em Faixa*, conhecido como *strip packing problem*, consiste em: dados uma lista de retângulos $L = (r_1, \ldots, r_n)$, onde $r_i = (x_i, y_i)$, e um retângulo $R = (a, \infty)$, empacotar os retângulos de L em R de forma que nenhum retângulo se sobreponha a outro e cada retângulo de L não esteja fora dos limites de R, minimizando o tamanho do empacotamento

na direção ilimitada do recipiente R. Mais ainda, o empacotamento deve ser *ortogonal*, *i.e.*, as arestas dos retângulos de L devem ser paralelas ou ortogonais às arestas de R. Consideramos uma versão com orientação fixa e outra onde rotações ortogonais são permitidas. Nesta última versão, cada retângulo r_i pode ser empacotado tratando-o como sendo da forma $r_i = (x_i, y_i)$ ou da forma (y_i, x_i). A versão com orientação fixa será denotada como PEF; e a versão que permite rotação será denotada por PEF^r.

O *Problema de Empacotamento em Placas*, também conhecido como *two dimensional bin packing problem*, consiste em: dados uma lista de retângulos $L = (r_1, \ldots, r_n)$, onde $r_i = (x_i, y_i)$, e placas da forma $R = (a, b)$, empacotar os retângulos de L em placas R de forma a usar o menor número destas placas. O empacotamento deve ser de modo que nenhum retângulo se sobreponha a outro e nem esteja fora dos limites da placa onde este foi empacotado. Mais ainda, o empacotamento deve ser ortogonal. A versão totalmente orientada será referenciada como PEP, e a versão permitindo rotação, como PEP^r.

O *Problema de Empacotamento Tridimensional*, também conhecido como *tridimensional packing problem*, consiste em: dados uma lista de caixas $L = (b_1, \ldots, b_n)$, onde $b_i = (x_i, y_i, z_i)$, e uma caixa $B = (a, b, \infty)$, empacotar as caixas de L em B de forma ortogonal e minimizando o tamanho do empacotamento na direção ilimitada do recipiente. Consideramos uma versão totalmente orientada (PET) e outra orientada no eixo z (PET^r). Na versão totalmente orientada, uma caixa só pode ser empacotada na orientação dada inicialmente. Na versão orientada no eixo z, é permitido fazer rotações em torno do eixo z. Isto é, uma caixa $b_i = (x_i, y_i, z_i)$ pode ser empacotada tratando-a como sendo da forma (x_i, y_i, z_i) ou da forma (y_i, x_i, z_i).

O *Problema de Empacotamento em Contêineres*, conhecido como *box packing problem*, consiste em: dados uma lista de caixas $L = (b_1, \ldots, b_n)$, onde $b_i = (x_i, y_i, z_i)$, e caixas $B = (a, b, c)$, o problema consiste em empacotar as caixas de L dentro de caixas B, usando o menor número destas caixas. Novamente, este empacotamento deve ser ortogonal. Consideramos a versão orientada (PEC), e a versão permitindo rotações (PEC^r), em que cada caixa $b_i = (x_i, y_i, z_i)$ pode ser empacotada tratando-a como sendo de uma das formas seguintes: (x_i, y_i, z_i), (x_i, z_i, y_i), (y_i, x_i, z_i), (y_i, z_i, x_i), (z_i, x_i, y_i) e (z_i, y_i, x_i).

Nesta tese, ao nos referirmos a *problemas de empacotamento* estamos nos referindo a qualquer um dos problemas acima. Mais ainda, diversas vezes nos referimos a um *empacotamento* de uma lista de itens (para um dos problemas acima) em determinada dimensão, sem especificar se rotações são permitidas ou não, deixando que o contexto defina a qual dos problemas este empacotamento se refere.

1.3 Notação e Definições

Nosso espaço de trabalho é o espaço euclideano $\mathbb{R}^3$, com o sistema de coordenadas xyz. Para o problema de empacotamento unidimensional, este espaço se reduz a uma das dimensões x ou y ou z. Para o caso bidimensional, usamos os planos xy ou yz ou zx; e finalmente, consideramos todo o espaço xyz para os problemas de empacotamento tridimensional.

Um item e a ser empacotado tem suas dimensões definidas por $x(e)$, $y(e)$ e $z(e)$, onde cada uma dessas dimensões é a medida no correspondente eixo do sistema xyz. Para os casos uni e bidimensional alguns destes valores não estarão definidos. Quando o item e é unidimensional, este é referido como uma *barra*; quando é bidimensional, é chamado de *retângulo* ou *quadrado*; e quando é tridimensional é chamado de *caixa*.

No caso dos recipientes, suas dimensões são especificadas analogamente; sendo que quando uma delas é ilimitada, indicamos isso por ∞, como por exemplo em $B = (a, b, \infty)$. Quando o recipiente é unidimensional, este é referido como uma *barra*; quando é bidimensional é referido como *placa*, se é da forma (a, b); e como *faixa*, se é da forma (a, ∞). Recipientes tridimensionais da forma (a, b, c) são chamados *contêineres* e os da forma (a, b, ∞) são chamados *caixas* (de altura ilimitada).

Quando abordamos o problema de empacotamento unidimensional no espaço x, o *comprimento* de um objeto da instância (item ou recipiente) é a sua dimensão no eixo x. Quando abordamos o problema de empacotamento em faixa, no plano xy, dizemos que um objeto da instância tem *largura* definida pela dimensão do objeto no eixo x e *altura* definida pela dimensão do objeto no eixo y; no exemplo apresentado na introdução desta tese (corte de tecido), apesar de usarmos comprimento em vez de altura, preferimos apresentar este problema como sendo de minimização de altura dado que esta é a forma comumente encontrada na literatura. No caso do problema de empacotamento em placas, associamos o *comprimento* à dimensão do objeto no eixo x e a *largura* à dimensão do objeto no eixo y. Para o problema de empacotamento tridimensional e em contêineres, associamos o *comprimento* de um objeto da instância à sua dimensão no eixo x, sua *largura* à sua dimensão no eixo y e sua *altura* à sua dimensão no eixo z.

Quando consideramos o problema de empacotamento em faixa ou o problema de empacotamento tridimensional, chamamos de *altura do empacotamento* a medida do empacotamento na direção ilimitada do recipiente. Tal medida também é chamada de *tamanho*.

Muitas vezes um item com duas ou três dimensões, pode ser visto como um item de dimensões menores. Assim, uma caixa $b_i = (x_i, y_i, z_i)$, no sistema de coordenadas xyz, pode ser visto como um retângulo $r'_i = (x_i, y_i)$ no plano bidimensional xy, ou mesmo como um item de comprimento $p'_i = (x_i)$ em relação ao eixo x.

Quando nos referimos a um retângulo $r_i = (x_i, y_i)$, denotamos por $S(r_i)$ a área deste

retângulo. Quando nos referimos a uma caixa, digamos $b_i = (x_i, y_i, z_i)$, denotamos por $S(b_i)$ a área de fundo da caixa b_i, sendo que o *fundo* depende de como esta caixa está sendo vista, *i.e.*, qual é o eixo que está sendo considerado como correspondente à altura. Quando o eixo correspondente à altura não estiver muito óbvio, usamos índices sobrescritos para identificar qual o plano de fundo. Assim, $S^{xy}(b_i)$ refere-se a $x(b_i)y(b_i)$.

O *volume* de uma caixa $b_i = (x_i, y_i, z_i)$, é denotado por $V(b_i) := x_i \cdot y_i \cdot z_i$.

Dada uma função $f : C \to \mathbb{R}$ e um subconjunto $C' \subseteq C$, denotamos por $f(C')$ a soma $\sum_{e \in C'} f(e)$.

Embora uma lista com n elementos seja dada como uma n-upla, quando a ordem dos itens é irrelevante, a lista correspondente pode ser vista como um conjunto. Se L é uma lista de caixas, então a soma dos volumes das caixas de L é denotada por $V(L)$. Denotamos por $\mathit{first}(L)$ o primeiro elemento da lista L, se $L \neq \emptyset$; caso $L = \emptyset$ então definimos $\mathit{first}(L)$ como sendo $\emptyset$. Dadas listas $L_1, L_2, \ldots, L_k$, onde $L_i = (e_i^1, e_i^2, \ldots, e_i^{n_i})$, definimos a *concatenação* dessas listas, denotada por $L_1 \| L_2 \| \ldots \| L_k$, como sendo a lista $(e_1^1, \ldots, e_1^{n_1}, e_2^1, \ldots, e_2^{n_2}, \ldots, e_k^1, \ldots, e_k^{n_k})$.

A seguir, apresentamos uma notação conveniente para especificar tipos especiais de itens. Aqui, estamos considerando que as dimensões máximas de cada um dos itens da lista de entrada L, em relação aos eixos x, y e z, é a, b e c, respectivamente.

$$\begin{aligned}
\mathcal{X}[p,q] &:= \{e : \quad p \cdot a < x(e) \leq q \cdot a\}, \\
\mathcal{Y}[p,q] &:= \{e : \quad p \cdot b < y(e) \leq q \cdot b\}, \\
\mathcal{Z}[p,q] &:= \{e : \quad p \cdot c < z(e) \leq q \cdot c\},
\end{aligned}$$

$$\begin{aligned}
\mathcal{Q}^{xy}[p,q] &:= \{e : \quad p \cdot a < x(e) = y(e) \leq q \cdot a\}, \\
\mathcal{Q}^{yz}[p,q] &:= \{e : \quad p \cdot a < y(e) = z(e) \leq q \cdot a\},
\end{aligned}$$

$$\begin{aligned}
\mathcal{C}^{xy}[p_1, q_1 \ ; \ p_2, q_2] &:= \mathcal{X}[p_1, q_1] \cap \mathcal{Y}[p_2, q_2], \\
\mathcal{C}^{yz}[p_1, q_1 \ ; \ p_2, q_2] &:= \mathcal{Y}[p_1, q_1] \cap \mathcal{Z}[p_2, q_2], \\
\mathcal{C}^{zx}[p_1, q_1 \ ; \ p_2, q_2] &:= \mathcal{Z}[p_1, q_1] \cap \mathcal{X}[p_2, q_2], \\
\mathcal{C}^{xyz}[p_1, q_1 \ ; \ p_2, q_2 \ ; \ p_3, q_3] &:= \mathcal{X}[p_1, q_1] \cap \mathcal{Y}[p_2, q_2] \cap \mathcal{Z}[p_3, q_3],
\end{aligned}$$

$$\mathcal{C}_m := \mathcal{C}^{xy}\left[0, \frac{1}{m} \ ; \ 0, \frac{1}{m}\right], \quad \mathcal{Q}_m := \mathcal{Q}^{xy}\left[0, \frac{1}{m}\right],$$

$$\wp_1 := \mathcal{C}^{xy}\left[0, \frac{1}{2} \ ; \ 0, \frac{1}{2}\right], \quad \wp_2 := \mathcal{C}^{xy}\left[0, \frac{1}{2} \ ; \ \frac{1}{2}, 1\right],$$

$$\wp_3 := \mathcal{C}^{xy}\left[\frac{1}{2}, 1 \ ; \ 0, \frac{1}{2}\right], \quad \wp_4 := \mathcal{C}^{xy}\left[\frac{1}{2}, 1 \ ; \ \frac{1}{2}, 1\right],$$

$$\mathcal{C}_y^x[p,q] := \{e : \quad x(e) \geq y(e) \cdot \frac{a}{b}\}, \quad \mathcal{C}_x^y[p,q] := \{e : \quad x(e) < y(e) \cdot \frac{a}{b}\}.$$

Dizemos que um item s tem dimensão *pequena* no eixo x se $x(s) \leq \frac{a}{m}$, para $m \geq 2$ inteiro. Esta mesma definição é usada para os demais eixos. Dizemos que um item é *pequeno* se este tem dimensões pequenas em relação a todos os eixos. O valor de m ficará em aberto, sendo que os algoritmos desenvolvidos para este tipo de instância terão m como um dos parâmetros de especificação do algoritmo. Em geral, estes algoritmos serão importantes na construção de algoritmos mais genéricos.

Dados uma lista de itens $L = (e_1, e_2, \ldots, e_n)$ e um algoritmo $\mathcal{A}$ para empacotar L, dizemos que $\mathcal{A}$ é um algoritmo *on-line* se

1. $\mathcal{A}$ empacota os itens na ordem dada por L;
2. $\mathcal{A}$ empacota cada item e_i sem levar em conta qualquer item e_j, $j > i$; e
3. $\mathcal{A}$ nunca move um item já empacotado.

Os algoritmos de empacotamento que não são *on-line*, são chamados de algoritmos *off-line*.

Muitas vezes, usamos vários algoritmos *on-line* como subrotinas na construção de um outro algoritmo *on-line*. Nesses casos, para simplificar podemos descrever o novo algoritmo como sendo *off-line*. Notamos porém que um tal algoritmo pode ser transformado em um *on-line* equivalente. A seguir, damos um exemplo de como isto pode ser feito.

Suponha que temos algoritmos *on-line* Alg_1, Alg_2 e Alg_3 para empacotar itens de conjuntos $\mathcal{T}_1$, $\mathcal{T}_2$ e $\mathcal{T}_3$, respectivamente. Considere que $\mathcal{T}_i \cap \mathcal{T}_j = \emptyset$ para $1 \leq i \neq j \leq 3$. Agora considere o seguinte algoritmo, chamado Alg_4, que usa os algoritmos Alg_i, $i = 1, 2, 3$, como subrotinas.

Algoritmo Alg_4

Entrada: Lista de retângulos L.
Saída: Empacotamento de L em placas R.

1 Particione L em sublistas L_1, L_2 e L_3 da seguinte maneira.
$L_i \leftarrow L \cap \mathcal{T}_i$, $i = 1, 2, 3$.

2 $\mathcal{P}_i \leftarrow \text{Alg}_i(L_i)$, $i = 1, 2, 3$.

3 $\mathcal{P} \leftarrow \mathcal{P}_1 \| \mathcal{P}_2 \| \mathcal{P}_3$.

4 Retorne $\mathcal{P}$.

Fim algoritmo.

O algoritmo Alg_4 é *off-line*, mas pode ser facilmente transformado em *on-line*. O seguinte algoritmo é uma descrição *on-line* equivalente do algoritmo Alg_4.

Algoritmo Alg_4'

Entrada: Lista de retângulos $L = (r_1, \ldots, r_n)$.

Saída: Empacotamento de L em placas R.

1 $\mathcal{P} \leftarrow \emptyset$.

2 Para $i = 1$ até n faça

2.1 Seja $\mathcal{P}_j$ o empacotamento em placas de $\mathcal{P}$, que foram geradas pelo algoritmo Alg_j, $j = 1, 2, 3$.

2.2 Seja k tal que $r_i \in \mathcal{T}_k$, $k \in \{1, 2, 3\}$.

2.3 Empacote r_i nas placas de $\mathcal{P}_k$ usando o algoritmo Alg_k. Caso necessário, empacote r_i em uma nova placa em $\mathcal{P}$.

3 Retorne $\mathcal{P}$.

Fim algoritmo.

Desta forma, temos que o algoritmo Alg_4' é *on-line* e o empacotamento gerado por esse algoritmo é igual ao empacotamento gerado pelo algoritmo Alg_4.

Dado um empacotamento $\mathcal{P}$, denotamos por $H(\mathcal{P})$ a altura do empacotamento $\mathcal{P}$ e por $\#(\mathcal{P})$ o número de recipientes usado pelo empacotamento $\mathcal{P}$.

Dada uma lista de itens $L = (e_1, e_2, \ldots, e_n)$, onde cada item e_i é uma t-upla, denotamos por $\Gamma(L)$ o conjunto de listas $\{L' = (e_1', e_2', \ldots, e_n') : e_i' \text{ é uma permutação de } e_i\}$. Dado um item e, definido por uma t-upla, denotamos por $\rho(e)$ a t-upla onde todas as entradas são iguais às de e, exceto nos eixos x e y, onde elas estarão trocadas. Assim, se $e_i = (x_i, y_i, z_i)$ então $\rho(e_i) = (y_i, x_i, z_i)$; e se $e_i = (x_i, y_i)$ então $\rho(e_i) = (y_i, x_i)$. Também denotamos por $\Gamma^z(L)$ o conjunto de listas $\{L' = (e_1', e_2', \ldots, e_n') : e_i' \in \{e_i, \rho(e_i)\}\}$.

Na descrição de alguns algoritmos, usamos as funções $\text{ftype}(L, \mathcal{S})$ (*fixed type*), $\text{rtype}(L, \mathcal{S})$ (*rotatable type*) e $xy\text{-type}(L, \mathcal{S})$ (*xy-rotatable type*), cujas imagens são os seguintes conjuntos:

$$\begin{aligned}
\text{ftype}(L, \mathcal{S}) &:= \{e \in L : e \in \mathcal{S}\}, \\
xy\text{-type}(L, \mathcal{S}) &:= \{e \in L : e \in \mathcal{S} \quad \text{ou} \quad \rho(e) \in \mathcal{S}\}, \\
\text{rtype}(L, \mathcal{S}) &:= \{e \in L : \text{ alguma permutação de } e \text{ pertence a } \mathcal{S}\}.
\end{aligned}$$

1.4 Medidas de Desempenho dos Algoritmos

> "*The asymptotic ratios seem to be the more important ones for bin packing, although for other problems the absolute ratios may be more appropriate*"
>
> Garey e Johnson [25] pág. 128.

São duas as medidas padrões usadas para especificar o desempenho dos algoritmos de aproximação: o limite de desempenho absoluto e o limite de desempenho assintótico.

Dada uma lista L de itens a serem empacotados, e um algoritmo $\mathcal{A}$, denotamos por $\mathcal{A}(L)$ a solução gerada pelo algoritmo $\mathcal{A}$ quando aplicada à lista L. Tal solução tanto pode ser a altura do empacotamento ou o número de recipientes usados pelo empacotamento, conforme o problema em questão. Denotamos por $\mathrm{OPT}(L)$ o valor correspondente a um empacotamento ótimo de L. Apesar de OPT ser usado da mesma forma para problemas distintos, esperamos que seu significado e uso esteja claro pelo contexto.

Consideramos que as dimensões de todos os itens a serem empacotados são limitados por uma constante.

Dizemos que um algoritmo $\mathcal{A}$ tem *um limite de desempenho absoluto* α se

$$\frac{\mathcal{A}(L)}{\mathrm{OPT}(L)} \leq \alpha, \qquad \text{para toda lista de entrada } L.$$

Dizemos que um algoritmo $\mathcal{A}$ tem *um limite de desempenho assintótico* α se existe uma constante β tal que

$$\mathcal{A}(L) \leq \alpha \cdot \mathrm{OPT}(L) + \beta, \qquad \text{para toda lista de entrada } L.$$

Mais ainda, se para todo ϵ pequeno e todo N grande, ambos positivos, existe uma instância L tal que

$$\mathcal{A}(L) > (\alpha - \epsilon) \cdot \mathrm{OPT}(L) \text{ e } \mathrm{OPT}(L) > N,$$

então dizemos que α é um *limite justo*. Neste caso, também dizemos que *o limite de desempenho assintótico* de $\mathcal{A}$, denotado por $r(\mathcal{A})$, é igual a α.

Se para todo $M > 1$, há uma instância L tal que

$$\mathcal{A}(L) > M \cdot \mathrm{OPT}(L),$$

então dizemos que $\mathcal{A}$ tem um *desempenho de pior caso ilimitado*. Neste caso, claramente $\mathcal{A}$ não é um bom algoritmo, do ponto de vista da teoria de algoritmos de aproximação.

1.5 Complexidade dos Problemas de Empacotamento

O Problema de Empacotamento Unidimensional é um problema $\mathcal{NP}$-difícil [25]. Como este problema é um caso particular dos problemas de empacotamento em faixa, placas e contêineres, todos esses problemas são também $\mathcal{NP}$-difíceis. Um caso bem mais restrito do Problema de Empacotamento Unidimensional, que já é $\mathcal{NP}$-completo, é decidir se $3k$ itens $s_1, s_2, \ldots, s_{3k}$ onde cada item s_i tem comprimento $\frac{1}{4} < x(s_i) < \frac{1}{2}$, podem ser empacotados em barras de comprimento 1. Este problema é conhecido como Problema da 3-Partição e foi provado ser $\mathcal{NP}$-completo por Garey e Johnson [24]. Um outro caso que é bem restrito e também é $\mathcal{NP}$-completo, é decidir se uma lista L de itens com $x(L) = 2C$ pode ser empacotada em duas barras de comprimento C. Este problema é conhecido como *problema da partição* e foi um dos problemas $\mathcal{NP}$-completos apresentados por Karp em 1972 [36]. Note que este último problema mostra que o limite de desempenho absoluto do problema de empacotamento unidimensional deve ser pelo menos $\frac{3}{2}$, a menos que $\mathcal{P} = \mathcal{NP}$. Caso exista um algoritmo polinomial com limite de desempenho absoluto $\alpha < \frac{3}{2}$, podemos usá-lo para decidir se uma lista L, com $x(L) = 2C$ pode ser empacotada em 2 barras de comprimento C. Assim, vale o seguinte teorema (*cf.* Garey e Johnson [25]).

Teorema 1.5.1. *O Problema de Empacotamento Unidimensional não é aproximável dentro de* $\frac{3}{2} - \epsilon$, *para qualquer* $\epsilon > 0$, *a menos que* $\mathcal{P} = \mathcal{NP}$.

1.6 Problema de Corte $\times$ Problema de Empacotamento

Muitas aplicações práticas são vistas como problemas de corte, em vez de problemas de empacotamentos. Assim, em vez de um item ser empacotado em um recipiente, este é cortado de um material (recipiente). Basicamente estes problemas podem ser considerados da mesma forma. Mas muitas vezes os problemas de corte têm restrições de como os itens devem ser cortados. Uma destas restrições é a de usar apenas corte-guilhotina.

Dizemos que um corte é do tipo *guilhotina*, ou simplesmente *corte-guilhotina* se este corte é representado por um plano que corta o recipiente de ponta a ponta e é ortogonal a algum dos eixos do empacotamento. Um corte guilhotina divide o recipiente em duas partes que podem ser novamente submetidas a cortes-guilhotina. Um empacotamento é dito *guilhotinável* se todos os itens empacotados podem ser separados um dos outros a partir de uma seqüência de um ou mais cortes-guilhotina. Este tipo de corte é muito comum em problemas de corte de vidros, espumas ou isopor.

O leitor pode encontrar outras restrições práticas na resenha de Yanasse, Furtado, Lorena, Arenales, Soma, Maculan e Morábito [60].

1.7 Considerações sobre Rotações Ortogonais

Muitos dos problemas aqui tratados têm versões em que os itens podem sofrer rotações quanto a alguns dos eixos. Consideramos apenas rotações ortogonais (*i.e.,* de 90°). Como pode ser verificado no Apêndice B, há muitas aplicações relativas a essas versões. Na literatura encontramos poucos resultados a respeito dessas versões, no que concerne ao desenvolvimento de algoritmos de aproximação. A maior parte dos resultados relativos a algoritmos de aproximação é referente ao caso com orientação fixa. Ao longo desta tese, apresentamos vários algoritmos para problemas com rotações.

Quando consideramos itens pequenos em um problema permitindo rotações, consideramos que os itens são pequenos em relação a uma dada orientação inicial.

Uma primeira idéia para resolver os problemas que permitem rotação é adaptar algoritmos do caso orientado para o caso com rotação. Considere PROB um dos problemas descritos anteriormente, para o caso orientado, e PROB^r para o caso com rotação. Dizemos que um item e tem orientação viável se $x(e) \leq a$, $y(e) \leq b$ e $z(e) \leq c$, onde a, b e c são as dimensões máximas que um item pode ter nas respectivas dimensões. Uma abordagem natural, é gerar para cada instância $L = (e_1, e_2, \ldots, e_n)$ de PROB^r, uma nova instância L' de PROB, tal que $L' \in \phi(L) = (d_1, d_2, \ldots, d_n)$, onde

$$d_i = \begin{cases} e_i, & \text{se } e_i \text{ tem orientação viável,} \\ e_i' & \text{caso contrário, onde } e_i' \text{ é uma permutação de } e_i \text{ que tem orientação viável;} \end{cases}$$

e então aplicar um algoritmo de PROB sobre L'. Desta maneira, obtemos também um empacotamento para PROB^r.

Para cada algoritmo $\mathcal{A}$ para o caso orientado, denote por $\widehat{\mathcal{A}}$ um correspondente algoritmo para PROB^r, como descrito acima. Isto é, para toda instância L do PROB^r, o algoritmo $\widehat{\mathcal{A}}$ aplica o algoritmo $\mathcal{A}$ sobre uma lista de $\phi(L)$. É fácil ver que este algoritmo $\widehat{\mathcal{A}}$ pode não preservar o limite de desempenho assintótico original de $\mathcal{A}$.

Vamos exemplificar isto para o PEP^r. O próximo resultado, é uma adaptação de um resultado provado em [46] do PET^r para o problema de empacotamento em placas. O resultado mostra que não existe algoritmo $\widehat{\mathcal{A}}$ para o PEP^r, obtido de um algoritmo $\mathcal{A}$ para o PEP —como descrito anteriormente—, que tem um limite de desempenho assintótico menor que 3.

Proposição 1.7.1. *Se $\widehat{\mathcal{A}}$ é um algoritmo para* PEP^r *obtido de um algoritmo $\mathcal{A}$ para o* PEP*, como descrito acima, então o limite de desempenho assintótico de $\widehat{\mathcal{A}}$ é pelo menos 3.*

Prova. Para ver isto, considere a seguinte instância $L = (r_1, r_2, \ldots, r_{3k})$ do PEP^r para empacotamento em placas (3,75 , 2), onde $r_1 = r_2 = \ldots = r_{3k} = (2\ ,\ 1{,}25)$ e k é um inteiro positivo.

Primeiramente, observe que é possível empacotar L em k níveis, *i.e.*, $\text{OPT}(L) = k$. Para verificar isto, gire cada retângulo de L previamente e construa um empacotamento colocando três retângulos por placa.

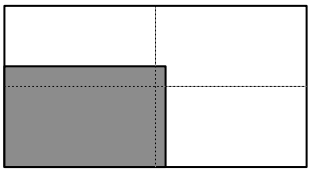


Figura 1.1: Um retângulo por placa no empacotamento de L gerado por $\widehat{\mathcal{A}}$.

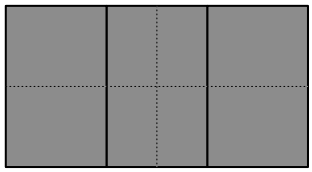


Figura 1.2: Três retângulos por placa no empacotamento ótimo de L.

Por outro lado, note que $\phi(L) = \{L\}$ e qualquer algoritmo $\mathcal{A}$ do PEP é tal que $\widehat{\mathcal{A}}(L) \geq 3k$, já que qualquer algoritmo do PEP empacota cada retângulo de L em apenas uma placa. Portanto, o limite de desempenho assintótico de $\widehat{\mathcal{A}}$ é pelo menos 3. □

Para todos os próximos algoritmos apresentados nesta tese, consideramos sem perda de generalidade que $L \in \phi(L)$.

Em [7], Coffman, Garey e Johnson (1984) discutem a possibilidade de que, ao se permitir rotações ortogonais, algoritmos com limites de desempenho melhores do que aos conhecidos para o problema com orientação fixa possam ser encontrados. O seguinte texto, onde é discutido o Problema de Empacotamento em Placas e Faixa, justifica esta afirmação:

> "*Having introduced the case where ninety degree rotations are allowed, we should mention that some of the worst case results mentioned above also apply to this case, in that the values of R_A and R_A^∞ are unchanged if such rotations are allowed in the construction of optimal packings. This holds true in particular for NFDH and BLDW, since the proofs of the bounds for these algorithms are based on pure area arguments. So far no algorithm has been found that attains improved guarantees by actually using such rotations itself ...*"
>
> E. G. Coffman, M. R. Garey e D. S. Johnson [7].

Na discussão acima, R_A e R_A^∞ são os limites de desempenho absoluto e assintótico, respectivamente. O seguinte trecho foi extraído de outro artigo [6], sobre o Problema de Empacotamento em Placas.

> "*4. Directions for further research [...] A second line of attack would be to design and analyse algorithms which could make use of the fact that, in some applications,* 90° *rotations of rectangles might be allowable.*
>
> *Algorithms which consider the possibility of rotations might well yield improvements. Can one prove worst case bounds that reflect these improvementes ?*"
>
> F. R. Chung, M. R. Garey e D. S. Johnson [6].

No contexto acima, "*worst case bounds*" são relativos aos limites de desempenho como definimos. Em [9] Coffman *et al.* também questionam sobre rotações ortogonais.

As considerações acima nos levam a pensar que os autores destes artigos achavam que a possibilidade de girar os itens poderia levar a algoritmos com melhores limites de desempenho. Apesar de esses artigos serem do início da década de 80, desde então pouco foi feito sobre as versões onde rotações são permitidas.

Veremos a seguir que, para qualquer um dos casos gerais dos problemas descritos na seção 1.2, o problema de empacotamento com rotação é tão difícil (no sentido de $\mathcal{NP}$-difícil) quanto o problema totalmente orientado. Mais ainda, se $\mathcal{A}^r$ é um algoritmo para um dos problemas permitindo rotações, então este algoritmo pode ser usado por um outro algoritmo $\mathcal{A}'$, para o problema equivalente totalmente orientado, de forma a manter o mesmo limite de desempenho.

Ou seja, suponha que $\mathcal{A}^r$ tem limite de desempenho assintótico do tipo:

$$\mathcal{A}^r(L) \leq \alpha \cdot \mathrm{OPT}(L) + \beta, \quad \text{para toda instância } L \text{ (do caso com rotação).}$$

Então é possível usar este algoritmo, para obter um outro algoritmo $\mathcal{A}'$, para o problema totalmente orientado, com o mesmo limite de desempenho assintótico. Ou seja,

$$\mathcal{A}'(L) \leq \alpha \cdot \mathrm{OPT}(L) + \beta, \quad \text{para toda instância } L \text{ (do caso orientado).}$$

A prova da afirmação acima é relativamente simples. Considere por exemplo o Problema de Empacotamento Bidimensional em Placas orientado (onde rotações não são permitidas) e a versão em que rotações ortogonais (de 90°) são permitidas.

Dada uma instância $\mathcal{I}$ para a versão orientada, que consiste de uma lista $L = (r_1, \ldots, r_n)$ e placas $R = (a, b)$, redimensione uma das dimensões das placas $R = (a, b)$ e dos retângulos de L na mesma proporção, obtendo placas R' e lista L', de forma que nenhum retângulo da nova lista L' possa ser empacotado nas placas R' se sofrer uma rotação ortogonal (*cf.* Figura 1.3). A seguir, use um algoritmo (digamos $\mathcal{A}^r$) da versão permitindo rotação sobre a instância redimensionada.

É fácil ver que este novo algoritmo resolve o problema totalmente orientado (já que nenhum retângulo sofre rotação).

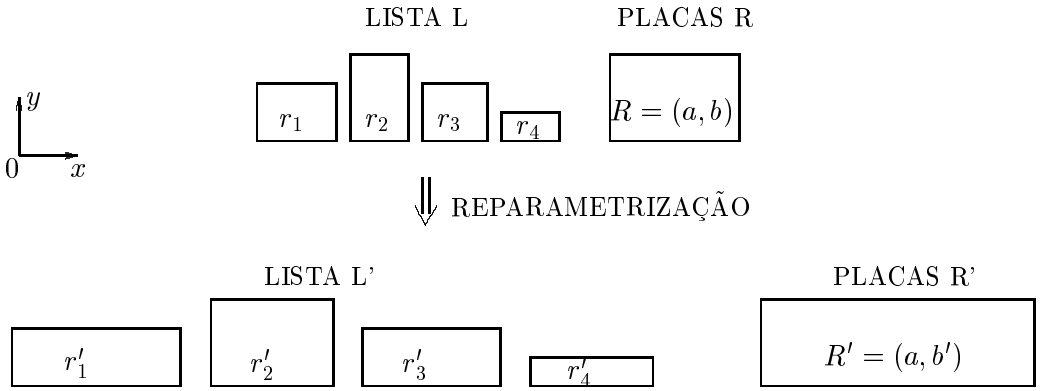


Figura 1.3: Reparametrização de instância (no eixo x) para desabilitar rotações ortogonais.

Este mesmo raciocínio pode ser usado para todos os problemas listados que consideram versões com rotação. Assim, se tivermos um algoritmo para o problema tridimensional permitindo rotação em torno dos eixos x, y e z, então usando o procedimento acima este algoritmo mantém o limite de desempenho, tanto para o problema totalmente orientado quanto para o problema em que permitimos rotações em torno de apenas um eixo. Assim, vale o seguinte resultado.

Teorema 1.7.2. *Seja* PROB^r *um dos problemas definidos anteriormente,* α *e* β *constantes, e* $\mathcal{A}^r$ *um algoritmo considerando rotações de* 90° *em torno de algum dos eixos* x *ou* y *ou* z *(podendo ser em vários eixos) tal que,*

$$\mathcal{A}^r(L) \leq \alpha \cdot \mathrm{OPT}(L) + \beta \quad \textit{para toda instância } L \textit{ de } \mathrm{PROB}^r.$$

Então, podemos adaptar este algoritmo para um algoritmo $\mathcal{A}$ *para a variante de* PROB^r*, chamado de* PROB*, onde fixamos a orientação dos itens sobre alguns dos eixos, de tal modo que a seguinte relação é válida para esta variante:*

$$\mathcal{A}(L) \leq \alpha \cdot \mathrm{OPT}(L) + \beta \quad \textit{para toda instância } L \textit{ de } \mathrm{PROB}.$$

O resultado acima refere-se aos casos gerais dos problemas em questão. Note que o procedimento descrito conduz a um algoritmo *off-line*, pois para poder achar um redimensionamento da instância, fixando-a (orientando) em alguns dos eixos, foi necessário encontrar o tamanho do menor item neste eixo e redimensionar a instância de forma que todos os itens ficassem orientados neste eixo.

A redução acima é polinomial (considerando uma representação conveniente para as instâncias), donde segue que PROB é $\mathcal{NP}$-difícil.

Observamos que, caso o algoritmo para PROB^r seja *on-line*, é possível construir um algoritmo para PROB usando esta construção e ainda mantendo o algoritmo *on-line*. Para isso, basta

manter dois empacotamentos em construção, digamos $\mathcal{P}$ e $\mathcal{P}^r$. O empacotamento $\mathcal{P}$ será o empacotamento gerado pelo algoritmo propriamente dito, e $\mathcal{P}^r$ será um empacotamento auxiliar de uso interno do algoritmo para PROB^r. Os empacotamentos $\mathcal{P}$ e $\mathcal{P}^r$ serão os mesmos, diferindo apenas em uma reparametrização de seus dados. O empacotamento $\mathcal{P}$ terá os valores como dados na entrada.

Inicialmente, $\mathcal{P}$ e $\mathcal{P}^r$ são empacotamentos vazios. Como sabemos, um algoritmo *on-line* deve empacotar os itens na ordem em que ocorrem na lista de entrada L. Para empacotar o próximo item, digamos e da lista L, primeiramente é feita uma cópia, e^r, de e, reparametrizada nas mesmas proporções de $\mathcal{P}^r$. Se este item e^r permite ainda ser girado no problema de empacotamento ligado a $\mathcal{P}^r$, de forma a ficar inviável para PROB, então reparametrize $\mathcal{P}^r$ e e^r nas mesmas proporções de forma que as possíveis rotações não permitidas em PROB não mais existam. Com isto, e^r é empacotado em $\mathcal{P}^r$ usando o algoritmo de PROB^r e em seguida o item e é empacotado em $\mathcal{P}$ em posição equivalente à de e^r em $\mathcal{P}^r$. Desta forma, este algoritmo ainda se mantém *on-line* e o mesmo resultado do Teorema 1.7.2 é válido para o caso *on-line*.

Teorema 1.7.3. *Se existe um algoritmo (on-line) para um dos problemas de empacotamento no caso geral, considerando rotações ortogonais sobre alguns dos eixos e com limite de desempenho α, então existe algoritmo (on-line) para a versão deste problema orientando o empacotamento em algum dos eixos (ou mesmo totalmente orientado) com o mesmo limite de desempenho assintótico α.*

Prova. Segue usando os mesmos argumentos usados acima. □

Ainda que as versões dos problemas de empacotamento considerando rotações ortogonais pareçam tão difíceis quanto suas respectivas versões orientadas, isto não quer dizer que esta dificuldade sempre vá ocorrer. Há casos onde restringimos o problema, considerando casos especiais, e tiramos proveito das rotações ortogonais. Isto será visto em algoritmos para casos especiais do Problema de Empacotamento em Placas e o Problema de Empacotamento Tridimensional, onde realmente conseguimos tirar proveito das rotações ortogonais para obter um melhor limite de desempenho assintótico.

O que nos parece também é que se considerarmos os casos gerais, o problema permitindo rotações ortogonais é mais complicado que sua versão orientada. Observamos que certos casos especiais são extremamente fáceis se considerados na versão totalmente orientada, mas ficam complicadas na versão onde rotações ortogonais são permitidas. Um exemplo disso são os problemas de carregamento de paletes (*pallet loading*), que descrevemos a seguir.

Dados uma lista L de retângulos iguais e um recipiente retangular, o objetivo é empacotar o maior número possível de retângulos de L no recipiente, permitindo-se que os itens de L sofram rotações ortogonais.

Claramente, se os retângulos não pudessem sofrer rotações seria fácil conseguir um algoritmo ótimo para este empacotamento: bastaria colocar os retângulos um ao lado do outro, obtendo assim um empacotamento homogêneo. Por outro lado, permitindo-se rotações ortogonais uma solução ótima não é nada óbvia. Algumas pesquisas têm sido conduzidas sobre este tema [17, 18, 50, 56], mas não conhecemos nenhuma prova de que este problema é realmente $\mathcal{NP}$-difícil.

Na mesma linha, em [29], Hoffman apresenta um problema especial de empacotamento tridimensional onde rotações ortogonais podem ser feitas. O problema consiste em encontrar um empacotamento de um conjunto de caixas idênticas em um contêiner de dimensões especificadas, de modo a minimizar o número de contêineres. Hoffman apresenta algumas condições sobre os dados da instância, de forma a tornar a busca do empacotamento ótimo mais complicada[1].

Muitas vezes, usaremos os algoritmos desenvolvidos para o caso orientado em problemas onde rotações em torno de alguns eixos são permitidas. A única consideração a ser feita é que os itens a serem empacotados devem estar em orientação viável para a construção do empacotamento por estes algoritmos.

[1] Hoffman diz ter apresentado este problema à David Klarner, que construiu 27 caixas de dimensões $7 \times 8 \times 10$ para serem empacotadas dentro de uma caixa de dimensão $25 \times 25 \times 25$, obtendo assim um belo quebra-cabeça.

Capítulo 2

Problema de Empacotamento Unidimensional

2.1 Introdução

Neste capítulo descrevemos vários algoritmos de empacotamento unidimensional. Apresentamos além daqueles que serão utilizados nos algoritmos desenvolvidos nesta tese (para casos bi- e tridimensionais) alguns outros devido a sua importância no desenvolvimento de outros algoritmos de aproximação que serão mencionados aqui.

As idéias presentes nos algoritmos de empacotamentos unidimensionais têm sido bastante usadas em algoritmos de empacotamentos para problemas bi-, tri- e até multi-dimensionais. Muitos desses algoritmos são bastante simples e elegantes. E muitas vezes, esta simplicidade não deixa de estar associada a bons limites de desempenho assintóticos e absolutos. Os resultados relativos a tais limites de desempenho não serão provados (indicamos as referências onde podem ser encontradas as provas destes resultados).

Descrevemos resultados referentes aos algoritmos NF *(Next Fit)*, FF *(First Fit)*, BF *(Best Fit)*, H_M *(Harmonic M)*, FFD *(First Fit Decreasing)* e VL_ϵ *(Vega and Lucker algorithm)*. Ao final, apresentamos algumas técnicas usadas para os problemas bidimensionais e tridimensionais aplicadas ao caso unidimensional.

Os algoritmos NF, FF, BF, H_M são algoritmos *on-line*. Os limites de desempenho assintótico dos algoritmos NF, FF e BF são 2 (*cf.* Johnson [30]); 1,7 (*cf.* Garey, Graham e Ullman [23]) e 1,7 (*cf.* Johnson, Demers, Ullman, Garey e Graham [33]), respectivamente. O Algoritmo H_M —que depende do parâmetro M— pode ter seu limite de desempenho assintótico tão próximo de 1,6910... quanto se queira (*cf.* Lee e Lee [37]), bastando para isto escolher um valor bem grande para M. Há que se notar, porém, que o valor da constante aditiva β, associado a este limite, cresce à medida que o valor de M cresce. O Algoritmo H_M é de grande importância neste

contexto, pois todos os melhores limites de desempenho assintóticos de algoritmos *on-line* para o caso geral, para a maioria dos problemas estudados nesta tese, se baseiam neste algoritmo ou em alguma variante deste.

Os algoritmos FFD e VL_ϵ são *off-line*. O Algoritmo FFD tem limite de desempenho assintótico igual a $\frac{11}{9} = 1{,}222\ldots$ e o Algoritmo VL_ϵ pode ter seu limite de desempenho assintótico tão próximo de 1 quanto se queira, bastando para isto tomar o valor de ϵ tão próximo de 0 quanto se queira. Aqui também, à medida que ϵ diminui, o valor de β, a constante aditiva associada ao limite, cresce.

Em [55], Simchi-Levi provou que os algoritmos FF e BF têm limites de desempenho absoluto iguais a 1,75. Neste mesmo artigo o autor também prova que o limite de desempenho absoluto do algoritmo FFD é 1,5. Observamos que este último resultado é o melhor possível, no sentido que não há algoritmo para o PEU com limite de desempenho absoluto menor que 1,5, a menos que $\mathcal{P} = \mathcal{NP}$ [25].

Além destes algoritmos, existem outros com limites de desempenho assintótico muito bons, pelo menos do ponto de vista teórico. Para empacotamentos *on-line*, Richey [52], desenvolveu um algoritmo com um limite de desempenho assintótico 1,5888. Por outro lado, van Vliet [59] mostrou que todo algoritmo deste tipo deve ter limite de desempenho assintótico pelo menos 1,5401.

Considerando itens pequenos, ($L \subset \mathcal{X}[0, \frac{1}{m}]$), Johnson *et al.* [33] mostraram que o limite de desempenho do Algoritmo FF é de $\frac{m+1}{m}$ e Csirik [13] mostrou que o Algoritmo FFD tem limite de desempenho assintótico de $(m+3)/(m+2) - 2/(m(m+1)(m+2))$ para m par e de $(m+3)/(m+2) - 1/(m(m+1)(m+2))$ para m ímpar. Em [5], Brown, Baker e Katseff mostraram que $\frac{m^3}{m^3-m+1}$ é um limite inferior para o limite de desempenho assintótico para qualquer algoritmo *on-line* do PEU com itens de L ordenados de forma não crescente.

Karmarkar e Karp [35] desenvolveram um algoritmo que garante um empacotamento que não usa mais do que $\mathrm{OPT}(L) + O(log^2 \mathrm{OPT}(L))$ barras.

Uma resenha dos principais resultados sobre os problemas de empacotamento unidimensional pode ser encontrada em [7] e [8].

Não mencionamos os autores de alguns algoritmos que descrevemos aqui, mas quando possível, mencionamos os autores dos resultados relacionados a estes algoritmos. O motivo disto é que muitos algoritmos para o caso unidimensional já eram usados na prática muito antes de aparecerem em artigos científicos. Muitos destes eram usados ditados pela intuição de um bom desempenho, mas faltava ainda uma análise mais formal que comprovasse os resultados da prática.

Mostraremos neste capítulo algoritmos clássicos para o problema unidimensional, indicando seus limites de desempenho assintótico. Os algoritmos apresentados serão divididos em *on-line*

e *off-line*.

Vamos descrever primeiro os algoritmos *on-line* e em seguida os algoritmos *off-line*. A descrição destes algoritmos é feita apenas informalmente, uma vez que eles são bem simples.

2.2 Definições

A seguir, definimos o problema de empacotamento unidimensional.

Problema. *Problema de Empacotamento Unidimensional (PEU): Dados uma constante C e uma lista de itens $L = (p_1, p_2, \ldots, p_n)$, onde a cada item p_i está associado um valor (ou comprimento) $x(p_i)$ satisfazendo $0 < x(p_i) < C$, encontrar o menor inteiro m tal que L pode ser particionado em m listas $L_1, L_2, \ldots, L_m$, onde cada L_i satisfaz $x(L_i) := \sum_{j \in L_i} x(p_j) \leq C$, $i = 1, \ldots, m$.*

Um empacotamento $\mathcal{P}$ de uma lista de itens $L = (p_1, p_2, \ldots, p_n)$, satisfazendo as condições acima, é uma função $\mathcal{P} : L \rightarrow (\mathbb{Z}^+, [0, C))$. Denotamos por $(\mathcal{P}^b(p), \mathcal{P}^x(p))$ o par associado a $\mathcal{P}(p)$, onde p é um item a ser empacotado. $\mathcal{P}$ deve ser tal que se $\{L_1, L_2, \ldots, L_m\}$ é uma partição de L viável para o PEU, então $\mathcal{P}^b(p) = k$ se $p \in L_k$, e $\mathcal{P}^x(p_j^k) = \sum_{i=1}^{j-1} x(p_i^k)$, sendo $L_k = (p_1^k, p_2^k, \ldots, p_{n_k}^k)$, $k = 1, \ldots, m$.

Cada lista L_k indica quais são os itens que serão empacotados em uma mesma barra vista como sendo o conteúdo de uma *barra* de capacidade C, e o objetivo do problema é minimizar o número de barras necessárias para empacotar L.

Na Figura 2.1 ilustramos uma instância para o PEU e na Figura 2.2 ilustramos um empacotamento de barras.

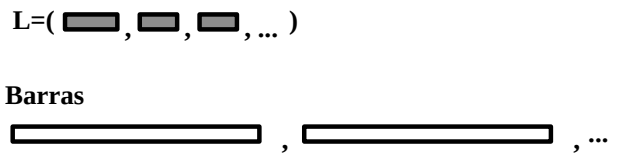


Figura 2.1: Instância para o PEU.

Figura 2.2: Empacotamento para o PEU.

Usamos a notação PEU(C) para nos referirmos ao problema PEU com parâmetro de entrada C.

2.3 Algoritmos *On-Line*

2.3.1 Algoritmo NF

Provavelmente o primeiro a usar o termo *Next Fit* foi Johnson [30] (*cf.* Coffman *et al.* [8]). Este talvez seja o algoritmo de aproximação mais simples para o problema de empacotamento unidimensional. Este algoritmo pode ser descrito da seguinte forma.

Dada uma lista de itens $L = (p_1, \ldots, p_n)$, o Algoritmo NF considera para cada item a ser empacotado apenas uma barra, a *barra corrente*, e gera uma partição de L, gerando primeiro uma sublista L_1, depois uma sublista L_2, e assim por diante. Cada lista L_i conterá os itens que serão empacotados em uma mesma barra. O algoritmo sempre testa cada item (na ordem que ocorre em L), verificando se o mesmo pode ser empacotado na lista corrente L_i. Enquanto isto é possível, os itens são empacotados em L_i. Quando um item p não pode ser empacotado em L_i, o algoritmo empacota p em uma nova lista L_{i+1}, que passa a ser a nova lista corrente. O algoritmo pára quando todos os itens tiverem sido empacotados, e retorna a partição $(L_1, \ldots, L_m)$, onde L_m é a última lista gerada pelo algoritmo.

O seguinte resultado é válido para este algoritmo [30, 8].

Teorema 2.3.1. *Para toda lista de itens L, tem-se que* $\mathrm{NF}(L) \leq 2 \cdot \mathrm{OPT}(L) - 1$. *Mais ainda, o limite de desempenho assintótico do Algoritmo NF é justo.*

2.3.2 Algoritmo FF

O Algoritmo NF, sempre que vai empacotar um item na barra corrente, nunca testa nas barras anteriores. Neste aspecto os algoritmos FF e BF podem ser considerados como melhorias do Algoritmo NF, já que estes testam se um item pode ser empacotado nas barras anteriormente criadas. O Algoritmo FF pode ser descrito da seguinte forma.

Seja $\{L_1, L_2, \ldots, L_k\}$ a partição gerada pelo Algoritmo FF até um certo momento. Para cada item p_j de uma lista de itens $L = (p_1, \ldots, p_n)$ a ser empacotada o algoritmo procura a lista L_i com o menor índice i tal que $x(L_i) + x(p_j) \leq C$. Caso encontre, insere p_j na lista L_i. Caso contrário, uma nova lista L_{k+1} é inserida na partição contendo o item p_j. Este procedimento é repetido até que todos os itens sejam empacotados.

Em 1972, Garey *et al.* [23], mostraram que o limite de desempenho assintótico do Algoritmo FF é 1,7.

Teorema 2.3.2. *Para toda lista de itens L, tem-se que* $\mathrm{FF}(L) \leq 1{,}7 \cdot \mathrm{OPT}(L) + 2$. *Mais ainda, o limite de desempenho assintótico* 1,7 *do Algoritmo FF é justo.*

E em 1976, Garey *et al.* [22] mostraram que o resultado acima pode ser melhorado para $\mathrm{FF}(L) \leq \lceil 1{,}7 \cdot \mathrm{OPT}(L) \rceil$.

2.3.3 Algoritmo BF

O Algoritmo BF se assemelha ao Algoritmo FF. A diferença está no modo como a lista L_i, escolhida para receber o novo item p_j, é determinada. O Algoritmo BF procura uma lista L_i tal que $x(L_i) + x(p_j) \leq C$ e $x(L_i)$ é máximo entre as listas L_i da partição corrente. Caso haja duas ou mais listas da partição onde as duas condições sejam satisfeitas é escolhida a lista de menor índice.

Em [33] Johnson *et al.* provaram o seguinte resultado sobre o Algoritmo BF.

Teorema 2.3.3. *Para toda lista de itens L, tem-se que* $\mathrm{BF}(L) \leq 1{,}7 \cdot \mathrm{OPT}(L) + 2$. *Mais ainda, o limite de desempenho assintótico* 1,7 *do Algoritmo BF é justo.*

2.3.4 Algoritmo $\mathrm{H_M}$

O Algoritmo $\mathrm{H_M}$ foi desenvolvido por Lee e Lee [37] e pode ser descrito da seguinte forma.

Para cada inteiro positivo $M \geq 2$ definimos um algoritmo denotado por H_M.

O Algoritmo H_M divide o intervalo $(0, C]$ em M subintervalos $I_k,\ k = 1, \ldots M$, $[0, C) = \cup_{k=1}^{M} I_k$, onde

$$I_k = \begin{cases} \left(\frac{C}{k+1}, \frac{C}{k}\right] & \text{para } k = 1, \ldots, M-1 \\ \\ \left(0, \frac{C}{M}\right] & \text{para } k = M\ . \end{cases}$$

Um item p_i é chamado um I_k-*item* se $x(p_i) \in I_k,\ k = 1, \ldots, M$. O Algoritmo H_M empacota os itens na ordem dada por L. Dado um item p, o Algoritmo H_M toma o valor k tal que p é um I_k-item. Em seguida o algoritmo tenta empacotar p na última barra onde um I_k-item foi empacotado. Caso isto não seja possível, o algoritmo empacota p em uma nova barra. O algoritmo termina quando todos os itens de L tiverem sido empacotados.

Lee e Lee mostraram o seguinte resultado sobre o limite de desempenho assintótico deste algoritmo.

Teorema 2.3.4. *Para toda lista de itens L, tem-se que*

$$\mathrm{H}_M(L) \leq \alpha_M \cdot \mathrm{OPT}(L) + M - 1,$$

onde $\alpha_M = \sum_{j=1}^{i} \frac{1}{c_j} + \frac{M}{c_{i+1}(M-1)}$, *e com* $c_i < M \leq c_{i+1}$ *para algum* $i \geq 1$, *sendo* $c_1, c_2, \ldots$ *definidos como* $c_1 = 1$ *e* $c_i = c_{i-1}(c_{i-1} + 1)$ *para* $i \geq 2$. *Mais ainda, o limite de desempenho assintótico* α_M *do Algoritmo* H_M *é justo. Além disso,* $\lim_{M \to \infty} \alpha_M = 1{,}691\ldots$

2.4 Algoritmos *Off-Line*

A seguir, consideramos os algoritmos FFD e VL_ϵ.

2.4.1 Algoritmo FFD

O Algoritmo FFD, é bem simples, elegante e tem um limite de desempenho assintótico bastante bom. Pode ser descrito como segue.

Algoritmo FFD

Entrada: Lista $L = (p_1, \ldots, p_n)$ e constante C.
Saída: Partição $\{L_1, \ldots, L_m\}$ de L onde cada L_i $(1 \leq i \leq m)$ satisfaz $\sum_{p \in L_i} x(p) \leq C$.

1. Faça uma ordenação não-crescente da lista L.

2. Aplique o Algoritmo FF à nova lista.

3. Retorne a solução encontrada.

Fim algoritmo.

Em 1973, Johnson [30] mostrou que o Algoritmo FFD tem limite de desempenho assintótico de 11/9. Mais precisamente,

Teorema 2.4.1. *Para toda lista de itens L,*

$$\mathrm{FFD}(L) \leq \frac{11}{9}\mathrm{OPT}(L) + 4 \ .$$

Em 1991, Yue [61] mostrou que a constante 4, dada no teorema acima, pode ser trocada por 1.

2.4.2 Algoritmo VL_ϵ

O Algoritmo VL_ϵ que será usado, mas não será apresentado, foi desenvolvido por Fernandez de la Vega e Lucker [16] e constitui um esquema de aproximação assintótico para o problema de empacotamento unidimensional. O leitor interessado pode encontrar sua descrição em [16]. Este algoritmo depende de um valor positivo ϵ que pode ser tomado tão próximo de 0 quanto se queira. Estes autores provaram o seguinte resultado para este algoritmo.

Teorema 2.4.2. *Para todo $\epsilon > 0$, existe um algoritmo linear VL_ϵ tal que $\mathrm{VL}_\epsilon(L) \leq (1+\epsilon) \cdot \mathrm{OPT}(L) + \left(\frac{1}{\epsilon}\right)^2$.*

Em [8], Coffman *et al.* mostram que o Algoritmo VL_ϵ pode ser implementado de forma a ficar com complexidade de tempo $\mathcal{O}(n \log n)$, onde n é o número de itens na lista L; além disso, abaixando a constante aditiva para $\frac{1}{\epsilon}$.

Teorema 2.4.3. *Para todo $\epsilon > 0$, o Algoritmo VL_ϵ pode ser implementado em tempo polinomial, com garantia $\mathrm{VL}_\epsilon(L) \leq (1+\epsilon) \cdot \mathrm{OPT}(L) + \left(\frac{1}{\epsilon}\right)$.*

Este algoritmo, VL_ϵ, é um FPAS (*Fully Polynomial Approximate Scheme* [25]) e tem complexidade de tempo $\mathcal{O}(\epsilon^{-c} n \log n)$ para uma constante positiva c. Usando técnicas mais sofisticadas, Karmarkar e Karp desenvolveram um algoritmo polinomial $\mathcal{A}$ tal que $\mathcal{A}(L) \leq \mathrm{OPT}(L) + \mathcal{O}\left(\log^2(\mathrm{OPT}(L))/\mathrm{OPT}(L)\right)$.

2.5 Algumas Técnicas

Para introduzir o leitor às técnicas e idéias presentes nos algoritmos para os problemas de empacotamento bi- e tridimensionais, vejamos algumas das idéias que usamos (algumas delas desenvolvidas nesta tese) nos algoritmos para o PEU. Estas técnicas aplicadas a estes exemplos não melhoram nenhum limite de desempenho já existente para o PEU, mas ilustram o uso das mesmas assim como as suas análises para algoritmos simples do PEU. Assim, é importante entender estas técnicas uma vez que elas serão muito usadas nos capítulos seguintes. Sem perda de generalidade, vamos considerar nesta seção o PEU(1).

2.5.1 Usando Garantia de Comprimento

Uma das técnicas mais usadas na análise de limites de desempenho, é garantir um mínimo de comprimento usado pelos itens empacotados em cada barra.

Um exemplo simples disto pode ser observado no Algoritmo NF. Seja $\mathcal{P}$ o empacotamento de uma lista de itens L gerado pelo Algoritmo NF; $B_1, B_2, \ldots, B_k$, $k \geq 1$, as barras usadas pelo Algoritmo NF, nesta ordem, e $x(B_i)$ a soma dos comprimentos dos itens empacotados em B_i.

Considere a soma $S_i := x(B_{2i-1}) + x(B_{2i})$, $i = 1, \ldots, \lfloor \frac{k}{2} \rfloor$. Seja s o item que o Algoritmo NF não conseguiu empacotar na barra B_{2i-1}, e que foi empacotada na barra B_{2i}. Neste caso, temos que $x(B_{2i-1}) + x(s) > 1$, e portanto, $S_i > 1$.

Assim, se considerarmos o empacotamento $\mathcal{P}$ como um todo, o comprimento de cada barra B_i que é ocupada pelos itens é pelo menos $\frac{1}{2}$. Somando todos os valores de S_i, $i = 1, \ldots, \lfloor \frac{k}{2} \rfloor$ temos que

$$x(L) \geq \sum_{i=1}^{\lfloor k/2 \rfloor} S_i > \left\lfloor \frac{k}{2} \right\rfloor,$$

ou seja,

$$x(L) \geq \left\lfloor \frac{k}{2} \right\rfloor \geq \frac{k-1}{2}.$$

Substituindo k por $\mathrm{NF}(L)$, obtemos

$$\mathrm{NF}(L) \leq \frac{1}{1/2} x(L) + 1. \tag{2.1}$$

Como $x(L) \leq \mathrm{OPT}(L)$,

$$\mathrm{NF(L)} \leq 2 \cdot \mathrm{OPT}(L) + 1.$$

A fração $\frac{1}{2}$ na inequação (2.1) é considerada uma *garantia de comprimento* do empacotamento gerado pelo Algoritmo NF e representa como um todo, a ocupação mínima dos itens nas barras (em relação ao comprimento). Assim, temos que 2 é um limite de desempenho assintótico do Algoritmo NF. O leitor pode facilmente mostrar que 2 é um limite de desempenho absoluto do Algoritmo NF. Mais ainda, este limite é justo quando consideramos uma lista com itens dos seguintes valores $(\frac{1}{2}, \epsilon, \frac{1}{2}, \epsilon, \ldots, \frac{1}{2})$. Colocando ϵ pequeno, é fácil ver que em cada barra o comprimento ocupado pelos itens é próximo de $\frac{1}{2}$.

Por outro lado, não podemos ter garantia de comprimento melhor que $\frac{1}{2}$. Considere listas de itens onde todos os itens têm valor $\frac{1}{2} + \epsilon$. Colocando ϵ bem pequeno, isto mostra que a ocupação dos itens em cada barra não pode ter garantia maior que $\frac{1}{2}$. Por outro lado, o leitor pode estar pensando: "Por que olhar para a garantia de comprimento, deste exemplo, se colocar um item por barra já nos dá um empacotamento ótimo deste tipo de itens ?" De fato, esta instância em particular mostra bem que não devemos considerar apenas o comprimento garantido pelos itens em cada barra. Em alguns casos podemos obter um empacotamento ótimo ou próximo do ótimo para algumas instâncias.

A seguinte subseção mostra como analisar empacotamentos considerando tanto a garantia de comprimento como a análise de instâncias particulares que facilmente geram empacotamento ótimo.

2.5.2 Ótimo × Garantia de Comprimento

Em 1989, Coppersmith e Raghavan [12] analisaram o limite de desempenho assintótico de um empacotamento considerando duas partes. Uma parte pela garantia de espaço (como na garantia de comprimento usada na subseção anterior) e na outra parte usaram a informação de que em alguns empacotamentos é possível conseguir um empacotamento ótimo. Eles apresentaram esta idéia para o problema de empacotamento em placas e em contêineres. A seguir, adaptamos esta idéia para o problema de empacotamento unidimensional.

Considerando que o valor da garantia de comprimento tem um papel fundamental no limite de desempenho do Algoritmo NF, uma estratégia natural seria dividir a lista L em sublistas

de forma que ao aplicar o Algoritmo NF em cada sublista tenhamos uma melhor garantia de comprimento. Com isso, podemos também identificar as sublistas que fornecem pouca garantia de comprimento. Vejamos como seria o comportamento de um algoritmo que usa a estratégia de construir sublistas e aplicar o Algoritmo NF em cada sublista.

Seja L_1 o conjunto de itens de L com comprimento maior que $\frac{1}{2}$, $L_2 \leftarrow \{s \in L : \frac{1}{3} < x(s) \leq \frac{1}{2}\}$ e $L_3 \leftarrow \{s \in L : x(s) \leq \frac{1}{3}\}$. Note que L_1, L_2 e L_3 formam uma partição de L. É fácil ver que o Algoritmo NF empacota um item de L_1 por barra. Como mais do que um item de L_1 não pode ser empacotado em uma mesma barra, temos que

$$\mathrm{NF}(L_1) = \mathrm{OPT}(L_1) \leq \mathrm{OPT}(L). \tag{2.2}$$

Por outro lado, como a garantia de comprimento do empacotamento de L_1, é $\frac{1}{2}$, temos que

$$\mathrm{NF}(L_1) \leq \frac{1}{1/2} x(L_1). \tag{2.3}$$

Para a lista L_2, note que o Algoritmo NF empacota dois itens de L_2 por barra, exceto talvez na última barra. Assim, concluímos que (de forma análoga à feita na última seção)

$$\mathrm{NF}(L_2) \leq \frac{1}{2/3} x(L_2) + 1. \tag{2.4}$$

Para a lista L_3, note que o Algoritmo NF deixa de empacotar um item s de L_3 na barra corrente B somente se $x(s) + x(B) > 1$. Como $x(s) \leq \frac{1}{3}$ temos que $x(B) \geq \frac{2}{3}$ e portanto para esta lista também vale que

$$\mathrm{NF}(L_3) \leq \frac{1}{2/3} x(L_3) + 1. \tag{2.5}$$

Seja $\mathcal{P}_i$ o empacotamento da sublista L_i gerado pela aplicação do Algoritmo NF sobre esta sublista, para $i = 1, 2, 3$. Considere $\mathcal{P}$ a concatenação destes três empacotamentos e defina $h_1 := \#(\mathcal{P}_1)$ e $h_2 := \#(\mathcal{P}_2) + \#(\mathcal{P}_3) - 2$. Das inequações (2.2) e (2.3) temos que

$$\mathrm{OPT}(L) \geq h_1 \tag{2.6}$$

$$h_1 \leq \frac{1}{1/2} x(L_1). \tag{2.7}$$

Das inequações (2.4) e (2.5) temos que

$$\#(\mathcal{P}_2) + \#(\mathcal{P}_3) \leq \frac{1}{2/3} x(L_2 \cup L_3) + 2,$$

e portanto,

$$h_2 \leq \frac{1}{2/3} x(L_2 \cup L_3). \tag{2.8}$$

Das inequações (2.7) e (2.8) e considerando que $\mathrm{OPT}(L) \geq x(L)$, temos que

$$\mathrm{OPT}(L) \geq x(L) = x(L_1) + x(L_2 \cup L_3) \geq \frac{1}{2}h_1 + \frac{2}{3}h_2. \tag{2.9}$$

Das inequações (2.6) e (2.9) temos que,

$$\mathrm{OPT}(L) \geq \max\{h_1, \frac{1}{2}h_1 + \frac{2}{3}h_2\}. \tag{2.10}$$

Por outro lado, temos que $\#(\mathcal{P}) = \#(\mathcal{P}_1) + \#(\mathcal{P}_2) + \#(\mathcal{P}_3) = (h_1 + h_2) + 2$. Logo,

$$\#(\mathcal{P}) \leq \frac{h_1 + h_2}{\mathrm{OPT}(L)} \cdot \mathrm{OPT}(L) + 2.$$

Usando a inequação (2.10), obtemos

$$\#(\mathcal{P}) \leq \alpha \cdot \mathrm{OPT}(L) + 2,$$

onde $\alpha = \frac{h_1+h_2}{\max\{h_1, \frac{1}{2}h_1+\frac{2}{3}h_2\}}$.

Agora, vamos mostrar que $\alpha \leq 1{,}75$ analisando os dois casos onde o máximo ocorre no denominador de α.

Caso (a): $\max\{h_1, \frac{1}{2}h_1 + \frac{2}{3}h_2\} = h_1$.

Neste caso temos, $h_2 \leq \frac{3}{4}h_1$, e portanto

$$\begin{aligned} \frac{h_1 + h_2}{\max\left\{h_1, \frac{1}{2}h_1 + \frac{2}{3}h_2\right\}} &= \frac{h_1 + h_2}{h_1} \\ &\leq \frac{h_1 + \frac{3}{4}h_1}{h_1} \\ &= \frac{7}{4}. \end{aligned}$$

Caso (b): $\max\{h_1, \frac{1}{2}h_1 + \frac{2}{3}h_2\} = \frac{1}{2}h_1 + \frac{2}{3}h_2$.

Neste caso, temos $h_1 \leq \frac{4}{3}h_2$. Note também que $f(h_1, h_2) := \frac{h_1+h_2}{\frac{1}{2}h_1+\frac{2}{3}h_2}$ é uma função estritamente crescente de h_1, e portanto quando $h_1 = \frac{4}{3}h_2$ a função $f(h_1, h_2)$ atinge seu valor máximo. Assim,

$$\begin{aligned} \frac{h_1 + h_2}{\max\left\{h_1, \frac{1}{2}h_1 + \frac{2}{3}h_2\right\}} &= \frac{h_1 + h_2}{\frac{1}{2}h_1 + \frac{2}{3}h_2} \\ &\leq \frac{\frac{4}{3}h_2 + h_2}{\frac{1}{2}\frac{4}{3}h_2 + \frac{2}{3}h_2} \\ &= \frac{7}{4}. \end{aligned}$$

A partir dos dois casos analisados, temos que a seguinte desigualdade é válida:

$$\#(\mathcal{P}) \leq 1{,}75 \cdot \mathrm{OPT}(L) + 2.$$

Note que este algoritmo pode ser implementado de forma a ficar *on-line*. Basta manter como correntes as últimas barras criadas para cada uma das sublistas. Sempre que um item da lista L_i, $i = 1, 2, 3$, deve ser empacotado, o algoritmo tenta empacotar na barra corrente relativa aos itens da lista L_i. Se necessário, usa uma nova barra para empacotar este item, sendo que esta barra se torna a barra corrente para a correspondente lista. Note que este é o Algoritmo H_3 desenvolvido por Lee e Lee [37].

Este mesmo algoritmo pode ser generalizado para o empacotamento de itens pequenos. Para este tipo de instância, use o Algoritmo H_{m+2}. Este algoritmo divide a lista L em sublistas L_1, L_2 e L_3 da seguinte maneira.

$$\begin{aligned} L_1 &\leftarrow L \cap \mathcal{X}[\frac{1}{m+1}, \frac{1}{m}], \\ L_2 &\leftarrow L \cap \mathcal{X}[\frac{1}{m+2}, \frac{1}{m+1}], \\ L_3 &\leftarrow L \cap \mathcal{X}[0, \frac{1}{m+2}]. \end{aligned}$$

Fazendo a mesma análise do empacotamento gerado por este algoritmo, podemos obter as seguintes inequações:

$$\begin{aligned} \mathrm{OPT}(L) &\geq h_1, \\ \mathrm{OPT}(L) &\geq x(L) \\ &= x(L_1) + x(L_2 \cup L_3) \\ &\geq \frac{m}{m+1} h_1 + \frac{m+1}{m+2} h_2. \end{aligned}$$

Note que podemos empacotar no máximo m itens de L_1 por barra.

Assim, obtemos

$$\#(\mathcal{P}) \leq \alpha_m \cdot \mathrm{OPT}(L) + 2,$$

onde $\alpha_m = \frac{h_1+h_2}{\max\{h_1, \frac{m}{m+1}h_1 + \frac{m+1}{m+2}\}}$. Analisando os dois casos onde o máximo ocorre no denominador de α_m, obtemos que $\alpha_m \leq \frac{m+2}{m+1} + \frac{1}{(m+1)^2}$. O Algoritmo H_M pode obter limites de desempenho melhores se considerados valores bem grandes de M. Em [21], Galambos apresenta limites inferiores para os limites de desempenho assintótico de algoritmos *on-line* que empacotam itens pequenos. Na tabela seguinte, apresentamos estes limites inferiores e os valores de $\alpha(H_{m+2})$ e $\alpha(H_M)$, com valores suficientemene grandes de M, para m entre 1 e 4.

Na próxima subseção, utilizamos uma outra estratégia para mostrar que o algoritmo pode ser modificado de forma a ter um melhor limite de desempenho assintótico.

m	Limite inferior	$\alpha(H_{m+2})$	$\lim_{M\to\infty}\alpha(H_M)$
1	1,536	1,750	1,691
2	1,365	1,445	1,423
3	1,274	1,313	1,302
4	1,219	1,240	1,233

Tabela 2.1: Valores de $\alpha(H_{m+2})$, $\alpha(H_M)$ e limites inferiores para o limite de desempenho assintótico dos algoritmos *on-line* para empacotamento de itens pequenos.

2.5.3 Combinando Sublistas com Comprimento Crítico

O algoritmo descrito na subseção anterior, Algoritmo H_3, pode ser visto sob o seguinte aspecto. Basicamente, ele faz uma subdivisão da lista L em duas partes. Uma parte correspondente à lista L_1, para a qual o empacotamento gerado é ótimo e que tem garantia de comprimento $\frac{1}{2}$ e uma outra parte correspondente à lista $L_2 \cup L_3$, para a qual foi obtida uma (melhor) garantia de comprimento $\frac{2}{3}$.

Agora, considerando que podemos identificar em $L_2 \cup L_3$ os itens que induzem pouca garantia de comprimento, uma idéia natural seria combiná-los em grupos de forma a melhorar a garantia de comprimento desses grupos. Vejamos um exemplo.

Considere a seguinte subdivisão de L em sublistas.

$$\begin{aligned} L'_i &\leftarrow \{s \in L : \frac{1}{i+1} < x(s) \leq \frac{1}{i}\}, \quad i = 1, \ldots, 5, \\ L'_6 &\leftarrow \{s \in L : x(s) \leq \frac{1}{6}\}. \end{aligned}$$

Lembramos que na subseção anterior, obtivemos duas garantias de comprimento, uma de $\frac{1}{2}$ em L_1 e outra de $\frac{2}{3}$ em $L_2 \cup L_3$. Aliás, cabe aqui notar que à medida que as sublistas são constituídas de itens cada vez menores, a garantia de comprimento dos empacotamentos produzidos pelo Algoritmo NF vai melhorando (aumenta). Por exemplo, considere as sublistas $L'_3, \ldots, L'_6$. Para estas sublistas, é possível obter as seguintes inequações.

$$\begin{aligned} \mathrm{NF}(L'_i) &\leq \frac{1}{i/(i+1)} x(L'_i) + 1, \quad i = 3, \ldots, 5, \\ \mathrm{NF}(L'_6) &\leq \frac{1}{5/6} x(L'_6) + 1. \end{aligned}$$

Ou seja, temos uma garantia melhor que $\frac{2}{3}$ para as correspondentes sublistas. A pouca garantia está associada às sublistas L'_1 e L'_2 e conseqüentemente a lista L'_2 faz com que o conjunto $L'_2 \cup \ldots \cup L'_6$ tenha também pouca garantia de comprimento. Se melhorarmos a garantia de

comprimento em uma delas, já é possível melhorar o limite de desempenho assintótico feito na subseção anterior. Note que podem existir empacotamentos dos itens de L'_1 que têm garantia de comprimento tão próxima de $\frac{1}{2}$ quanto se queira. E também podem existir empacotamentos de itens de L'_2, com garantia de comprimento tão próxima de $\frac{2}{3}$ quanto se queira. Vamos chamar o conjunto de itens destas sublistas que estão associados à garantia de comprimento próximo de $\frac{1}{2}$ e próximo de $\frac{2}{3}$ de *conjuntos críticos*. Para os demais problemas (e para cada algoritmo) definimos outros conjuntos críticos, cada um apropriado para cada situação. Estes conjuntos estão relacionados aos itens com pouca garantia de espaço *(comprimento, área, volume)*. Chamamos os itens dos conjuntos críticos de *itens críticos*.

O que fazemos é um empacotamento que combina itens críticos destas duas listas (L'_1 e $L'_2 \cup \ldots \cup L'_5$). Este empacotamento terá uma garantia de comprimento de pelo menos $\frac{2}{3}$ e conterá todos os itens críticos de uma das sublistas. Vamos definir como L_A e L_B os conjuntos críticos das listas L'_1 e $L'_2 \cup \ldots \cup L'_5$, respectivamente, da seguinte maneira.

$$\begin{aligned} L_A &\leftarrow \{s \in L'_1 : x(s) \leq 1 - p\}, \\ L_B &\leftarrow \{s \in L'_2 \cup \ldots \cup L'_5 : x(s) \leq p\}, \end{aligned}$$

onde $p = 0{,}408248$. Este valor atribuído à p ficará claro mais adiante.

Seja $\mathcal{P}_{AB}$ um empacotamento parcial de $L_A \cup L_B$ tal que cada barra deste empacotamento contém exatamente dois itens, um de L_A e outro de L_B, e além disso ou todos os itens de L_A ou todos os itens de L_B são empacotados em $\mathcal{P}_{AB}$. Analisando a garantia de comprimento em $\mathcal{P}_{AB}$, obtemos que cada barra B de $\mathcal{P}_{AB}$ é tal que $x(B) > \frac{1}{2} + \frac{1}{6} = \frac{2}{3}$. E portanto, aqui também vale que

$$\#(\mathcal{P}_{AB}) \leq \frac{1}{2/3} x(L_{AB}),$$

onde L_{AB} são os itens em $\mathcal{P}_{AB}$.

Seja $L''_i \leftarrow L'_i \setminus L_{AB}$. Aplique o Algoritmo NF sobre estas sublistas gerando os empacotamentos $\mathcal{P}_i$, $i = 1, \ldots, 6$ da seguinte maneira.

$$\mathcal{P}_i \leftarrow \mathrm{NF}(L''_i), \quad i = 1, \ldots, 6.$$

Considere $\mathcal{P}_{aux}$ como sendo a concatenação dos empacotamentos $\mathcal{P}_2, \ldots, \mathcal{P}_6$ (*i.e.*, $\mathcal{P}_{aux} \leftarrow \mathcal{P}_2 \| \ldots \| \mathcal{P}_6$); $\mathcal{P}_{opt}$ a concatenação dos empacotamentos $\mathcal{P}_1$ e $\mathcal{P}_{AB}$ e seja $\mathcal{P}$ a concatenação dos empacotamentos $\mathcal{P}_{opt}$ e $\mathcal{P}_{aux}$. Assim, $\mathcal{P}$ é um empacotamento total de L. Agora, vamos analisar o desempenho assintótico do algoritmo.

Primeiramente, considere o empacotamento $\mathcal{P}_{AB}$. Há dois casos a considerar. Se todos os itens de L_A ou todos os de L_B são empacotados em $\mathcal{P}_{AB}$.

Caso 1. Todos os itens de L_A são empacotados em $\mathcal{P}_{AB}$.

Neste caso temos uma melhora na garantia de comprimento para os itens de L_1''. Assim, obtemos

$$\#(\mathcal{P}_1) \leq \frac{1}{1-p} x(L_1''). \tag{2.11}$$

Como $1 - p \leq \frac{2}{3}$, temos que

$$\#(\mathcal{P}_{opt}) = \#(\mathcal{P}_1 \| \mathcal{P}_{AB}) \leq \frac{1}{1-p} x(L_1 \cup L_{AB}).$$

Para a lista $L_2'' \cup \ldots \cup L_6''$ não foi possível melhorar, mantendo a mesma garantia de $\frac{2}{3}$ do comprimento, i.e.,

$$\#(\mathcal{P}_i) \leq \frac{1}{2/3} x(L_i'') + 1, \quad i = 2, \ldots, 6.$$

Usando as inequações obtidas para os empacotamentos $\mathcal{P}_2, \ldots, \mathcal{P}_6$ obtemos

$$\#(\mathcal{P}_{aux}) \leq \frac{1}{2/3} x(L_{aux}) + 5, \tag{2.12}$$

onde L_{aux} é o conjunto de itens empacotados em $\mathcal{P}_{aux}$.

Considerando (2.11) e (2.12), podemos fazer as mesmas análises feitas anteriormente, obtendo

$$\#(\mathcal{P}) = \#(\mathcal{P}_{opt}) + \#(\mathcal{P}_{aux}) \leq \alpha_1'' \cdot \mathrm{OPT}(L) + 5,$$

onde $\alpha_1'' = \frac{h_1 + h_2}{\max\{h_1, (1-p)h_1 + \frac{2}{3}h_2\}}$.

Calculando um limite superior para α chegamos à seguinte inequação:

$$\#(\mathcal{P}) \leq 1{,}613 \cdot \mathrm{OPT}(L) + 5.$$

Caso 2. Todos os itens de L_B são empacotados em $\mathcal{P}_{AB}$.

Neste caso a inequação para $\mathcal{P}_{opt}$ permanece

$$\#(\mathcal{P}_{opt}) \leq \frac{1}{1/2} x(L_1'').$$

E para o empacotamento $\mathcal{P}_2$ temos uma melhora para

$$\#(\mathcal{P}_2) \leq \frac{1}{2p} x(L_2'') + 1,$$

já que podemos empacotar dois itens de L_2'' em cada barra, cada item com tamanho pelo menos p.

Como todos os itens em $L_3 \cup \ldots \cup L_5$ são empacotados em L_{AB} e $2p \leq \frac{5}{6}$, temos a seguinte inequação para $\mathcal{P}_{aux}$

$$\#(\mathcal{P}_{aux}) \leq \frac{1}{2p} x(L_{aux}) + 2.$$

Procedendo da mesma forma que no Caso 1, obtemos

$$\#(\mathcal{P}) = \#(\mathcal{P}_{opt}) + \#(\mathcal{P}_{aux}) \leq \alpha_2'' \cdot \mathrm{OPT}(L) + 3,$$

onde $\alpha_2'' = \frac{h_1+h_2}{\max\{h_1, \frac{1}{2}h_1+2ph_2\}} \leq 1{,}613$.

Juntando os casos 1 e 2, concluímos que o algoritmo descrito acima para gerar o empacotamento $\mathcal{P}$ tem limite de desempenho assintótico 1,613. O parâmetro p, tomado como sendo $p = 0{,}408248$, foi calculado de modo que os dois casos acima resultassem no mesmo limite.

Note que este método para combinar conjuntos críticos faz com que o algoritmo tenha um limite de desempenho assintótico melhor, mas este deixa de ser *on-line*. Esta técnica será muito usada posteriormente nos problemas de empacotamento bi- e tridimensional. Veremos que, muitas vezes a definição de sublistas críticas envolverá conjuntos mais complexos e será crucial para a melhora de alguns limites de desempenho assintótico.

Muitas vezes, a busca do valor de p, usado para definir os conjuntos críticos, requer muitos cálculos, principalmente quando o valor de p envolve literais. Nestes casos, usamos o software *Maple V* da Waterloo Maple Software para fazer tais cálculos.

2.6 Resumo dos Algoritmos para o PEU

Algoritmo	**Tipo**	α	β	**Ref.**	**Condição**
FF	*on-line*	$\frac{m+1}{m}$	2	[33]	Itens pequenos
H_M	*on-line*	$\alpha(\mathrm{H}_M)$	$\mathcal{O}(M)$	[37]	Itens pequenos
FFD	*off-line*	$\frac{m+3}{m+2} - \frac{1}{m(m+1)(m+2)}$	4	[13]	Itens pequenos
FFD,BFD	*off-line*	1,5	—	[55]	Caso Geral
FFD	*off-line*	1,222 ...	1	[30]	Caso Geral
VL_ϵ	*off-line*	$1+\epsilon$	$\left(\frac{1}{\epsilon}\right)^2$	[16]	Caso Geral
KK	*off-line*	1	$\mathcal{O}\left(\frac{log^2 \mathrm{OPT}(L)}{\mathrm{OPT}(L)}\right)$	[35]	Caso Geral
H_M	*on-line*	$\asymp 1{,}691\ldots$	$M-1$	[37]	Caso Geral
Harmonic + 1	*on-line*	$\asymp 1{,}5888\ldots$	$\mathcal{O}(M)$	[52]	Caso Geral

Problema de Empacotamento em Faixa

3.1 Introdução

O Problema de Empacotamento em Faixa (PEF) foi primeiramente proposto por Baker, Coffman e Rivest [3] em 1980. Neste problema o objetivo é empacotar uma lista L de retângulos em um retângulo R de largura a e altura infinita, de modo a minimizar a altura do empacotamento. Os retângulos devem ser empacotados ortogonalmente e com orientação fixa.

Uma aplicação interessante deste problema ocorre em projetos de escalonamento de processos. A altura de um item corresponde ao tempo de processamento requerido pelo item e sua largura corresponde à quantidade de memória contínua requerida pelo processo. O valor a (largura do retângulo R) é a quantidade de memória disponível. Assim, um empacotamento $\mathcal{P}$, de uma lista de processos L, fornece uma maneira de se escalonar esses processos, sendo que a altura do empacotamento $\mathcal{P}$ corresponde ao tempo para executar todos esses processos.

Em 1980, Coffman, Garey, Johnson e Tarjan [9], desenvolveram os algoritmos chamados NFDH$^{(f)}$ *(Next Fit Decreasing Height)* e FFDH$^{(f)}$ *(First Fit Decreasing Height)*, e provaram que o primeiro tem limite de desempenho assintótico 2 e o segundo tem limite de desempenho assintótico 1,7. Quando consideramos instâncias com características especiais o Algoritmo FFDH$^{(f)}$ apresenta melhor desempenho assintótico. Se todos os itens a serem empacotados são quadrados, esses autores mostraram que o Algoritmo FFDH$^{(f)}$ tem limite de desempenho assintótico 1,5. Já no caso em que todos os retângulos de L não têm largura maior que $\frac{1}{m}$ da largura da faixa, provaram que o Algoritmo FFDH$^{(f)}$ tem limite de desempenho assintótico $\frac{m+1}{m}$. Neste mesmo artigo, esses autores apresentam o algoritmo com melhor limite de desempenho assintótico, de $\frac{m+2}{m+1}$, conhecido para o caso onde os itens têm largura pequena.

Para instâncias quaisquer, o algoritmo com melhor limite de desempenho assintótico con-

hecido é devido a Baker, Brown e Katseff [2]. Este algoritmo é chamado de UD *(Up and Down)* e tem limite de desempenho assintótico $\frac{5}{4}$. Observamos que o empacotamento gerado por este algoritmo pode não ser guilhotinável. Assim, todos os algoritmos que fizerem uso deste algoritmo em particular, não poderão ser considerados guilhotináveis também.

No caso de algoritmos *on-line*, Csirik e Woeginger [15] apresentam um algoritmo que gera empacotamentos divididos em níveis com limite de desempenho assintótico que pode se tornar tão próximo de 1,691... quanto se queira. Eles mostram também que o limite inferior para o limite de desempenho assintótico de um algoritmo *on-line* para o PEF, que gera o empacotamento dividido em níveis, deve ser pelo menos 1,691.... Em [5], Brown, Baker e Katseff mostraram que o limite de desempenho absoluto de qualquer algoritmo *on-line* para o PEF não pode ser menor do que 2. Recentemente, Schiermeyer [54] e Steinberg [57] apresentaram algoritmos *off-line* com limite de desempenho absoluto 2 para o PEF.

Nas seções subseqüentes descrevemos alguns dos algoritmos que mencionamos e apresentamos os resultados mais significativos a respeito desses algoritmos. Também apresentamos um algoritmo *off-line* para o caso com rotações, com limite de desempenho assintótico não maior que 1,613 e outro algoritmo *on-line* com limite de desempenho assintótico não maior que 1,75.

3.2 Definições

Sem perda de generalidade, neste capítulo vamos trabalhar no plano xy.

Um empacotamento bidimensional orientado de uma lista de retângulos $L = (r_1, \ldots, r_n)$ em um retângulo $R = (a, b)$ é uma função $\mathcal{P} : L \to [0, a) \times [0, b)$, tal que

$$\mathcal{P}^x(r_i) + x(r_i) \leq a \quad \text{e} \quad \mathcal{P}^y(r_i) + y(r_i) \leq b,$$

onde $\mathcal{P}(r_i) := (\mathcal{P}^x(r_i), \mathcal{P}^y(r_i))$, $i = 1, \ldots, n$.

Mais ainda, se $\mathcal{R}(r_i)$ é definido como

$$\mathcal{R}(r_i) := [\mathcal{P}^x(r_i), \mathcal{P}^x(r_i) + x(r_i)) \times [\mathcal{P}^y(r_i), \mathcal{P}^y(r_i) + y(r_i)),$$

então

$$\mathcal{R}(r_i) \cap \mathcal{R}(r_j) = \emptyset \quad \text{para todo} \quad i, j, \ 1 \leq i \neq j \leq n.$$

Dado um empacotamento bidimensional orientado $\mathcal{P}$ de L em R, denotamos por $H(\mathcal{P})$ a *altura* do empacotamento $\mathcal{P}$, *i.e.*, $H(\mathcal{P}) := \max\{\mathcal{P}^y(r) + y(r) : \ r \in L\}$.

Um *nível* em um empacotamento $\mathcal{P}$ para o PEF é definido como uma região

$$N = [0, a) \times [Y_1, Y_2),$$

na qual há um conjunto S de retângulos tais que

$$\text{para todo } r \in S : \mathcal{P}^y(r) = Y_1 \text{ e } Y_2 - Y_1 \geq \max_{r \in S}(y(r)).$$

Quando a altura do nível não é previamente definida pelo algoritmo, consideramos que $Y_2 - Y_1 = \max_{r \in S}(y(r))$.

A Figura 3.1 ilustra um empacotamento dividido em níveis para o problema de empacotamento em faixa.

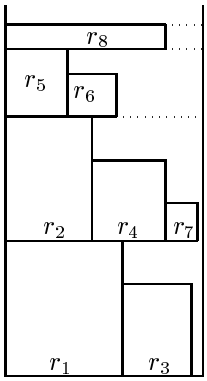


Figura 3.1: Exemplo de empacotamento em níveis para o PEF

Dado um valor p, $0 < p < 1$, definimos um *conjunto de arredondamento* como o conjunto $\mathcal{S} := \{1, p^1, p^2, \ldots, p^i, \ldots\}$. Este nome vem do fato de considerarmos uma dimensão dos itens arredondadas para um valor em $\mathcal{S}$. Com este conjunto de arredondamento, definimos uma *i-faixa* como um nível de altura p^i. Uma i-faixa sempre estará associada com um conjunto de arredondamento.

Problema. *Problema de Empacotamento em Faixa (PEF): Dados uma lista de retângulos $L = (r_1, \ldots, r_n)$ e um retângulo (faixa) $R = (a, \infty)$, encontrar um empacotamento bidimensional orientado de L em R, cuja altura seja mínima.*

A seguir, definimos o Problema de Empacotamento em Faixa com rotações ortogonais.

Problema. *Problema de Empacotamento em Faixa com Rotação (PEF^r): Dados uma lista de retângulos $L = (r_1, \ldots, r_n)$ e um retângulo (faixa) $R = (a, \infty)$, encontrar um empacotamento bidimensional orientado de uma lista $L' \in \Gamma(L)$ em R, cuja altura seja mínima.*
Lembramos que uma lista em $\Gamma(L)$ contém possíveis orientações para os retângulos de L. Assim, o objetivo deste problema é encontrar uma orientação (para os retângulos de L) e um

empacotamento destes na faixa de modo a minimizar a altura do empacotamento.

Denotaremos por PEF(a) (PEF$^r(a)$) o PEF (PEF^r) cuja faixa de entrada seja $R = (a, \infty)$.

3.3 Caso Orientado

Nesta seção, apresentamos os algoritmos NFDH$^{(f)}$, FFDH$^{(f)}$ e UD, bem como seus limites de desempenho assintóticos. Apresentamos um resultado sobre garantia de área, de forma equivalente ao feito para a garantia de comprimento para o problema de empacotamento unidimensional.

3.3.1 Algoritmo NFDH$^{(f)}$

O empacotamento gerado pelo Algoritmo NFDH$^{(f)}$ é constituído de níveis, sendo que todos os retângulos de um nível são empacotados com um lado sobre a linha (inferior) que define cada nível. Primeiramente o Algoritmo NFDH$^{(f)}$ ordena a lista $L = (r_1, \ldots, r_n)$, de forma que $y(r_1) \geq y(r_2) \geq \cdots \geq y(r_n)$. Feito isto, começa a empacotar os retângulos de L segundo essa ordenação. Para empacotar um novo retângulo r, o algoritmo tenta empacotá-lo no último nível criado, dispondo-o mais à esquerda. Caso não consiga, cria um novo nível no topo do empacotamento em construção e empacota r neste novo nível. Neste caso, r é empacotado mais à esquerda, sendo que a altura deste nível é definida pela altura de r. (Note que os itens estão ordenados em ordem não-crescente de altura).

O seguinte resultado foi provado por Coffman *et al* [9].

Teorema 3.3.1. *Para qualquer instância L do PEF, onde nenhum retângulo tem altura superior a Z, tem-se que*

$$\text{NFDH}^{(f)}(L) \leq 2 \cdot \text{OPT}(L) + Z.$$

A seguir, apresentamos um resultado que relaciona a garantia de largura ocupada em cada nível com seu limite de desempenho assintótico.

Lema 3.3.2. *Seja $\mathcal{P}$ um empacotamento de uma lista L de retângulos gerado pelo Algoritmo* NFDH$^{(f)}$ *para o* PEF(a) (PEF$^r(a)$). *Sejam $N_1, \ldots, N_v$ os níveis criados, nesta ordem e $x(N_i)$ a soma das larguras dos retângulos em N_i. Se $x(N_i) \geq l \cdot a$ para $i = 1, \ldots, v-1$, então*

$$\text{NFDH}^{(f)}(L) \leq \frac{1}{l}\frac{S(L)}{a} + Z.$$

Prova. Seja h_i a altura de cada nível N_i. Com isto temos que

$$S(N_i) \geq l \cdot a \cdot h_{i+1}, \quad \text{para} \quad i = 1, \ldots, v-1.$$

Somando todas as inequações para $i = 1, \ldots, v-1$ temos

$$\begin{aligned} S(L) &> \sum_{i=1}^{v-1} S(N_i) \\ &\geq l \cdot a \cdot \sum_{i=1}^{v-1} h_{i+1} \\ &= l \cdot a \cdot (H(\mathcal{P}) - h_1). \end{aligned}$$

Como $h_1 \leq Z$, temos que

$$H(\mathcal{P}) \leq \frac{1}{l}\frac{S(L)}{a} + Z. \tag{3.1}$$

□ Chamaremos o valor l na inequação 3.1 de *garantia de largura* do empacotamento $\mathcal{P}$.

Vamos ilustrar o uso deste lema no seguinte algoritmo para empacotamento de itens de largura pequena. Vamos chamá-lo de SSP_m.

Algoritmo $\text{SSP}_m(L)$

Entrada: Lista de retângulos $L \subset \mathcal{X}[0, \frac{1}{m}]$ para o $\text{PEF}(a)$.

Saída: Empacotamento $\mathcal{P}$ de L em $R = (a, \infty)$.

1 Particione a lista L em duas sublistas L_1 e L_2, tais que

$$\begin{aligned} L_1 &\leftarrow L \cap \mathcal{X}[\frac{1}{m+1}, \frac{1}{m}], \\ L_2 &\leftarrow L \cap \mathcal{X}[0, \frac{1}{m+1}]. \end{aligned}$$

2 $\mathcal{P}_i \leftarrow \text{NFDH}^{(f)}(L_i)$, $i = 1, 2$;

3 $\mathcal{P} \leftarrow \mathcal{P}_1 \| \mathcal{P}_2$.

4 Retorne $\mathcal{P}$.

Fim algoritmo.

Teorema 3.3.3. *Para qualquer instância $L \subset \mathcal{X}[0, \frac{1}{m}]$ do PEF, onde nenhum retângulo tem altura superior a Z, tem-se que*

$$\text{SSP}_m(L) \leq \left(\frac{m+1}{m}\right) \text{OPT}(L) + 2Z.$$

Prova. Primeiramente observe que em cada um dos empacotamentos $\mathcal{P}_1$ e $\mathcal{P}_2$, as faixas geradas têm uma largura ocupada pelos retângulos de pelo menos $\frac{m}{m+1}$, exceto talvez na última faixa de cada um deles. Assim, usando o Lema 3.3.2, temos

$$H(\mathcal{P}_i) \leq \frac{m+1}{m} S(L_i) + Z.$$

Portanto,

$$\begin{aligned} H(\mathcal{P}) &\leq \left(\frac{m+1}{m}\right) S(L) + 2Z \\ &\leq \left(\frac{m+1}{m}\right) \mathrm{OPT}(L) + 2Z. \end{aligned}$$

□

3.3.2 Algoritmo $\mathrm{FFDH}^{(f)}$

O Algoritmo $\mathrm{FFDH}^{(f)}$ é muito semelhante ao Algoritmo $\mathrm{NFDH}^{(f)}$. Na verdade pode-se dizer que é uma versão melhorada do Algoritmo $\mathrm{NFDH}^{(f)}$. O Algoritmo $\mathrm{NFDH}^{(f)}$ sempre que cria um novo nível, nunca mais tenta empacotar um retângulo nos níveis anteriores. Já o Algoritmo $\mathrm{FFDH}^{(f)}$, sempre que vai empacotar um novo retângulo r, tenta empacotá-lo em algum nível anteriormente criado, colocando-o no primeiro nível em que isso for possível, justificando-o sempre mais à esquerda. Caso não seja possível colocar r em nenhum dos níveis criados, então $\mathrm{FFDH}^{(f)}$ cria um novo nível como no Algoritmo $\mathrm{NFDH}^{(f)}$.

O seguinte resultado foi provado por Coffman *et al.* [9].

Teorema 3.3.4. *Para qualquer instância L do PEF, onde nenhum retângulo tem altura superior a Z, tem-se que*

$$\mathrm{FFDH}^{(f)}(L) \leq 1{,}7 \cdot \mathrm{OPT}(L) + Z.$$

3.3.3 Algoritmo UD

O Algoritmo UD foi desenvolvido por Baker, Brown e Katseff [2] e tem o melhor limite de desempenho assintótico conhecido para o PEF. Estes autores provaram o seguinte resultado.

Teorema 3.3.5. *Para qualquer lista L de retângulos de altura no máximo Z,*

$$\mathrm{UD}(L) \leq \frac{5}{4}\mathrm{OPT}(L) + \frac{53}{8}Z.$$

O Algoritmo UD é relativamente sofisticado, não sendo possível descrevê-lo em menos de 2 páginas. Apesar de se tratar de um algoritmo que será utilizado em alguns dos algoritmos que desenvolvemos, optamos por não incluí-lo, já que para fazer a análise dos algoritmos que desenvolvemos basta conhecer seu limite de desempenho assintótico. O leitor interessado pode encontrar sua descrição no artigo de Baker *et al* [2].

3.3.4 Empacotamento *on-line* de itens pequenos

Nesta seção apresentamos dois algoritmos *on-line* para o empacotamento de itens pequenos para o PEF. Um deles tem um limite de desempenho assintótico que pode se tornar tão próximo de $\frac{m+2}{m+1} + \frac{1}{(m+1)^2}$ quanto se queira.

A técnica usada aqui é muito comum no desenvolvimento de algoritmos *on-line* e também será usada em muitos algoritmos *on-line* apresentados nesta tese. Dado $0 < p < 1$, seja $\mathcal{S} := \{p^i : i \in \mathbb{Z}, \quad i \geq 0\}$ um conjunto de arredondamento. O conjunto de arredondamento é usado para arredondar (para cima) a altura dos retângulos de modo a termos retângulos com alturas em $\mathcal{S}$. Fazendo este arredondamento podemos considerar os retângulos com mesma altura, digamos p^i, para serem empacotados em níveis, também de altura p^i (i-faixas). Com isso, basta considerar o empacotamento de retângulos em faixas como sendo um empacotamento unidimensional *on-line*.

A seguir, apresentamos a descrição do Algoritmo $\text{OSSP}_{m,p}(L)$.

Algoritmo $\text{OSSP}_{m,p}(L)$.

Entrada: Lista de retângulos $L \subset \mathcal{X}[0, \frac{1}{m}]$ para o $\text{PEF}^r(a)$.

Saída: Empacotamento $\mathcal{P}$ de L em $R = (a, \infty)$.

1 Empacote os retângulos $r \in L$ na ordem em que estes aparecem em L.

- **1.1** Seja i tal que $p^{i+1} < \frac{y(r)}{Z} \leq p^i$.
- **1.2** Se $\frac{a}{m+1} < x(r) \leq \frac{a}{m}$ então empacote r nas i-faixas geradas para estes retângulos. Para isto, use o Algoritmo NF para empacotar o retângulo r em um dos níveis. Caso necessário, empacote r em um novo nível de altura p^i.
- **1.3** Se $0 < x(r) \leq \frac{a}{m+1}$ então empacote r de forma análoga a feita no passo 1.3, mas empacote r nos níveis gerados para os retângulos com $0 < x(r) \leq \frac{a}{m+1}$.

2 Retorne o empacotamento gerado.

Fim algoritmo.

Teorema 3.3.6. *Para toda lista de entrada $L \subset \mathcal{X}[0, \frac{1}{m}]$ para o $\text{PEF}(a)$, onde os retângulos de L têm altura menores ou iguais a Z, e dado um valor p, $0 < p < 1$, temos que*

$$\text{OSSP}_{m,p}(L) \leq \frac{1}{p}\left(\frac{m+1}{m}\right)\text{OPT}(L) + \frac{2Z}{1-p}.$$

Prova. Seja $L_1 := L \cap \mathcal{X}[\frac{1}{m+1}, \frac{1}{m}]$ e $L_2 := L \cap \mathcal{X}[0, \frac{1}{m+1}]$.

Seja $\mathcal{F}_k^i$ o conjunto dos níveis com altura p^i geradas para os retângulos de L_k; $n_k^i = |\mathcal{F}_k^i|$; h_k^i a soma das alturas dos níveis em $\mathcal{F}_k^i$ e $h_k := \sum_{i \geq 0} h_k^i$, para $k = 1, 2$.

Como a ocupação, no eixo x, pelos retângulos em cada nível de $\mathcal{F}_1^i$, exceto talvez no último, é pelo menos $\frac{m}{m+1}$, temos que

$$\begin{aligned}\frac{S(L_1)}{a} &= \sum_{i\geq 0}\frac{S(L_1^i)}{a} \\ &\geq \sum_{i\geq 0} p^{i+1}\cdot(n_1^i-1)\cdot\left(\frac{m}{m+1}\right)\cdot Z \\ &\geq p\cdot\left(\frac{m}{m+1}\right)\left(h_1-\frac{1}{1-p}Z\right). \qquad (3.2)\end{aligned}$$

Resultado análogo pode ser obtido para a lista L_2. Assim,

$$\frac{S(L_2)}{a}\geq p\cdot\left(\frac{m}{m+1}\right)\left(h_2-\frac{1}{1-p}Z\right). \qquad (3.3)$$

De (3.2) e (3.3) obtemos

$$\frac{S(L)}{a}\geq p\cdot\left(\frac{m}{m+1}\right)\left(\mathrm{OSSP}_{m,p}(L)-\frac{2}{1-p}Z\right). \qquad (3.4)$$

Isolando $\mathrm{OSSP}_{m,p}(L)$, temos

$$\begin{aligned}\mathrm{OSSP}_{m,p}(L) &\leq \frac{1}{p}\left(\frac{m+1}{m}\right)\frac{S(L)}{a}+\frac{2Z}{1-p} \\ &\leq \frac{1}{p}\left(\frac{m+1}{m}\right)\mathrm{OPT}(L)+\frac{2Z}{1-p}.\end{aligned}$$

□

O Algoritmo $\mathrm{OSSP}_{m,p}$ pode ser refinado de forma a obtermos um melhor limite de desempenho assintótico. Para isso, dividimos a lista L em duas sublistas L_1 e L_2, $L = L_1 \cup L_2$, e aplicamos o Algoritmo $\mathrm{OSSP}_{m,p}$ sobre a lista L_1 e o Algoritmo $\mathrm{OSSP}_{m+1,p}$ sobre a lista L_2. Por fim, retornamos a concatenação destes empacotamentos. Denominamos este algoritmo de $\mathrm{IOSSP}_{m,p}$.

Teorema 3.3.7. *Para toda lista de entrada $L \subset \mathcal{X}[0,\frac{1}{m}]$ para o* PEF(1)*, onde os retângulos de L têm altura menores ou iguais a Z, e dado um valor p, $0<p<1$, temos que*

$$\mathrm{IOSSP}_{m,p}(L)\leq \alpha_m\cdot\mathrm{OPT}(L)+\frac{4Z}{1-p},$$

onde $\lim_{p\to 1}\alpha_m = \frac{m+2}{m+1}+\frac{1}{(m+1)^2}$.

Prova. Vamos analisar o limite de desempenho assintótico deste algoritmo. Seja $\mathcal{P}_i$ o empacotamento gerado para a lista L_i, $i=1,2$. Podemos verificar que o empacotamento $\mathcal{P}_1$ pode ser tomado assintóticamente tão próximo do ótimo quanto se queira, bastando para isso, colocar p

próximo de 1. Considere os níveis do empacotamento $\mathcal{P}_1$, de altura $p^i \cdot Z$. Todos os níveis deste tipo têm m retângulos, exceto talvez o último. Mais ainda, não podemos empacotar mais que m retângulos de L_1 lado a lado. Como cada retângulo destes níveis tem altura maior que $p^{i+1} \cdot Z$, temos que

$$\mathrm{OPT}(L_1) \geq \sum_{i \geq 0} p \cdot \left(h'_i - p^i \cdot Z\right),$$

onde h'_i é a altura total dos níveis de L_1 com altura p^i. A subtração de $p^i \cdot Z$ de h'_i foi para desconsiderar o último nível da soma de h'_i. Portanto, temos que

$$H(\mathcal{P}_1) = \sum_{i \geq 0} h'_i \leq \frac{1}{p}\mathrm{OPT}(L) + \frac{Z}{1-p}.$$

Considerando as garantias de largura dos empacotamentos $\mathcal{P}_1$ e $\mathcal{P}_2$ obtemos as seguintes inequações usando o Teorema 3.3.6.

$$\begin{aligned} H(\mathcal{P}_1) &\leq \frac{1}{p}\left(\frac{m+1}{m}\right) S(L_1) + \frac{2Z}{1-p}, \\ H(\mathcal{P}_2) &\leq \frac{1}{p}\left(\frac{m+2}{m+1}\right) S(L_2) + \frac{2Z}{1-p}. \end{aligned}$$

Seja $h_i := H(\mathcal{P}_i) - \frac{2Z}{1-p}$, $i = 1, 2$. Com isto, temos

$$\begin{aligned} \mathrm{OPT}(L) &\geq p \cdot h_1, \\ \mathrm{OPT}(L) &\geq S(L) \\ &= S(L_1) + S(L_2) \\ &\geq p \cdot \left(\frac{m}{m+1}\right) \cdot h_1 + p \cdot \left(\frac{m+1}{m+2}\right) \cdot h_2. \end{aligned}$$

Ou seja, $\mathrm{OPT}(L) \geq \max\{p \cdot h_1, p \cdot \left(\frac{m}{m+1}\right) \cdot h_1 + p \cdot \left(\frac{m+1}{m+2}\right) \cdot h_2\}$. Pelos mesmos argumentos usados anteriormente, temos

$$\mathrm{IOSSP}_{m,p}(L) \leq \alpha_m \cdot \mathrm{OPT}(L) + \frac{4Z}{1-p},$$

onde $\lim_{p \to 1} \alpha_m = \frac{m+2}{m+1} + \frac{1}{(m+1)^2}$. □

3.4 Caso com Rotações

Como já observado nas *Preliminares*, os algoritmos desenvolvidos para o PEF podem ser usados para gerar empacotamentos do PEF^r. Mas em geral, os limites de desempenho já não são válidos neste caso.

Em [7], Coffman *et al.* observam a validade do limite de desempenho assintótico do Algoritmo NFDH$^{(f)}$ para o PEF, já que a análise deste se baseia apenas em argumentos de área. De fato, o Lema 3.3.2 é válido também para o PEF^r. Assim, o Algoritmo SSP$_m$ também tem limite de desempenho assintótico de $\frac{m+1}{m}$. Observe que para isso os retângulos dever ser dados com orientação válida para serem empacotados.

A seguir, apresentamos um algoritmo com limite de desempenho não maior que 1,613 para o PEF^r, melhorando assim o limite anterior de 2.

3.4.1 Algoritmo SPR

Descrevemos nesta seção o Algoritmo SPR *(Strip Packing algorithm using Rotations)* para o PEF$^r(a)$. Este algoritmo se assemelha muito ao algoritmo mencionado para o problema unidimensional, que combina listas críticas. Vale notar que também a demonstração de seu limite de desempenho é similar.

Primeiramente, vamos descrever um algoritmo chamado COLUMN$^{(f)}$. Este algoritmo constrói duas colunas, onde cada coluna é uma pilha de retângulos, colocados um em cima do outro (veja a Figura 3.2).

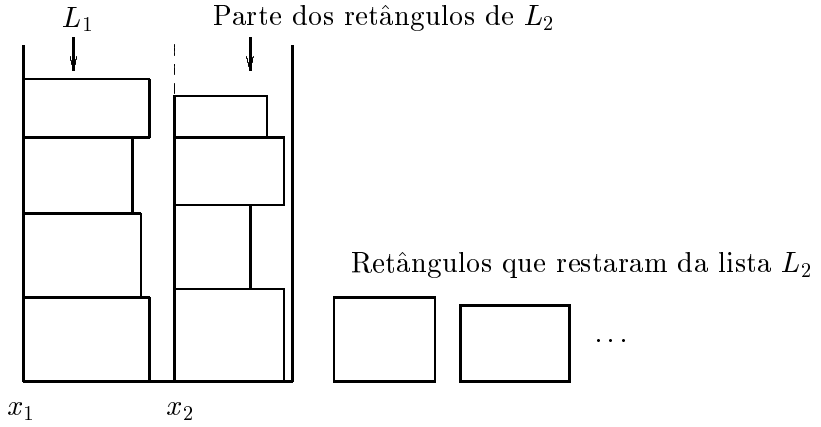


Figura 3.2: Empacotamento gerado pelo Algoritmo COLUMN$^{(f)}$

O Algoritmo COLUMN$^{(f)}$ é chamado com parâmetros (L_1, L_2, x_1, x_2), onde L_1 e L_2 são duas sublistas e x_1 e x_2 são posições onde as pilhas estarão alinhadas à esquerda, desde o fundo do empacotamento. Cada coluna conterá apenas os retângulos de uma das listas, e colunas distintas corresponderão a listas distintas.

Vamos chamar de *altura da coluna* a soma das alturas de todos os retângulos dessa coluna.

Para empacotar um retângulo, o algoritmo escolhe primeiramente uma das colunas que tem

a menor altura. Seja h a altura desta coluna e L_i a lista associada a esta coluna. Caso nem todos os retângulos dessa lista tenham sido empacotados, o algoritmo empacota o próximo retângulo de L_i, digamos r, na posição (x_i, h). Em seguida, a lista L_i é atualizada, removendo-se o retângulo r. Este processo é repetido até que todos os retângulos de uma das listas tenham sido totalmente empacotados (ou seja, $L_1 = \emptyset$ ou $L_2 = \emptyset$). Ao final, o algoritmo retorna um par $(\mathcal{P}', L')$, onde $\mathcal{P}'$ é o empacotamento construído e L' é o conjunto de retângulos empacotados em $\mathcal{P}'$. Neste ponto, vamos assumir que as posições x_1 e x_2 e as listas L_1 e L_2 são escolhidas de modo que elas não possam gerar empacotamentos inviáveis.

O seguinte lema é válido para este algoritmo.

Lema 3.4.1. *Seja $\mathcal{P}$ um empacotamento de $L' \subseteq L_1 \cup L_2$ gerado pelo Algoritmo COLUMN$^{(f)}$ quando aplicado às listas L_1 e L_2 para o* PEF$^r(a)$. *Se $x(r) > l_i \cdot a$ para todo retângulo $r \in L_i$, $i = 1, 2$, e nenhum retângulo de L tem altura maior que Z, então $H(\mathcal{P}) \leq \frac{1}{l_1+l_2}\frac{S(L')}{a} + Z$.*

Prova. Considere as duas colunas do empacotamento $\mathcal{P}$. Note que cada uma das colunas tem altura pelo menos $(H(\mathcal{P}) - Z)$. Como a largura de cada retângulo de L_i é pelo menos $l_i \cdot a$, então a área do empacotamento é pelo menos $S(L) \geq (l_1 + l_2) \cdot a \cdot (H(\mathcal{P}) - Z)$. Portanto, $H(\mathcal{P}) \leq \frac{1}{l_1+l_2}\frac{S(L')}{a} + Z$. □

A idéia deste algoritmo é fazer com que o empacotamento final fique dividido em duas partes, como feito para o empacotamento unidimensional na seção 2.5.3. Neste empacotamento, a parte associada ao empacotamento parcial ótimo será feito para retângulos com largura maior que $\frac{a}{2}$. Note que em um empacotamento orientado, se L_1 contém apenas retângulos deste tipo, então um empacotamento ótimo pode ser construído empilhando-se os retângulos, um em cima do outro. Mas quando permitimos rotações, devemos primeiro re-orientar todos os retângulos que podem *sair* deste conjunto L_1. Desta forma, ficam apenas os retângulos que ou não podem ser re-orientados, ou se podem, ainda continuam no conjunto L_1. Assim, introduzimos um novo passo que consiste em re-orientar os retângulos de L_1 de forma que cada retângulo fique com a menor altura possível, mantendo uma orientação viável. Desta forma, conseguimos um empacotamento ótimo de L_1, apenas colocando um retângulo sobre o outro. O restante do algoritmo se assemelha ao algoritmo unidimensional da seção 2.5.3, diferindo pelo fato de as inequações envolverem área em vez de comprimento.

Algoritmo SPR(L).

Entrada: Lista de retângulos L para o PEF$^r(a)$

Saída: Empacotamento $\mathcal{P}$ de L em $R = (a, \infty)$.

1 Gire todos os retângulos r com $x(r) > \frac{a}{2}$ e $y(r) \leq \frac{a}{2}$.

2 Gire todos os retângulos com $x(r) > \frac{a}{2}$ e $x(r) < y(r) \leq a$.

3 Seja $p \leftarrow 0{,}40824$.

$$\begin{aligned} L'_i &\leftarrow \{s \in L : \frac{a}{i+1} < x(s) \leq \frac{a}{i}\}, \quad i = 1, \ldots, 4, \\ L'_5 &\leftarrow \{s \in L : (1-2p)a < x(s) \leq \frac{a}{5}\}, \\ L'_6 &\leftarrow \{s \in L : x(s) \leq (1-2p) \cdot a\}, \\ L_A &\leftarrow \{r \in L'_1 : x(r) \leq (1-p) \cdot a\}, \\ L_B &\leftarrow \{r \in L'_2 \cup \ldots \cup L'_5 : x(r) \leq p \cdot a\}. \end{aligned}$$

4 $(\mathcal{P}_{AB}, L_{AB}) \leftarrow \text{COLUMN}^{(f)}(L_A, L_B, 0, 1-p)$;

$L_i \leftarrow L'_i \setminus L_{AB}$, para $i = 1, \ldots, 6$;

5 $\mathcal{P}_i \leftarrow \text{NFDH}^{(f)}(L_i)$, $i = 1, \ldots, 6$;

6 $\mathcal{P}_{opt} \leftarrow \mathcal{P}_1 \| \mathcal{P}_{AB}$;

7 $\mathcal{P}_{aux} \leftarrow \mathcal{P}_2 \| \ldots \| \mathcal{P}_6$.

8 Retorne $\mathcal{P}_{opt} \| \mathcal{P}_{aux}$.

Fim algoritmo.

Teorema 3.4.2. *Para toda lista de entrada L para o $\text{PEF}^r(a)$, onde os retângulos de L têm dimensões menores ou iguais a Z, temos que*

$$\text{SPR}(L) \leq 1{,}613 \cdot \text{OPT}(L) + 6Z.$$

Prova. Vamos calcular a garantia de largura dos empacotamentos gerados ao empacotar as sublistas $L_1, \ldots, L_6, L_{AB}$. Na lista L_{AB} temos que a largura de cada retângulo de L_A é pelo menos $\frac{1}{2}$ e a largura de cada retângulo de L_B é pelo menos $\frac{1}{6}$. Assim, pelo Lema 3.4.1 temos que

$$H(\mathcal{P}_{AB}) \leq \frac{1}{1/2 + 1/6} \frac{S(L_{AB})}{a} + Z$$

e portanto, vale também que

$$H(\mathcal{P}_{AB}) \leq \frac{1}{1-p} \frac{S(L_{AB})}{a} + Z. \tag{3.5}$$

Seja L_{opt} o conjunto de retângulos empacotados em $\mathcal{P}_{opt}$. É fácil ver que $\mathcal{P}_{opt}$ é um empacotamento assintótico ótimo de L_{opt}, pois os retângulos *grandes* (com $x(r) > \frac{1}{2}$) de $L_1 \cup \ldots \cup L_{AB}$ ou não podem ser girados, ou se podem, estes recaem no conjunto definido para L_1. Mais ainda, os retângulos grandes de L_{opt} estão empacotados de forma que sua altura é a menor possível. Portanto, temos

$$H(\mathcal{P}_{opt}) \leq \text{OPT}(L) + Z. \tag{3.6}$$

Agora vamos dividir a demonstração em duas partes, da mesma forma que fizemos para o problema unidimensional.

Caso 1. $L_A \subseteq L_{AB}$.

Neste caso, temos que todos os retângulos de $L_1 := L'_1 \setminus L_{AB}$ têm largura maior que $(1-p)\cdot a$. Usando (3.5), obtemos $S(L_{opt}) \geq (H(\mathcal{P}_{opt}) - Z)\cdot(1-p)\cdot a$, i.e.,

$$H(\mathcal{P}_{opt}) \leq \frac{1}{1-p}\frac{S(L_{opt})}{a} + Z. \tag{3.7}$$

Para cada lista L_i, $(i = 2, \ldots, 6)$ temos que a garantia de largura em cada nível de $\mathcal{P}_i$, exceto talvez no último, é pelo menos $\frac{2}{3}$. Assim, pelo Lema 3.3.2 temos que

$$H(\mathcal{P}_i) \leq \frac{1}{2/3}\frac{S(L_i)}{a} + Z, \quad i = 2, \ldots, 6. \tag{3.8}$$

Como $\mathcal{P}_{aux} = \mathcal{P}_2 \| \ldots \| \mathcal{P}_6$, temos que

$$H(\mathcal{P}_{aux}) \leq \frac{1}{2/3}\frac{S(L_{aux})}{a} + 5Z. \tag{3.9}$$

Como

$$\mathrm{OPT}(L) \geq \frac{S(L)}{a} \geq \frac{S(L_{opt})}{a} + \frac{S(L_{aux})}{a} \geq (1-p)\cdot h_1 + \frac{2}{3}\cdot h_2, \tag{3.10}$$

onde $h_1 := H(\mathcal{P}_{opt}) - Z$ e $h_2 := H(\mathcal{P}_{aux}) - 5Z$. De (3.10) e (3.6) temos que

$$\mathrm{OPT}(L) \geq \max\{h_1, (1-p)\cdot h_1 + \frac{2}{3}\cdot h_2\},$$

e portanto obtemos que

$$H(\mathcal{P}) \leq \alpha_1 \cdot \mathrm{OPT}(L) + 6Z,$$

onde $\alpha_1 = \frac{h_1+h_2}{\max\{h_1,(1-p)\cdot h_1+\frac{2}{3}\cdot h_2\}} \leq \frac{2+3p}{2}$.

Caso 2. $L_B \subseteq L_{AB}$.

A análise deste caso usa dos mesmos argumentos usados no caso 1 e é análoga a prova do caso 2 feita para o algoritmo da Seção 2.5.3. Assim, apresentamos apenas as inequações que podem ser obtidas:

$$\begin{aligned} H(\mathcal{P}_{opt}) &\leq \frac{1}{1/2}\frac{S(L_{opt})}{a} + Z, \\ H(\mathcal{P}_{aux}) &\leq \frac{1}{2p}\frac{S(L_{aux})}{a} + 4Z, \\ \mathrm{OPT}(L) &\geq \max\{h_1, \frac{1}{2}h_1 + 2ph_2\}. \end{aligned}$$

Juntando essas inequações e procedendo como no caso anterior, temos que

$$H(\mathcal{P}) \leq \alpha_2 \cdot \mathrm{OPT}(L) + 6Z,$$

onde $\alpha_2 = \frac{h_1+h_2}{\max\{h_1, \frac{1}{2}h_1+2ph_2\}} \leq \frac{4p+1}{4p}$.

Tomando se p tal que $\frac{2+3p}{2} = \frac{4p+1}{4p}$, ou seja, $p = \frac{1}{\sqrt{6}}$, temos que $\alpha_1 \leq \frac{2+3p}{2} = 1{,}6123\ldots$ e $\alpha_2 \leq \frac{4p+1}{4p} = 1{,}6123\ldots$. Logo, $H(\mathcal{P}) \leq 1{,}613 \cdot \mathrm{OPT}(L) + 6Z$. □

Proposição 3.4.3. *O limite de desempenho assintótico do Algoritmo SPR mostrado no Teorema 3.4.2 é justo.*

Prova. Considere a seguinte instância L do $\mathrm{PEF}^r(1)$, $L = L_1 \| L_2$, onde

$$L_1 = (r'_1, \ldots, r'_{n_1}) \quad \text{sendo} \quad r'_i = \left(\frac{1}{2} + \epsilon, Z\right)$$

$$L_2 = (r''_1, \ldots, r''_{n_2}) \quad \text{sendo} \quad r''_i = \begin{cases} (1-2p, Z - \xi \cdot i) & \text{se } i \bmod M = 1 \quad \text{e} \\ \left(\frac{1}{k}, Z - \xi \cdot i\right) & \text{c.c.} \end{cases}$$

onde M é o menor inteiro tal que $(M-1)\frac{1}{k} + (1-2p) > 2p$ e ϵ e ξ são valores apropriados bem pequenos e $Z > 1$.

Essa instância L foi construída de modo que o empacotamento ótimo de L tenha n_1 níveis, onde cada um destes níveis contém exatamente um retângulo de L_1 e o restante do nível contém retângulos de L_2 (veja a Figura 3.3 (c) e (d)), ficando o nível praticamente preenchido.

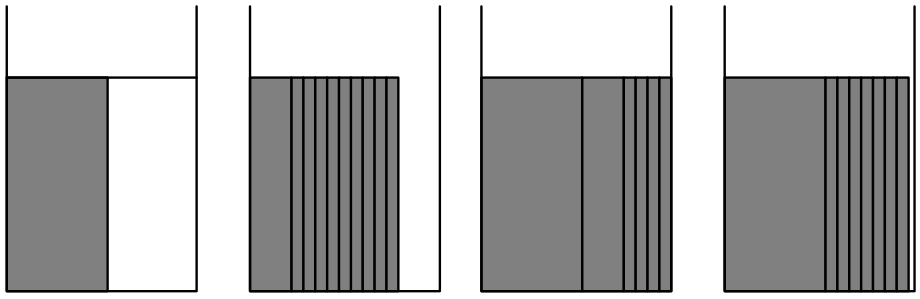


Figura 3.3: Tipos de níveis no empacotamento gerado pelo Algoritmo SPR e um empacotamento ótimo.

Aplicando o algoritmo, obtemos um empacotamento $\mathcal{P}_1$ com altura $H(\mathcal{P}_1) = n_1 \cdot Z$ (onde os níveis são como os apresentados na Figura 3.3 (a)) e um empacotamento $\mathcal{P}_2$ com altura próxima de $\frac{n_1}{4p}Z$ (onde os níveis são como os apresentados na Figura 3.3 (b)). Portanto, $H(\mathcal{P}) \approx \left(1 + \frac{1}{4p}\right) n_1 \cdot Z$. Como o empacotamento ótimo tem altura $n_1 \cdot Z$, tomando-se n_1 bem grande, podemos obter $\frac{H(\mathcal{P})}{\mathrm{OPT}(L)}$ tão próximo de $1{,}6123\ldots$ quanto se queira. □

3.4.2 Algoritmo OSPR$_p$

Nesta seção apresentamos um algoritmo *on-line* para o caso geral do PEF^r. Este algoritmo tem limite de desempenho assintótico que pode se tornar tão próximo de 1,75 quanto se queira. O algoritmo também usa um conjunto de arredondamento para *arredondar* a altura dos retângulos.

Algoritmo OSPR$_p(L)$.

Entrada: Lista de retângulos L para o PEF$^r(a)$.

Saída: Empacotamento $\mathcal{P}$ de L em $R = (a, \infty)$.

1 Seja $\mathcal{P}$ um empacotamento vazio.

2 Empacote os retângulos $r \in L$ na ordem em que estes aparecem em L.

2.1 Se $\left(x(r) > \frac{a}{2} \quad \text{e} \quad y(r) > \frac{a}{2}\right)$ então

2.1.1 Seja $r' \in \{r, \rho(r)\}$ tal que $y(r')$ seja o mínimo possível.

2.1.2 Empacote r' em um novo nível de altura $y(r')$ no topo de $\mathcal{P}$.

2.2 Senão

2.2.1 Seja $r' \in \{r, \rho(r)\}$ tal que $x(r')$ seja mínimo.

2.2.2 Use o Algoritmo OSSP$_{2,p}$ para empacotar o retângulo r'.

3 Retorne $\mathcal{P}$.

Fim algoritmo.

Teorema 3.4.4. *Para toda lista de entrada L para o* PEF$^r(a)$*, onde os retângulos de L têm dimensões menores ou iguais a Z, temos que*

$$\text{OSPR}_p(L) \leq \alpha_p \cdot \text{OPT}(L) + \frac{2Z}{1-p},$$

onde $\lim_{p \to 1} \alpha_p \leq 1{,}75$.

Prova. Seja L_1 o conjunto de retângulos tal que $\left(x(r) > \frac{a}{2} \quad \text{e} \quad y(r) > \frac{a}{2}\right)$ e $L_2 := L \setminus L_1$.

Considere que todos os retângulos $r \in L_1$ têm altura mínima possível. Seja $h_1 := y(L_1)$.

Como não podemos empacotar dois retângulos de L_1 lado a lado, temos que

$$\begin{aligned} \text{OPT}(L) &\geq \text{OPT}(L_1) \\ &= h_1. \end{aligned} \tag{3.11}$$

Como todos os retângulos da lista L_1 têm largura maior que $\frac{a}{2}$, temos

$$S(L_1) \geq h_1 \cdot \frac{a}{2}. \tag{3.12}$$

Como a lista L_2, é empacotada pelo Algoritmo $\mathrm{OSSP}_{2,p}$, usando (3.4), temos

$$S(L_2) \geq p \cdot \frac{2}{3}\left(\mathrm{OSSP}_{2,p}(L) - \frac{2}{1-p}Z\right). \tag{3.13}$$

Definindo h_2 como $h_2 := \left(\mathrm{OSSP}_{2,p}(L) - \frac{2}{1-p}Z\right)$ temos que

$$\begin{aligned} \mathrm{OPT}(L) &\geq \frac{S(L)}{a} \\ &\geq \frac{S(L_1)}{a} + \frac{S(L_2)}{a} \\ &\geq \frac{1}{2}h_1 + \frac{2}{3}p \cdot h_2. \end{aligned} \tag{3.14}$$

De (3.11) e (3.14), obtemos

$$\mathrm{OPT}(L) \geq \max\{h_1, \frac{1}{2}h_1 + \frac{2}{3}p \cdot h_2\}.$$

Procedendo como feito anteriormente e usando o fato que $\mathrm{OSPR}_p(L) = h_1 + h_2 + \frac{2}{1-p}Z$, temos

$$\mathrm{OSPR}_p(L) \leq \alpha_p \cdot \mathrm{OPT}(L) + \frac{2Z}{1-p},$$

onde $\alpha_p = \frac{h_1+h_2}{\max\{h_1, \frac{1}{2}h_1+\frac{2}{3}p\cdot h_2\}}$. Assim, $\lim_{p\to 1}\alpha_p \leq 1{,}75$. □

3.5 Resumo dos Algoritmos

A seguir apresentamos um resumo dos algoritmos, para o problema de empacotamento em faixa, nas duas tabelas seguintes.

Os algoritmos desenvolvidos nesta tese estão marcados com $\star$ na coluna da referência.

Algoritmos para o PEF

Algoritmo	**Tipo**	α	β	**Ref.**	**Condição**
M,S	*off-line*	2	—	[54, 57]	Caso Geral
FFDH	*off-line*	$\left(\frac{m+1}{m}\right)$	$2Z$	[9]	Retângulos de largura pequena
SF	*off-line*	$\left(\frac{m+2}{m+1}\right)$	$2Z$	[9]	Retângulos de largura pequena
$\mathrm{IOSSP}_{m,p}$	*on-line*	$\asymp \frac{m+2}{m+1} + \frac{1}{(m+1)^2}$	$\frac{4Z}{1-p}$	$\star$	Retângulos de largura pequena
FFDH	*off-line*	1,7	Z	[9]	Caso Geral
UD	*off-line*	1,25	$\frac{53}{8}Z$	[2]	Caso Geral
$\mathrm{Shelf}(H_M, p)$	*on-line*	$\asymp 1{,}691$	$\frac{M}{1-p}$	[15]	Caso Geral

Algoritmos para o PEF^r

Algoritmo	**Tipo**	α	β	**Ref.**	**Condição**
SSP_m	*off-line*	$\frac{m+1}{m}$	2Z	$\star$	Retângulos de largura pequena
OSSP_m	*on-line*	$\asymp \frac{m+1}{m}$	$\frac{2Z}{1-p}$	$\star$	Retângulos de largura pequena
$\mathrm{NFDH}^{(f)}$	*off-line*	2	Z	[9, 7]	Caso Geral
BLDW	*off-line*	2	Z	[3, 7]	Caso Geral
SPR	*off-line*	1,6123...	$4Z$	$\star$	Caso Geral
OSPR	*on-line*	$\asymp 1{,}75$	$\frac{2Z}{1-p}$	$\star$	Caso Geral

Problema de Empacotamento em Placas

4.1 Introdução

Em 1982 foi apresentado o primeiro algoritmo com estudo de limite de desempenho assintótico para o Problema de Empacotamento em Placas, PEP, com orientação fixa. Este algoritmo, devido a Chung, Garey e Johnson [6], tem limite de desempenho assintótico não maior que 2,125, sendo o melhor limite conhecido até o momento.

Para o caso *on-line*, o primeiro algoritmo de aproximação que foi desenvolvido para o mesmo problema é devido a Coppersmith e Raghavan [12], com limite de desempenho assintótico não maior que 3,25. Os algoritmos com o melhor limite de desempenho assintótico conhecido são devidos a Li e Cheng [39] e Csirik e van Vliet [14], ambos com limite de desempenho assintótico igual a 2,86.

Por outro lado, Blitz, van Vliet e Woeginger [4] mostraram que qualquer algoritmo *on-line* para o PEP deve ter um limite de desempenho assintótico de pelo menos 1,907.

Apresentamos uma demonstração simples de que este problema não pode ter limite de desempenho absoluto menor que 2, a menos que $\mathcal{P} = \mathcal{NP}$.

Descrevemos algoritmos *on-line* e *off-line* para o empacotamento de itens de dimensões pequenas para o PEP com limite de desempenho $\left(\frac{m+1}{m}\right)^2$, além de uma versão melhorada para o caso *off-line*. Apresentamos também algoritmos quando a instância é restrita à quadrados.

Quando permitimos rotações ortogonais, PEP^r, descrevemos um algoritmo para o caso geral cujo limite de desempenho assintótico pode chegar tão próximo de 2,639... quanto se queira; para o caso *on-line*, apresentamos um algoritmo com um limite de desempenho assintótico 3,25 e no caso particular onde as placas são quadradas, um algoritmo com um limite de desempenho

assintótico que pode se tornar tão próximo de 2,6875 quanto se queira.

Além disso, apresentamos algoritmos, *on-line* e *off-line*, para empacotamento de retângulos de dimensões pequenas.

4.2 Definições

Sem perda de generalidade, estaremos trabalhando no plano xy.

Problema. *Problema de Empacotamento em Placas (PEP): Dados uma lista de retângulos* $L = (r_1, \ldots, r_n)$ *e retângulos (ou placas)* $R = (a, b)$*, achar uma partição* $L_1, \ldots, L_k$*, de* L *e empacotamentos bidimensionais orientados de cada lista* L_i *em placa* $R = (a, b)$ *de forma a minimizar* k*.*

Problema. *Problema de Empacotamento em Placas com Rotação (PEP*^{r}*): Dados uma lista de retângulos* $L = (r_1, \ldots, r_n)$ *e retângulos (ou placas)* $R = (a, b)$*, achar uma partição* $L_1, \ldots, L_k$*, de* L *e empacotamentos bidimensionais orientados de uma lista* $L'_i \in \Gamma(L_i)$ *em placa* $R = (a, b)$ *de forma a minimizar* k*.*

A Figura 4.1 ilustra um empacotamento de retângulos em placas.

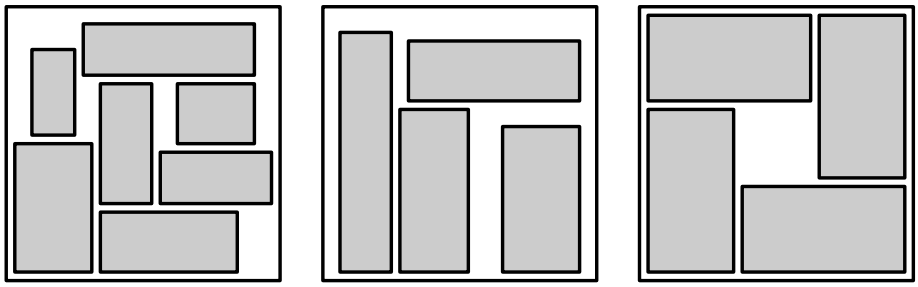


Figura 4.1: Exemplo de empacotamento de retângulos em placas.

Chamaremos de *comprimento* e *largura* de um retângulo r os valores $x(r)$ e $y(r)$, respectivamente.

Denotaremos por PEP(a, b) o problema de empacotamento em placas quando as placas R são da forma $R = (a, b)$.

Um empacotamento $\mathcal{P}$ de L em R, além de dar a coordenada na qual cada retângulo r foi empacotado, também deve dar a placa em que r foi empacotado. Denotaremos por $\mathcal{P}^p(r)$ a placa

onde r foi empacotada e por $(\mathcal{P}^x(r), \mathcal{P}^y(r))$ a coordenada na placa $\mathcal{P}^p(r)$, onde r foi empacotada.

Uma *faixa em uma placa* é uma região

$$F = [0, a) \times [Y_1, Y_2)$$

na qual há um conjunto S de retângulos tais que para todo $r', r'' \in S$, $\mathcal{P}^p(r') = \mathcal{P}^p(r'')$ e para todo $r \in S : \mathcal{P}^y(r) = Y_1$ e $Y_2 - Y_1 \geq \max_{r \in S}(y(r))$.

Quando a largura da faixa não for definida previamente pelo algoritmo, quando do seu uso, consideramos que $Y_2 - Y_1 = \max_{r \in S}(y(r))$. Dizemos que esta faixa foi gerada ortogonalmente ao eixo y e paralelamente ao eixo x. Faixas na direção do eixo y são igualmente possíveis.

Da mesma forma que a *garantia de comprimento* em barras é importante no empacotamento unidimensional, a *garantia de área* também é importante no problema de empacotamento em placas, e também será usado na análise de limites de desempenho. O seguinte lema ilustra como a garantia de área em placas pode ser usada para se analisar limites de desempenho de algoritmos.

Lema 4.2.1. *Suponha que exista um algoritmo $\mathcal{A}$ para o PEP(a, b) de forma que dada uma lista L de retângulos, o algoritmo $\mathcal{A}$ garante que a área ocupada pelos retângulos nas placas é pelo menos $s \cdot a \cdot b$ em cada placa, exceto em no máximo um número constante C delas. Então vale que*

$$\mathcal{A}(L) \leq \frac{1}{s}\frac{S(L)}{a \cdot b} + C \leq \frac{1}{s}\mathrm{OPT}(L) + C. \tag{4.1}$$

O valor s, acima, será chamado de *garantia de área* do empacotamento gerado pelo algoritmo $\mathcal{A}$ sobre a lista L.

A inequação (4.1) apresentada no Lema 4.2.1 será muito usada. Ela relaciona a ocupação de área s garantida por um algoritmo, com a qualidade do empacotamento gerado.

4.3 Aproximabilidade do Problema de Empacotamento em Placas

Nesta seção, vamos mostrar que o PEP (PEP^r) não é aproximável dentro de $2 - \epsilon$, $\epsilon > 0$, em relação ao limite de desempenho absoluto. O tipo de demonstração usado não é novo, tendo sido já usado para provar a não aproximabilidade absoluta do PEU dentro de $\frac{3}{2} - \epsilon$ [25] e do PET dentro de $(2 - \epsilon)$ [41].

O seguinte resultado, devido a Leung, Tam, Wong, Young e Chin [38], será usado no próximo teorema.

Lema 4.3.1. *Dados uma lista de quadrados L e um quadrado Q, decidir se L é empacotável em Q é um problema $\mathcal{NP}$-completo.*

Teorema 4.3.2. *O Problema de Empacotamento em Placas não é aproximável dentro de $2-\epsilon$, para qualquer $\epsilon > 0$, a menos que $\mathcal{P} = \mathcal{NP}$. Este resultado é válido mesmo que todos itens de entrada sejam quadrados.*

Prova. Suponha que exista um algoritmo polinomial $\mathcal{A}$ para o PEP tal que

$$\mathcal{A}(L) < 2 \cdot \mathrm{OPT}(L) \quad \text{para toda lista } L \text{ de entrada.} \tag{4.2}$$

Vamos mostrar que neste caso existe um algoritmo polinomial para decidir se uma lista de quadrados L pode ser empacotada em um quadrado Q. Chamemos de φ este último problema.

Considere uma lista de quadrados L e um quadrado Q, uma instância do problema φ. Se existisse um algoritmo polinomial $\mathcal{A}$ com $\mathcal{A}(L) < 2 \cdot \mathrm{OPT}(L)$, teríamos duas possibilidades:

(i) $\mathcal{A}(L) < 2$. Neste caso, temos uma resposta positiva para o problema φ.

(ii) $\mathcal{A}(L) \geq 2$. Neste caso, a resposta para o problema φ é negativa.

□

4.4 Caso Orientado

Nesta seção apresentamos algoritmos para o empacotamento de retângulos pequenos (*on-line* e *off-line*) e para o empacotamento de quadrados em quadrados. Apresentamos também alguns algoritmos conhecidos que serão usados como subrotinas, a saber os algoritmos $\mathrm{NF}^{(p)}$ e $\mathrm{NFDH}^{(p)}$. Além destes, apresentamos o Algoritmo HFF que ilustra como usar dois algoritmos como subrotinas, para problemas mais restritos, para fazer um algoritmo para o problema de empacotamento em placas.

No fim desta seção apresentamos a definição de dois conjuntos críticos importantes. Os conjuntos $\mathcal{A}^k_{k,i}$ e $\mathcal{B}^k_{k,i}$. Estes conjuntos serão bastante usados posteriormente.

4.4.1 Algoritmo $\mathrm{NF}^{(p)}$

O Algoritmo $\mathrm{NF}^{(p)}$ *(Next Fit)* gera empacotamentos por faixas e empacota cada retângulo na ordem em que este aparece em L. Para empacotar um retângulo r, o Algoritmo $\mathrm{NF}^{(p)}$ testa na faixa corrente, se o retângulo r pode ser empacotado no fim desta, de forma a ficar ajustado mais à esquerda e sem passar dos limites da placa. Caso este item não possa ser empacotado

desta forma, a faixa corrente F fica com a largura fixada em $\max_{r \in F}(y(r))$ e uma nova faixa é criada, contendo o retângulo r e sem definir sua largura, acima da faixa anterior. Caso isso não seja possível, esta faixa é colocada em uma nova placa. Esta última faixa criada se torna a faixa corrente. Este algoritmo também será denominado de NF^x, por gerar as faixas paralelamente ao eixo x. A versão deste algoritmo que gera faixas paralelamente ao eixo y será denominado de NF^y.

4.4.2 Algoritmo $\mathrm{NFDH}^{(p)}$

O Algoritmo $\mathrm{NFDH}^{(p)}$ *(Next Fit Decreasing Height)* ordena a lista L de forma que seus itens fiquem em ordem não-crescente de largura, i.e., de forma que $L = (r_1, \ldots, r_n)$ com $y(r_1) \geq y(r_2) \geq \cdots \geq y(r_n)$. Depois disto, aplica o Algoritmo NF^x sobre esta lista e retorna o empacotamento gerado.

4.4.3 Algoritmo HFF

O Algoritmo HFF *(Hibrid First Fit)* foi desenvolvido por Chung, Garey e Johnson [6] para o PEP. Trata-se de um algoritmo híbrido formado por dois outros algoritmos de empacotamento: o Algoritmo FFD para o PEU e o Algoritmo $\mathrm{FFDH}^{(f)}$ para o PEF.

Dadas uma lista de retângulos L e placas $R = (a, b)$, o Algoritmo HFF aplica o Algoritmo $\mathrm{FFDH}^{(f)}$, para o $\mathrm{PEF}(a)$, para gerar os níveis $N_1, \ldots, N_k$ na faixa $R = (a, \infty)$. Seja l_i a altura do nível N_i, $i = 1, \ldots, k$. Cada nível N_i é encarado como um item de comprimento l_i de um PEU onde o comprimento da barra é b. Estes níveis são empacotados em barras usando o Algoritmo FFD. A partir deste empacotamento é obtido o empacotamento para o $\mathrm{PEP}(a, b)$.

O Algoritmo HFF possui o melhor limite de desempenho assintótico conhecido para o PEP. O teorema abaixo nos dá este limite, provado em [6].

Teorema 4.4.1. *Para qualquer lista de entrada L do Problema de Empacotamento em Placas, temos que*

$$\mathrm{HFF}(L) < \frac{17}{8}\mathrm{OPT}(L) + 5.$$

4.4.4 Empacotamento de Retângulos Pequenos em Placas

A seguir, apresentamos um algoritmo para empacotar retângulos pequenos em placas. Este algoritmo usa o Algoritmo $\mathrm{NFDH}^{(p)}$ como subrotina, para o qual foi provado o seguinte resultado devido a Li e Cheng [41].

Lema 4.4.2. *Uma lista de retângulos $L = (r_1, \ldots, r_n)$ com $x(r_i) \leq \frac{1}{m}$ e $y(r_i) \leq \frac{1}{m}$ pode ser empacotada em uma placa (1,1) pelo Algoritmo $\mathrm{NFDH}^{(p)}$ do PEP(1,1), se $S(L) \leq \left(1 - \frac{1}{m}\right)^2$.*

A idéia do algoritmo é particionar a lista L em sublistas de forma a garantir uma área de $\left(\frac{m}{m+1}\right)^2$ para algumas listas usando-se o Algoritmo $\mathrm{NF}^{(p)}$. Para os demais itens, o algoritmo divide-os em outras sublistas, cada sublista contendo retângulos com área total não maior que $\left(1-\frac{1}{m'}\right)^2$, $m' = m+2$, e aplica o Algoritmo $\mathrm{NFDH}^{(p)}$ para o PEP para empacotar cada sublista em uma placa (1,1).

Algoritmo $\mathrm{BI_m}$

Entrada: Lista de retângulos $L = (r_1, \ldots, r_n)$ com $x(r_i) \leq \frac{1}{m}$ e $y(r_i) \leq \frac{1}{m}$.

Saída: Empacotamento de L em placas $R = (1,1)$.

1 Divida a lista L em sublistas $L_1, \ldots, L_6$ da seguinte maneira (*cf.* Figura 4.2).

$$
\begin{aligned}
L_1 &\leftarrow L \cap \mathcal{C}^{xy}\left[\tfrac{1}{m+1}, \tfrac{1}{m} \;;\; \tfrac{1}{m+1}, \tfrac{1}{m}\right],\\
L_2 &\leftarrow L \cap \mathcal{C}^{xy}\left[0, \tfrac{1}{m+1} \;;\; \tfrac{1}{m+1}, \tfrac{1}{m}\right],\\
L_3 &\leftarrow L \cap \mathcal{C}^{xy}\left[\tfrac{1}{m+1}, \tfrac{1}{m} \;;\; 0, \tfrac{1}{m+1}\right],\\
L_4 &\leftarrow L \cap \mathcal{C}^{xy}\left[0, \tfrac{1}{m+1} \;;\; \tfrac{1}{m+2}, \tfrac{1}{m+1}\right] \cap \mathcal{C}^{x}_{y}[0,1],\\
L_5 &\leftarrow L \cap \mathcal{C}^{xy}\left[\tfrac{1}{m+2}, \tfrac{1}{m+1} \;;\; 0, \tfrac{1}{m+1}\right] \cap \mathcal{C}^{y}_{x}[0,1],\\
L_6 &\leftarrow L \cap \mathcal{C}^{xy}\left[0, \tfrac{1}{m+2} \;;\; 0, \tfrac{1}{m+2}\right].
\end{aligned}
$$

2 $\mathcal{P}_i \leftarrow \mathrm{NF}^x(L_i), \quad i = 1, 2, 4;$

3 $\mathcal{P}_i \leftarrow \mathrm{NF}^y(L_i), \quad i = 3, 5.$

4 Gere um empacotamento $\mathcal{P}_6$ de L_6 da seguinte maneira.

Subdivida a lista L_6 em sublistas $L_6^1, \ldots, L_6^v$ tal que

$$
\begin{aligned}
S(L_6^i) &\leq \left(\tfrac{m}{m+2}\right) + \left(\tfrac{1}{m+2}\right)^2, \quad \text{para} \quad i = 1, \ldots, v;\\
S(L_6^i) + S(\mathrm{first}(L_6^{i+1})) &> \left(\tfrac{m}{m+2}\right) + \left(\tfrac{1}{m+2}\right)^2, \quad \text{para} \quad i = 1, \ldots, v-1.
\end{aligned}
$$

$\mathcal{P}_6^i \leftarrow \mathrm{NFDH}^{(p)}(L_6^i) \quad$ para $\quad i = 1, \ldots, v;$

$\mathcal{P}_6 \leftarrow \mathcal{P}_6^1 \| \ldots \| \mathcal{P}_6^v.$

5 $\mathcal{P} \leftarrow \mathcal{P}_1 \| \ldots \| \mathcal{P}_6.$

6 Retorne($\mathcal{P}$).

Fim algoritmo.

Lema 4.4.3. *Para toda lista de retângulos $L = (r_1, \ldots, r_n)$, onde $x(r_i) \leq \frac{1}{m}$ e $y(r_i) \leq \frac{1}{m}$, temos que* $\mathrm{BI}_m(L) \leq \left(\frac{m+1}{m}\right)^2 S(L) + 6$.

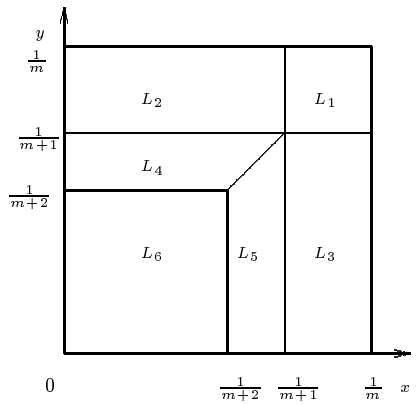


Figura 4.2: Partição da lista L gerada pelos algoritmos BI_m e $\mathrm{OFF}_m^{(2)}$

Prova. Vamos analisar os empacotamentos das sublistas $L_1, \ldots, L_6$. No caso da sublista L_1, o Algoritmo $\mathrm{NFDH}^{(p)}$ empacota pelo menos m^2 retângulos por placa, exceto talvez na última placa de $\mathcal{P}_1$. Como a área de cada retângulo de L_1 é pelo menos $\left(\frac{1}{m+1}\right)^2$, a área ocupada pelos retângulos de L_1 em cada placa de $\mathcal{P}_1$, exceto talvez na última placa, é pelo menos $\left(\frac{m}{m+1}\right)^2$ e portanto $S(L_1) \geq \left(\frac{m}{m+1}\right)^2 (\#(\mathcal{P}_1) - 1)$. Assim, obtemos

$$\#(\mathcal{P}_1) \leq \left(\frac{m+1}{m}\right)^2 S(L_1) + 1. \tag{4.3}$$

No caso da lista L_2, temos que o Algoritmo $\mathrm{NFDH}^{(p)}$ gera m faixas na direção do eixo x, exceto talvez na última placa de $\mathcal{P}_2$. Como a área de ocupação de cada faixa é pelo menos $\frac{1}{m+1}\left(1 - \frac{1}{m+1}\right)$, para cada placa, exceto talvez a última, temos que a área ocupada é pelo menos $m\frac{1}{m+1}\left(1 - \frac{1}{m+1}\right) = \left(\frac{m}{m+1}\right)^2$, e novamente temos

$$\#(\mathcal{P}_2) \leq \left(\frac{m+1}{m}\right)^2 S(L_2) + 1. \tag{4.4}$$

Para as listas L_3, L_4 e L_5, fazendo análise semelhante às anteriores, obtemos

$$\#(\mathcal{P}_i) \leq \left(\frac{m+1}{m}\right)^2 S(L_i) + 1 \quad \text{para} \quad i = 1, \ldots, 5. \tag{4.5}$$

Considerando o empacotamento $\mathcal{P}_6$, pelo Lema 4.4.2 temos uma garantia de área que é pelo menos $\left(\frac{m}{m+2}\right)$, e portanto também vale que

$$\#(\mathcal{P}_6) \leq \left(\frac{m+1}{m}\right)^2 S(L_6) + 1. \tag{4.6}$$

Somando as inequações (4.5) e (4.6), obtemos

$$\#(\mathcal{P}) \leq \left(\frac{m+1}{m}\right)^2 S(L) + 6.$$

□ Usando este lema, obtemos o seguinte teorema.

Teorema 4.4.4. *Para toda lista de retângulos* $L = (r_1, \ldots, r_n)$*, onde* $x(r_i) \leq \frac{1}{m}$ *e* $y(r_i) \leq \frac{1}{m}$*, temos que* $\mathrm{BI}_m(L) \leq \left(\frac{m+1}{m}\right)^2 \mathrm{OPT}(L) + 6$.

Em seguida apresentamos um outro algoritmo para o PEP com melhor limite de desempenho assintótico. O algoritmo usa como subrotina o procedimento COMB, cuja finalidade é combinar os itens críticos de L_1 com os itens críticos de $L_2 \cup \ldots \cup L_6$.

Vamos descrever duas variantes do Algoritmo COMB, a serem chamados de $\mathrm{COMB}^{\mathrm{x}}$ e $\mathrm{COMB}^{\mathrm{y}}$. Este algoritmo é chamado com três parâmetros: (L_A, L', i), onde a lista L_A consiste de retângulos em $\mathcal{C}^{xy}\left[\frac{1}{m+1}, q\ ;\ \frac{1}{m+1}, q\right]$ e L' pode ser qualquer lista em $\{L_B^1, \ldots, L_B^k, L_C^1, \ldots, L_C^k\}$. A lista L_B^i consiste de retângulos em $\mathcal{C}^{xy}\left[\frac{1}{k}, p\ ;\ \frac{1}{m+i}, \frac{1}{m-1+i}\right]$, e a lista L_C^i consiste de retângulos em $\mathcal{C}^{xy}\left[\frac{1}{m+i}, \frac{1}{m-1+i}\ ;\ \frac{1}{k}, p\right]$, onde q, p e k são valores apropriados tais que $\frac{1}{m+1} < q \leq \frac{1}{m}$, $\frac{1}{m+2} < p \leq \frac{1}{m+1}$ e $k \geq m+2$. Com estas três listas dadas como entrada, o Algoritmo COMB empacota $m^2 + m + i - 1$ retângulos em cada placa, exceto talvez em $2k$ placas (para cada conjunto L_B^i e L_C^i): m^2 retângulos de L_A e $m + i - 1$ retângulos de L_{BC}, $L_{BC} := L_B \cup L_C$, $L_B := L_B^1 \cup \ldots \cup L_B^k$ e $L_C := L_C^1 \cup \ldots \cup L_C^k$.

A lista L_{BC} corresponde aos itens definidos na região marcada como *vazia* da Figura 4.4 (b). A Figura 4.3 apresenta um exemplo do empacotamento, em uma placa, gerado pelo Algoritmo COMB. Os retângulos maiores pertencem a L_A e os retângulos menores pertencem a L_B^i.

Os valores de p e q são definidos de tal modo que estes empacotamentos sejam viáveis. O algoritmo primeiro combina a lista L_A com L_B e em seguida combina os itens não empacotados de L_A com os itens de L_C. O empacotamento final contém todos os itens de L_A ou todos os itens de $L_B \cup L_C$. O seguinte resultado é válido para este algoritmo.

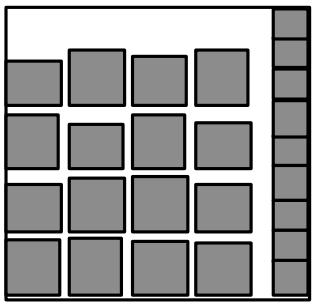


Figura 4.3: Exemplo de empacotamento gerado pelo Algoritmo COMB.

Lema 4.4.5. *Seja $\mathcal{P}_{ABC}$ um empacotamento de $L_{ABC} \subseteq L_A \cup L_B \cup L_C$ gerado pelo Algoritmo* COMB *aplicado às sublistas L_A e $L_B \cup L_C$. Então* $\#(\mathcal{P}_{ABC}) \leq \frac{1}{\left(\frac{m}{m+1}\right)^2 + \frac{m}{k(m+1)}} S(L_{ABC}) + 2k$.

Prova. Em cada placa do empacotamento $\mathcal{P}_{ABC}$, exceto em $2k$ delas ($2k$ placas das combinações feitas entre L_A e L_B^i e das combinações entre L_A e L_C^i, $i = 1, \ldots, k$) temos m^2 retângulos de L_1, cada uma com área de pelo menos $\frac{1}{(m+1)^2}$, e $m - 1 + i$ retângulos de $L_B^i \cup L_C^i$, $i \geq 2$, cada uma com área de pelo menos $\frac{1}{k}\frac{1}{m+i}$. Portanto temos uma ocupação de área, nestas placas, de pelo menos $m^2 \frac{1}{(m+1)^2} + (m-1+i)\frac{1}{k}\frac{1}{m+i} \geq \left(\frac{m}{m+1}\right)^2 + \frac{m}{k(m+1)}$. □

Algoritmo STP_m

Entrada: Lista de retângulos $L = (r_1, \ldots, r_n)$ com $x(r_i) \leq \frac{1}{m}$ e $y(r_i) \leq \frac{1}{m}$.

Saída: Empacotamento de L em placas $R = (1, 1)$.

1 Seja $p \leftarrow \frac{\sqrt{9m^4+34m^3+41m^2+20m+4} - m^2 - 3m - 2}{2m(m^2+3m+2)}$;

$q \leftarrow \frac{1-p}{m}$;

$k \leftarrow \left\lceil \frac{2m(m+2)}{3m^2+7m+2-\sqrt{9m^4+34m^3+41m^2+20m+4}} \right\rceil$.

2 Seja $L_A \leftarrow \left\{ r \in L : r \in \left(\frac{1}{m+1}, q\right] \times \left(\frac{1}{m+1}, q\right] \right\}$;

$L_B^i \leftarrow \left\{ r \in L : r \in \left(\frac{1}{k}, p\right] \times \left(\frac{1}{m+i}, \frac{1}{m-1+i}\right] \right\}, \quad i = 1, \ldots, k;$

$L_B \leftarrow L_B^1 \cup \ldots \cup L_B^k$;

$L_C^i \leftarrow \left\{ r \in L \setminus L_B : r \in \left(\frac{1}{m+i}, \frac{1}{m-1+i}\right] \times \left(\frac{1}{k}, p\right] \right\}, \quad i = 1, \ldots, k;$

$L_C \leftarrow L_C^1 \cup \ldots \cup L_C^k$.

3 Combine a lista L_A com L_B^i e L_C^i, $i = 1, \ldots, k$ da seguinte maneira.

$\mathcal{P}_{AB} \leftarrow \mathrm{COMB}^{\mathrm{x}}(L_A, L_B^1, m) \| \ldots \| \mathrm{COMB}^{\mathrm{x}}(L_A, L_B^k, m + k - 1)$.

Seja L_{AB} o conjunto dos retângulos empacotados em $\mathcal{P}_{AB}$.

$\mathcal{P}_{AC} \leftarrow \mathrm{COMB}^{\mathrm{y}}(L_A \setminus L_{AB}, L_C^1, m) \| \ldots \| \mathrm{COMB}^{\mathrm{y}}(L_A \setminus L_{AB}, L_C^k, m + k)$;

$\mathcal{P}_{ABC} \leftarrow \mathcal{P}_{AB} \| \mathcal{P}_{AC}$.

Seja L_{ABC} o conjunto dos retângulos empacotados em $\mathcal{P}_{ABC}$.

$L \leftarrow L \setminus L_{ABC}$.

4 Subdivida a lista L nas seguintes sublistas

$$L_1 \leftarrow L \cap \mathcal{C}^{xy}\left[\tfrac{1}{m+1}, \tfrac{1}{m} \,;\, \tfrac{1}{m+1}, \tfrac{1}{m}\right], \quad L_2 \leftarrow L \cap \mathcal{C}^{xy}\left[\tfrac{1}{m+2}, \tfrac{1}{m+1} \,;\, \tfrac{1}{m+1}, \tfrac{1}{m}\right],$$
$$L_3 \leftarrow L \cap \mathcal{C}^{xy}\left[\tfrac{1}{m+3}, \tfrac{1}{m+2} \,;\, \tfrac{1}{m+1}, \tfrac{1}{m}\right], \quad L_4 \leftarrow L \cap \mathcal{C}^{xy}\left[0, \tfrac{1}{m+3} \,;\, \tfrac{1}{m+1}, \tfrac{1}{m}\right],$$
$$L_5 \leftarrow L \cap \mathcal{C}^{xy}\left[\tfrac{1}{m+1}, \tfrac{1}{m} \,;\, \tfrac{1}{m+2}, \tfrac{1}{m+1}\right], \quad L_6 \leftarrow L \cap \mathcal{C}^{xy}\left[\tfrac{1}{m+1}, \tfrac{1}{m} \,;\, \tfrac{1}{m+3}, \tfrac{1}{m+2}\right],$$
$$L_7 \leftarrow L \cap \mathcal{C}^{xy}\left[\tfrac{1}{m+1}, \tfrac{1}{m} \,;\, 0, \tfrac{1}{m+3}\right], \quad L_8 \leftarrow L \cap \mathcal{C}^{xy}\left[\tfrac{1}{m+2}, \tfrac{1}{m+1} \,;\, \tfrac{1}{m+2}, \tfrac{1}{m+1}\right].$$

$$\text{Se } L_A \subseteq L_{ABC} \quad \begin{cases} L_9 \leftarrow L \cap \mathcal{C}^{xy}\left[0, \frac{1}{m+2}\ ;\ \frac{1}{m+2}, \frac{1}{m+1}\right], \\ L_{10} \leftarrow L \cap \mathcal{C}^{xy}\left[\frac{1}{m+2}, \frac{1}{m+1}\ ;\ 0, \frac{1}{m+2}\right], \\ L_{11} \leftarrow L \cap \mathcal{C}^{xy}\left[0, \frac{1}{m+2}\ ;\ 0, \frac{1}{m+2}\right]. \end{cases}$$

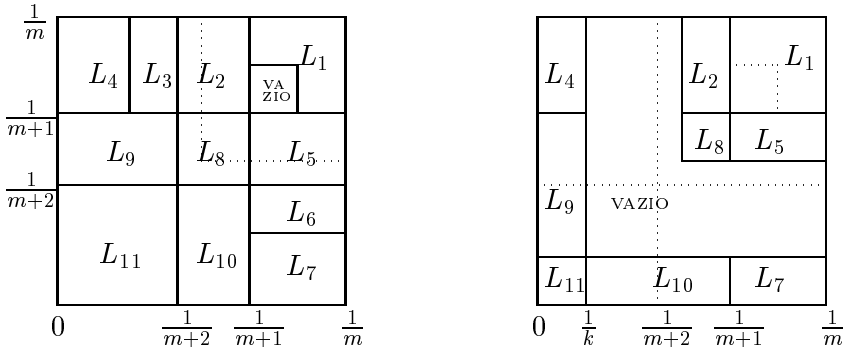
$$\text{senão} \quad \begin{cases} L_9 \leftarrow L \cap \mathcal{C}^{xy}\left[0, \frac{1}{m+2}\ ;\ \frac{1}{k}, \frac{1}{m+1}\right] \cap \mathcal{C}^{y}_{x}, \\ L_{10} \leftarrow L \cap \mathcal{C}^{xy}\left[\frac{1}{k}, \frac{1}{m+1}\ ;\ 0, \frac{1}{m+2}\right] \cap \mathcal{C}^{x}_{y}, \\ L_{11} \leftarrow L \cap \mathcal{C}^{xy}\left[0, \frac{1}{m+2}\ ;\ 0, \frac{1}{m+2}\right]. \end{cases}$$

5 Construa os seguintes empacotamentos

$\mathcal{P}_i \leftarrow \text{NFDH}^{\text{x}}(L_i), \quad i = 1, 2, 3, 4, 8, 9, 11;$
$\mathcal{P}_i \leftarrow \text{NFDH}^{\text{y}}(L_i), \quad i = 5, 6, 7, 10;$
$\mathcal{P}_{aux} \leftarrow \mathcal{P}_2 \| \ldots \| \mathcal{P}_{11};$
$\mathcal{P}_{opt} \leftarrow \mathcal{P}_1 \| \mathcal{P}_{ABC};$
$\mathcal{P} \leftarrow \mathcal{P}_{opt} \| \mathcal{P}_{aux}.$

6 Retorne $\mathcal{P}$.

Fim algoritmo.



(a) $L_A \subseteq L_{ABC}$ (b) $L_{BC} \subseteq L_{ABC}$

Figura 4.4: Subdivisão dos itens após combinação dos conjuntos L_A com L_{BC}.

O seguinte teorema apresenta um limite de desempenho assintótico para o Algoritmo STP_m.

Teorema 4.4.6. *Para qualquer lista L de retângulos, com dimensões menores ou iguais a $\frac{1}{m}$,*

$$\text{STP}_m(L) \leq \alpha_m \cdot \text{OPT}(L) + \mathcal{O}(m),$$

onde $\alpha_m = (2m^3 + 5m^2 + 5m + 2 + \sqrt{9m^4 + 34m^3 + 41m^2 + 20m + 4})/(2m(m+1)^2)$.

Prova. Primeiramente, note que o empacotamento $\mathcal{P}_{opt} := \mathcal{P}_1 \| \mathcal{P}_{ABC}$ é um empacotamento assintótico ótimo para a lista $L_{opt} := L_1 \cup L_{ABC}$. É suficiente notar que em cada empacotamento

parcial em $\mathcal{P}_{opt}$ cada placa usada, exceto talvez a última, contém m^2 retângulos com dimensões em $\left(\frac{1}{m+1}, \frac{1}{m}\right]$. Assim,

$$\#(\mathcal{P}_{opt}) \leq \mathrm{OPT}(L_{opt}) + 2k + 1. \tag{4.7}$$

No caso do empacotamento $\mathcal{P}_{ABC}$, note que para cada empacotamento gerado pelo Algoritmo COMB, a garantia de área é pelo menos $\left(\frac{m}{m+1}\right)^2 + \frac{m}{k(m+1)}$. Assim, a seguinte inequação é válida

$$\#(\mathcal{P}_{ABC}) \leq \frac{1}{\left(\frac{m}{m+1}\right)^2 + \frac{m}{k(m+1)}} S(L_{ABC}) + 2k. \tag{4.8}$$

Agora, dividimos a prova em dois casos, de acordo com o passo 3 na descrição do algoritmo. Considere primeiro o caso em que todos os retângulos de L_A são empacotados em $\mathcal{P}_{ABC}$ (o outro caso refere-se ao caso em que todos os retângulos de $L_B \cup L_C$ são empacotados em $\mathcal{P}_{ABC}$).

Caso 1. $L_A \subseteq L_{ABC}$

Vamos analisar cada um dos empacotamentos $\mathcal{P}_1, \ldots, \mathcal{P}_{11}, \mathcal{P}_{opt}$ e $\mathcal{P}_{aux}$. O empacotamento $\mathcal{P}_1$ é gerado aplicando-se o Algoritmo $\mathrm{NFDH}^{(p)}$ sobre a lista L_1. Como a área de cada retângulo de L_1 é pelo menos $q/(m+1)$, pelo Lema 4.4.5,

$$\#(\mathcal{P}_1) \leq \frac{m+1}{m^2 q} S(L_1) + 1. \tag{4.9}$$

Como $\frac{m^2 q}{m+1} \leq \left(\frac{m}{m+1}\right)^2 + \frac{m}{k(m+1)}$, de (4.8) e (4.9) temos

$$\#(\mathcal{P}_{opt}) \leq \frac{m+1}{m^2 q} S(L_{opt}) + 2k + 1. \tag{4.10}$$

Para os empacotamentos $\mathcal{P}_2, \ldots, \mathcal{P}_{11}$, as análises são análogas às feitas para o empacotamento $\mathcal{P}_1$. Assim,

$$\#(\mathcal{P}_{aux}) \leq \frac{1}{m/(m+2)} S(L_{aux}) + 10. \tag{4.11}$$

Sejam $\mathcal{H}_1$ e $\mathcal{H}_2$ definidos como $\mathcal{H}_1 := \#(\mathcal{P}_{opt}) - 2k - 1$ e $\mathcal{H}_2 := \#(\mathcal{P}_{aux}) - 10$.

Como $\mathrm{OPT}(L) \geq S(L)$, de (4.10), (4.11) e da definição de $\mathcal{H}_1$ e $\mathcal{H}_2$, temos

$$\mathrm{OPT}(L) \geq \frac{m^2 q}{m+1} \cdot \mathcal{H}_1 + \frac{m}{m+2} \mathcal{H}_2. \tag{4.12}$$

De (4.12) e usando a definição de $\mathcal{H}_1$ em (4.7) obtemos

$$\mathrm{OPT}(L) \geq \max\left\{\mathcal{H}_1, \frac{m^2 q}{m+1} \cdot \mathcal{H}_1 + \frac{m}{m+2} \mathcal{H}_2\right\}. \tag{4.13}$$

Como $\#(\mathcal{P}) = \#(\mathcal{P}_{opt}) + \#(\mathcal{P}_{aux}) = (\mathcal{H}_1 + \mathcal{H}_2) + 2k + 11$, temos

$$\#(\mathcal{P}) \leq \gamma'_m \cdot \mathrm{OPT}(L) + 2k + 11, \tag{4.14}$$

onde $\gamma'_m = \frac{\mathcal{H}_1+\mathcal{H}_2}{\max\left\{\mathcal{H}_1, \frac{m^2q}{m+1}\mathcal{H}_1+\frac{m}{m+2}\mathcal{H}_2\right\}}$. A prova deste caso segue observando que $\gamma'_m \leq \alpha_m$.

Caso 2. $L_{BC} \subseteq L_{ABC}$

Neste caso, a prova é análoga ao caso anterior. Basicamente, as inequações para este caso foram obtidas analisando-se a ocupação de área em cada placa, exceto talvez em um número fixo delas. Ou seja, cada inequação vem da aplicação direta do Lema 4.2.1.

$$\#(\mathcal{P}_1) \leq \frac{1}{m^2/(m+1)^2}S(L_1)+1; \tag{4.15}$$

$$\#(\mathcal{P}_{opt}) \leq \frac{1}{m^2/(m+1)^2}S(L_{opt})+2k+1. \tag{4.16}$$

Para as listas L_2 e L_5, obtemos um empacotamento onde cada placa, exceto talvez a última, tem $(m+1)m$ retângulos, cada um com área de pelo menos $p\frac{1}{m+1}$. Assim, temos

$$\#(\mathcal{P}_i) \leq \frac{1}{(m+1)mp/(m+1)}S(L_i)+1, \quad i=2,5. \tag{4.17}$$

Como no empacotamento das listas L_4, L_7 e L_{11}, temos que cada placa tem garantia de área de pelo menos $\left(1-\frac{1}{m}\right)\frac{m}{m+1}$, exceto talvez na última, temos

$$\#(\mathcal{P}_i) \leq \frac{1}{\left(1-\frac{1}{k}\right)\frac{m}{m+1}}S(L_i)+1. \quad i=3,4,6,7,8,11. \tag{4.18}$$

A sublista L_9 pode ser dividida em $k-m$ sublistas, $L_9^i = L_9 \cap \mathcal{Y}[\frac{1}{i+1}, \frac{1}{i}]$, $i=m+1,\ldots,k-1$. Como usamos o Algoritmo NFDH$^{(p)}$, sobre a lista L_9, temos que o empacotamento é dividido em faixas, cada uma com comprimento ocupado pelos retângulos de pelo menos $\left(1-\frac{1}{k}\right)$. A ocupação em relação à largura das faixas é pelo menos $\frac{m}{m+1}$. Portanto a ocupação de área para uma sublista L_9^i é pelo menos $\left(1-\frac{1}{k}\right)\frac{m}{m+1}$. O mesmo raciocínio pode ser feito para a lista L_{10}. Assim, a seguinte inequação é válida.

$$\#(\mathcal{P}_i) \leq \frac{1}{\left(1-\frac{1}{k}\right)\frac{m}{m+1}}S(L_i)+k-m, \quad i=9,10. \tag{4.19}$$

Como $mp \leq \left(1-\frac{1}{k}\right)\frac{m}{m+1}$, de (4.17), (4.18) e (4.19) obtemos

$$\#(\mathcal{P}_{aux}) \leq \frac{1}{mp}S(L_{aux})+2k-2m+8. \tag{4.20}$$

Definindo $\mathcal{H}_1$ e $\mathcal{H}_2$ como $\mathcal{H}_1 := \#(\mathcal{P}_{opt})-2k-1$ e $\mathcal{H}_2 := \#(\mathcal{P}_{aux})-(2k-2m+8)$, e procedendo como no caso 1, temos

$$\#(\mathcal{P}) \leq \gamma''_m \cdot \mathrm{OPT}(L)+4k-2m+8, \tag{4.21}$$

onde $\gamma''_m = \frac{\mathcal{H}_1+\mathcal{H}_2}{\max\left\{\mathcal{H}_1, \frac{m^2}{(m+1)^2}\cdot\mathcal{H}_1+mp\mathcal{H}_2\right\}}$. Como $\gamma''_m \leq \alpha_m$, a prova deste caso está completa.

Assim, a prova do teorema está completa. □

Para esclarecer os valores das constantes envolvidas, observamos que os valores de p e q são definidos de forma que ambos os casos dêem os mesmos limites de desempenho. O valor de k foi obtido de forma a ser o menor inteiro positivo tal que $\frac{m^2q}{m+1} \leq \left(\frac{m}{m+1}\right)^2 + \frac{m}{k(m+1)}$ (usado em (4.10)) e $mp \leq \left(1-\frac{1}{k}\right)\frac{m}{m+1}$ (usado em (4.20)).

Em [20], Frenk e Galambos apresentam um algoritmo para o empacotamento em placas, chamado HNF *(Hibrid Next Fit)*, e fazem sua análise para o empacotamento de itens pequenos. A seguinte tabela apresenta os valores de $\alpha_m = \alpha(\mathrm{STP}_m)$, $\alpha(\mathrm{HNF})$ e $p = p(m)$ para valores de m entre 1 e 10.

m	$p = p(m)$	$\alpha(\mathrm{STP}_m)$	$\alpha(\mathrm{HNF})$
1	0,36602540...	3,04903810...	3,38206041...
2	0,27041649...	2,02722200...	2,84623550...
3	0,21339180...	1,68340642...	1,95357991...
4	0,17603986...	1,51124783...	1,64587967...
5	0,14976179...	1,40805542...	1,48878775...
6	0,13028998...	1,33937902...	1,39323320...
7	0,11529032...	1,29041589...	1,32892571...
8	0,10338357...	1,25375953...	1,28267924...
9	0,09370374...	1,22529634...	1,24781941...
10	0,08568010...	1,20256011...	1,22060201...

Tabela 4.1: Valores de $\alpha(\mathrm{STP}_m)$ e $\alpha(\mathrm{HNF})$ para valores de m entre 1 e 10.

4.4.5 Empacotamento *On-Line* de Retângulos Pequenos em Placas

Nesta seção apresentamos um algoritmo *on-line* para empacotar retângulos pequenos em placas.

Primeiramente, descrevemos um algoritmo chamado $\mathrm{FF}_p^{(2)}$, para $0 < p < 1$, que é usado como subrotina e usa um conjunto de arredondamento $\mathcal{S} = \{1, p, p^2, \ldots, p^i, \ldots\}$. Cada retângulo tem sua largura *arredondada*, para cima, para o valor mais próximo em $\mathcal{S}$, digamos $p^i \in \mathcal{S}$. Neste caso, dizemos que este retângulo é um *i-retângulo*. Descrevemos abaixo este algoritmo.

Algoritmo $\mathrm{FF}_p^{(2)}$

Entrada: Lista de retângulos $L = (r_1, \ldots, r_n)$.

Saída: Empacotamento de L em placas $R = (1, 1)$.

1 Para $j \leftarrow 1$ até n faça

1.1 Seja i tal que r_j é um i-retângulo.

1.2 Considere o retângulo r_j como uma barra de comprimento $x(r_j)$ e considere as i-faixas criadas até o momento como barras de comprimento 1. Use a estratégia FF para empacotar r_j nas i-faixas.

1.3 Se uma nova i-faixa foi criada no passo 1.2, digamos $p^i \in \mathcal{S}$, então use novamente a estratégia FF para empacotar esta i-faixa dentro das placas. Caso necessário, empacote em nova placa.

2 Retorne o empacotamento gerado no passo 1.

Fim algoritmo.

Li e Cheng provaram que o Algoritmo $\mathrm{FF}_p^{(2)}$ tem um limite de desempenho assintótico que pode se tornar tão próximo de 2,89 quanto se queira, bastando para isso que p seja próximo de 1 e para algum s, $p^s = \frac{1}{2}$.

Descrevemos agora um algoritmo *on-line* para empacotamento de retângulos pequenos em placas. A idéia básica desse algoritmo é garantir uma área mínima de utilização em cada placa, para retângulos de L relativamente grandes, usando apenas subdivisão de listas. Para os retângulos de dimensões pequenas, é usado o Algoritmo $\mathrm{FF}_p^{(2)}$.

Algoritmo $\mathrm{OFF}_m^{(2)}$

Entrada: Lista de retângulos $L = (r_1, \ldots, r_n)$ com $x(r_i) \leq \frac{1}{m}$ e $y(r_i) \leq \frac{1}{m}$.

Saída: Empacotamento de L em placas $R = (1, 1)$.

1 Divida a lista L em sublistas $L_1, \ldots, L_6$ da seguinte forma (veja a Figura 4.2)

$$\begin{aligned}
L_1 &\leftarrow L \cap \mathcal{C}^{xy}\left[\tfrac{1}{m+1}, \tfrac{1}{m}\ ;\ \tfrac{1}{m+1}, \tfrac{1}{m}\right], \\
L_2 &\leftarrow L \cap \mathcal{C}^{xy}\left[0, \tfrac{1}{m+1}\ ;\ \tfrac{1}{m+1}, \tfrac{1}{m}\right], \\
L_3 &\leftarrow L \cap \mathcal{C}^{xy}\left[\tfrac{1}{m+1}, \tfrac{1}{m}\ ;\ 0, \tfrac{1}{m+1}\right], \\
L_4 &\leftarrow L \cap \mathcal{C}^{xy}\left[0, \tfrac{1}{m+1}\ ;\ \tfrac{1}{m+2}, \tfrac{1}{m+1}\right] \cap \mathcal{C}^{y}_{x}, \\
L_5 &\leftarrow L \cap \mathcal{C}^{xy}\left[\tfrac{1}{m+2}, \tfrac{1}{m+1}\ ;\ 0, \tfrac{1}{m+1}\right] \cap \mathcal{C}^{x}_{y}, \\
L_6 &\leftarrow L \cap \mathcal{C}^{xy}\left[0, \tfrac{1}{m+2}\ ;\ 0, \tfrac{1}{m+2}\right].
\end{aligned}$$

2 $\mathcal{P}_i \leftarrow \mathrm{FF}^{(x2)}(L_i), \quad i = 1, 2, 4;$

3 $\mathcal{P}_i \leftarrow \mathrm{FF}^{(y2)}(L_i), \quad i = 3, 5;$

4 $\mathcal{P}_6 \leftarrow \mathrm{FF}^{(2)}_p(L_6), \quad$ para $\quad p = \frac{(m+2)^2 m^2}{(m+1)^2 (m+1)^2}$;

5 $\mathcal{P} \leftarrow \bigcup_{i=1}^{6} \mathcal{P}_i$.

6 Retorne $\mathcal{P}$.

Fim algoritmo.

Mostramos a seguir que o Algoritmo $\mathrm{OFF}^{(2)}_m$ tem um limite de desempenho assintótico $((m+1)/m)^2$. A desvantagem deste algoritmo é que a constante aditiva β associado ao limite de desempenho assintótico, depende de m. Quando m cresce, o limite de desempenho assintótico se torna bem próximo de 1 mas o valor de β cresce. Note que se fixarmos p como uma constante —em vez de uma função em m— o fator de perda, devido ao arredondamento, também se torna constante.

Lema 4.4.7. *Para qualquer lista L de retângulos, com dimensões menores ou iguais a $\frac{1}{m}$, $\mathrm{OFF}^{(2)}_m(L) \leq \left(\frac{m+1}{m}\right)^2 \cdot S(L) + \mathcal{O}(m^2)$.*

Prova. É fácil ver que $\#(\mathcal{P}_i) \leq \left(\frac{m+1}{m}\right)^2 S(L_i) + 1$, para $i = 1, \ldots, 5$.

Para a lista L_6, provaremos que $\#(\mathcal{P}_6) \leq \left(\frac{m+1}{m}\right)^2 S(L_6) + \mathcal{O}(m^2)$.

Considere L^k_6 os retângulos de L_6 que têm largura em $(p^{k+1}, p^k]$, $k \geq 0$. Denotamos por n_k o número de faixas criadas para o empacotamento de L^k_6, temos que

$$\begin{aligned}
S(L^k_6) &= \sum_{r \in L^k_6} x(r) \cdot y(r) \\
&> p^{k+1} \cdot \left(1 - \frac{1}{m+2}\right) \cdot (n_k - 1) \\
&= p^{k+1} \cdot \left(\frac{m+1}{m+2}\right) \cdot (n_k - 1)\ .
\end{aligned}$$

Logo,

$$\begin{aligned} S(L_6) &= \sum_{k\geq 0} S(L_6^k) \\ &> \sum_{k\geq 0} p^{k+1} \cdot \left(\frac{m+1}{m+2}\right) \cdot (n_k - 1) \\ &= \left(\frac{m+1}{m+2}\right) p \left(\sum_{k\geq 0} p^k n_k - \sum_{k\geq 0} p^k\right). \end{aligned}$$

Como $\sum_{k\geq 0} p^k n_k$ é a soma das larguras de todas as faixas, e em cada placa, exceto talvez na última, a largura preenchida pelas faixas é pelo menos $\left(1 - \frac{1}{m+2}\right)$, temos:

$$\begin{aligned} S(L_6) &\geq \left(\frac{m+1}{m+2}\right) p \left(\left(1 - \frac{1}{m+2}\right)(\#(\mathcal{P}_6) - 1) - \mathcal{O}(m^2)\right) \\ &= \left(\frac{m}{m+1}\right)^2 \cdot \#(\mathcal{P}_6) - \mathcal{O}(m^2)\,. \end{aligned}$$

Assim, temos que

$$\#(\mathcal{P}) \leq \left(\frac{m+1}{m}\right)^2 S(L) + \mathcal{O}(m^2)\,. \tag{4.22}$$

□ Com este lema, concluímos o seguinte resultado.

Teorema 4.4.8. *Para qualquer lista L de retângulos, com dimensões menores ou iguais a $\frac{1}{m}$,*

$$\mathrm{OFF}_m^{(2)}(L) \leq \left(\frac{m+1}{m}\right)^2 \cdot \mathrm{OPT}(L) + \mathcal{O}(m^2).$$

Este algoritmo também pode ser refinado de modo a termos um melhor limite de desempenho assintótico. Podemos fazer isto da mesma forma como feito para o Algoritmo $\mathrm{IOSS}_{m,p}$ em relação ao Algoritmo $\mathrm{OSSP}_{m,p}$. Chamaremos esta versão refinada de IOFF_m. O Algoritmo IOFF_m divide a lista L em sublistas $L_1, \ldots, L_6$ como apresentado na figura seguinte.

Para a lista L_1 o algoritmo empacota m^2 retângulos por placa. Com isto temos um empacotamento ótimo para a lista L_1 com garantia de área $\left(\frac{m}{m+1}\right)^2$. Para as listas $L_2, \ldots, L_5$, o Algoritmo IOFF_m aplica uma variante do Algoritmo $\mathrm{NF}^{(p)}$ para cada sublista, obtendo uma garantia de área de pelo menos $m \cdot (m+1)\frac{1}{m+1}\frac{1}{m+2} = \frac{m}{m+2}$. Para a lista L_6, o Algoritmo IOFF_m aplica o Algoritmo $\mathrm{OFF}_{m+1}^{(2)}$, obtendo para este empacotamento uma garantia de área de pelo menos $\left(\frac{(m+1)+1}{m+1}\right)^2 \geq \frac{m}{m+2}$. Por fim, o Algoritmo IOFF_m retorna a concatenação dos empacotamentos gerados para cada sublista. Assim, novamente temos um empacotamento para L dividido em duas partes. Um empacotamento ótimo com garantia de área $\left(\frac{m}{m+1}\right)^2$ e outro

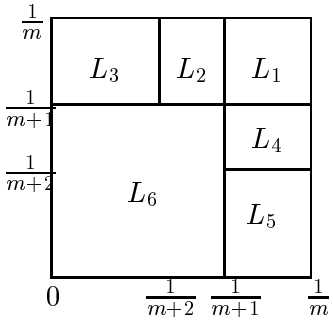


Figura 4.5: Partição da lista L gerada pelo Algoritmo IOFF$_m$.

com garantia de área $\frac{m}{m+2}$. Com isto, podemos ter um limite de desempenho assintótico para o Algoritmo IOFF$_m$ de $\alpha_m = \frac{h_1+h_2}{\max\{h_1,\left(\frac{m}{m+1}\right)^2 h_1+\left(\frac{m}{m+2}\right)h_2\}}$. Assim, vale que

Teorema 4.4.9. *Para qualquer lista L de retângulos, com dimensões menores ou iguais a $\frac{1}{m}$,*

$$\text{IOFF}_m(L) \leq \alpha_m \cdot \text{OPT}(L) + \mathcal{O}(m^2),$$

onde $\alpha_m \leq \left(\frac{m+2}{m+1}\right)^2 + \frac{2}{m(m+1)}$.

4.4.6 Empacotamento de quadrados em quadrados

Nesta seção estamos interessados no empacotamento de quadrados em quadrados. O algoritmo que apresentamos tem descrição que pode ser usada tanto para o caso geral (de empacotamento de quadrados em quadrados) como para empacotamento de itens pequenos. O limite do caso geral não é melhor que o limite de 2,125 do Algoritmo HFF, mas além de ser usado para o empacotamento de itens pequenos, o algoritmo é usado como subrotina de outro algoritmo para empacotamento de caixas de fundo quadrado para o PET.

Antes de descrevermos o algoritmo desta seção, descreveremos o Algoritmo GQ$_m^{(p)}$, que será usado como subrotina e é uma adaptação do Algoritmo GQ$_m$ descrito em [40], para empacotar caixas de $\mathcal{Q}_m$. Lembramos aqui que $\mathcal{Q}_m := \{b : \ 0 < x(b) = y(b) \leq \frac{a}{m}\}$, $m \geq 1$.

Meir e Moser [45] provaram que o Algoritmo NFDH$^{(p)}$ pode empacotar uma lista de quadrados $L_i \subset \mathcal{Q}_m$, $m \geq 2$, em um quadrado unitário se a área total de L_i não é superior a $\left(\frac{1}{m^2} + \left(\frac{m-1}{m}\right)^2\right)$. O Algoritmo GQ$_m^{(p)}$ divide L em sublistas $L_1, \ldots, L_v$ e usa o Algoritmo NFDH$^{(p)}$ para empacotar cada sublista L_i em apenas uma placa, sendo que para cada sublista L_i valem as seguintes inequações.

$$\begin{aligned} S(L_i) &\leq \left[\left(\tfrac{m-1}{m}\right)^2 + \left(\tfrac{1}{m}\right)^2\right] && \text{para } \ i = 1, \ldots, v, \\ S(L_i) + S(\mathit{first}(L_{i+1})) &> \left[\left(\tfrac{m-1}{m}\right)^2 + \left(\tfrac{1}{m}\right)^2\right] && \text{para } \ i = 1, \ldots, v-1. \end{aligned}$$

Lema 4.4.10. *Seja $L \subset \mathcal{Q}_m$, $m \geq 2$. Então,* $\mathrm{GQ}_m^{(p)}(L) \leq \left(\frac{m}{m-1}\right)^2 S(L) + 1$.

O próximo lema será usado na demonstração de um limite inferior para certos empacotamentos usando o Algoritmo $\mathrm{GQ}_m^{(p)}$.

Lema 4.4.11. *Seja $\mathcal{A}$ um algoritmo para o* PEP *para empacotar uma lista $L \subset \mathcal{Q}^{xy}[0,1]$ dentro de uma placa $B = (1,1)$. Se $\mathcal{A}$ subdivide a lista de entrada L em duas sublistas $L_1 \in \wp_4$ e $L_2 \in \mathcal{Q}_m$, $m \geq 2$, e aplica o Algoritmo $GQ_m^{(p)}$ para empacotar L_2, então $\mathcal{A}$ tem um limite de desempenho assintótico de pelo menos $\frac{4(m-1)^2+3m^2}{4(m-1)^2}$.*

Prova. Considere placas $P = (1,1)$. Seja L uma lista de quadrados, $L = L_1 \cup L_2$, com $L_1 \in \wp_4$ e $L_2 \in \mathcal{Q}_m$, $m \geq 3$. Seja $L_1 = (r'_1, \ldots, r'_{N'})$ e $L_2 = (r''_1, \ldots, r''_{M \cdot N''})$, onde

$$r'_i = \left(\frac{1}{2} + \frac{1}{k}, \frac{1}{2} + \frac{1}{k}\right), \quad \text{e} \quad r''_i = \begin{cases} \left(\frac{1}{m}, \frac{1}{m}\right), & i \mod M = 1 \\ \left(\frac{1}{k}, \frac{1}{k}\right), & \text{caso contrário.} \end{cases}$$

Lembrando que o Algoritmo $\mathrm{GQ}_m^{(p)}$ agrupa os primeiros quadrados com área total não maior que $\left(\frac{m-1}{m}\right)^2 + \left(\frac{1}{m}\right)^2$, esta instância foi construída de modo que no empacotamento gerado pelo Algoritmo $\mathrm{GQ}_m^{(p)}$ cada placa tenha M quadrados cujas áreas somam $\left(\frac{m-1}{m}\right)^2 + \left(\frac{1}{k}\right)^2$.

Note que o Algoritmo $\mathrm{GQ}_m^{(p)}$ divide a lista L_2 em N'' sublistas, cada sublista consistindo de um quadrado da forma $\left(\frac{1}{m}, \frac{1}{m}\right)$ e $M-1$ quadrados da forma $\left(\frac{1}{k}, \frac{1}{k}\right)$.

A instância $L = L_1 \cup L_2$, é construída de modo que o empacotamento ótimo consiste de N' placas, cada placa contendo um quadrado de L_1 e o espaço remanescente (em cada placa) é quase todo preenchido com placas de L_2. Tomando-se N' e k como inteiros bem grandes, $N'' = \left\lceil \frac{3}{4} N' \left(\frac{m}{m-1}\right)^2 \right\rceil$ e k um múltiplo de $2m$, podemos escolher M apropriadamente de forma que $r(\mathcal{A})$ pode ficar tão próximo de $\frac{4(m-1)^2+3m^2}{4(m-1)^2}$ quanto se queira. □

A seguir apresentamos o Algoritmo SS_1 para o empacotamento de quadrados em quadrados.

Algoritmo SS_1

Entrada: Lista de quadrados $L \subset \mathcal{Q}^{xy}[0,1]$.

Saída: Empacotamento $\mathcal{P}$ de L em placas $(1,1)$.

1 Seja $p \leftarrow 0{,}37123918$ e $q \leftarrow 1 - p$.

2 Divida a lista L em sublistas, $L'_1, L_A, L'_2, L_B, L_3$ e L_4 (veja Figura 4.6),

$$\begin{array}{ll} L'_1 \leftarrow L \bigcap \mathcal{Q}\left[\frac{1}{2}, 1\ ;\ \frac{1}{2}, 1\right], & L_A \leftarrow L \bigcap \mathcal{Q}\left[\frac{1}{2}, q\ ;\ \frac{1}{2}, q\right], \\ L'_2 \leftarrow L \bigcap \mathcal{Q}\left[\frac{1}{3}, \frac{1}{2}\ ;\ \frac{1}{3}, \frac{1}{2}\right], & L_B \leftarrow L \bigcap \mathcal{Q}\left[\frac{1}{3}, p\ ;\ \frac{1}{3}, p\right], \\ L_3 \leftarrow L \bigcap \mathcal{Q}\left[\frac{1}{4}, \frac{1}{3}\ ;\ \frac{1}{4}, \frac{1}{3}\right], & L_4 \leftarrow L \bigcap \mathcal{Q}\left[0, \frac{1}{4}\ ;\ 0, \frac{1}{4}\right]. \end{array}$$

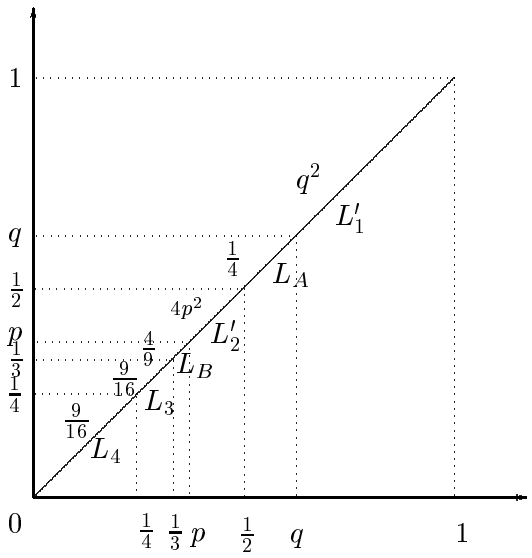


Figura 4.6: Partição da lista L feita pelo Algoritmo SS_1.

3 Empacote um quadrado de L_A e três quadrados de L_B por placa, até que ou todos os itens de L_A tenham sido empacotados, ou todos os itens de L_B tenham sido empacotados. Seja $\mathcal{P}_{AB}$ este empacotamento e L_{AB} o conjunto de quadrados empacotados em $\mathcal{P}_{AB}$.

4 $L_i \leftarrow L'_i \setminus L_{AB}$ para $i = 1, 2, 3, 4$;

5 $\mathcal{P}_{opt} \leftarrow \mathrm{NF}^{(p)}(L_1) \| \mathcal{P}_{AB}$;

6 $\mathcal{P}_{aux} \leftarrow \mathrm{NFDH}^{(p)}(L_2) \| \mathrm{NFDH}^{(p)}(L_3) \| \mathrm{GQ}_4^{(p)}(L_4)$;

7 $\mathcal{P} = \mathcal{P}_{opt} \| \mathcal{P}_{aux}$.

8 Retorne $\mathcal{P}$.

Fim algoritmo.

Teorema 4.4.12. *Para qualquer lista de quadrados L para o* $\mathrm{PEP}(1,1)$,

$$\mathrm{SS}_1(L) \leq 2{,}361 \cdot \mathrm{OPT}(L) + 4.$$

Prova. Analisaremos dois casos, considerando o empacotamento gerado no passo 3.

Caso 1. L_A é totalmente empacotado em $\mathcal{P}_{AB}$.

Note que neste caso a área dos quadrados de L_1 é pelo menos q^2 e cada uma das placas de $\mathcal{P}_{AB}$, exceto talvez na última, tem garantia de área de pelo menos $\frac{1}{4} + 3\frac{1}{9} \geq q^2$. Assim, temos

$$\#(\mathcal{P}_{opt}) \quad \leq \quad \frac{1}{q^2} S(L_1 \cup L_{AB}) + 1.$$

Para os demais empacotamentos, note que a garantia de área ocupada em cada placa, exceto talvez nas últimas de cada empacotamento, é pelo menos $\frac{4}{9}$. Este mínimo sendo atingido pela sublista L_2. Portanto,

$$\#(\mathcal{P}_{aux}) \leq \frac{9}{4}S(L_{aux}) + 3.$$

Como não é possível empacotar mais que um quadrado de L_1' por placa, temos

$$\#(\mathcal{P}_{opt}) \leq \mathrm{OPT}(L) + 1.$$

Procedendo como feito anteriormente, temos que $\#(\mathcal{P}) \leq \alpha_1 \cdot \mathrm{OPT}(L) + 4$, onde $\alpha_1 \leq \frac{\mathcal{H}_1+\mathcal{H}_2}{\max\{\mathcal{H}_1, q^2\mathcal{H}_1+\frac{4}{9}\mathcal{H}_2\}} \leq 2{,}361$.

Caso 2. L_B é totalmente empacotado em $\mathcal{P}_{AB}$.

Neste caso temos uma garantia de área de $\frac{1}{4}$ para o empacotamento das listas L_1 e L_{AB}. Assim,

$$\#(\mathcal{P}_{opt}) \leq 4S(L_1 \cup L_{AB}) + 1.$$

Como todos os quadrados de L_B foram empacotados em $\mathcal{P}_{AB}$, a área dos quadrados em L_2 é pelo menos p^2 e os demais quadrados têm dimensões no máximo $\frac{1}{3}$. Usando o Lema 4.2.1 para os empacotamentos de L_3 e L_4, obtemos uma garantia de área que é pelo menos $\frac{9}{16}$, $\frac{9}{16} \geq 4p^2$. Portanto as seguinte inequações são válidas.

$$\begin{aligned} \#(\mathcal{P}_{aux}) &\leq \frac{1}{4p^2}S(L_{aux}) + 3, \\ \#(\mathcal{P}_{opt}) &\leq \mathrm{OPT}(L) + 1. \end{aligned}$$

Neste caso, temos que $\#(\mathcal{P}) \leq \alpha_2 \cdot \mathrm{OPT}(L) + 4$, onde $\alpha_2 \leq \frac{\mathcal{H}_1+\mathcal{H}_2}{\max\{\mathcal{H}_1, \frac{1}{4}\mathcal{H}_1+4p^2\mathcal{H}_2\}} \leq 2{,}361$.

O teorema segue dos dois casos acima. □

Proposição 4.4.13. *O limite de desempenho assintótico do Algoritmo SS_1 está entre* 2,333 *e* 2,361.

Prova. Este resultado segue diretamente do Teorema 4.4.12 e Lema 4.4.11 (com $m = 4$). □

O Algoritmo SS_1 pode ser generalizado para o empacotamento de itens em $\mathcal{Q}^{xy}\left[0, \frac{1}{m}\right]$ como no Algoritmo STP_m. Não apresentaremos esta generalização pois ela é semelhante ao próprio Algoritmo SS_1 e os valores de p e α ficam muito complexos para serem dados em função de m. Chamaremos este algoritmo de SS_m. Na Tabela 4.2, apresentamos os limites de desempenho assintótico de SS_m para valores de m entre 1 e 10. Indicamos também o respectivo valor de $p = p(m)$.

m	$\alpha(\mathrm{SS}_m)$	$p = p(m)$
1	2,36048464...	0,37123918...
2	1,83521739...	0,27185838...
3	1,59715814...	0,21398550...
4	1,46308245...	0,17634050...
5	1,37755738...	0,14993476...
6	1,31842392...	0,13039854...
7	1,27516865...	0,11536290...
8	1,24218552...	0,10343448...
9	1,21622010...	0,09374082...
10	1,19525690...	0,08570794...

Tabela 4.2: Valores de $\alpha(\mathrm{SS}_m)$ para valores de m entre 1 e 10.

4.4.7 Conjuntos Críticos $\mathcal{A}_k^{xy}$ e $\mathcal{B}_k^{xy}$

Uma estratégia muito importante que exploramos nesta tese é o conceito de conjuntos críticos. Vistos apenas isoladamente, estes conjuntos críticos apresentam uma garantia de área pequena se empacotados sozinhos (mesmo em um empacotamento ótimo). É claro que o conceito de "pequena garantia de área" depende do contexto sendo analisado. Assim, $\frac{1}{3}$ pode ser uma pequena garantia de área em um determinado contexto, mas pode ser uma boa garantia em outro contexto.

A definição destes conjuntos críticos depende muito do problema que estamos tratando. Dois destes são $\mathcal{A}_k^{xy}$ e $\mathcal{B}_k^{xy}$ que tratamos nesta seção. Estes conjuntos são usados como conjuntos críticos do PEP, PET e também são usados para definir conjuntos críticos do PEC.

Antes de definir os conjuntos $\mathcal{A}_k^{xy}$ e $\mathcal{B}_k^{xy}$, que dependem de um parâmetro k, precisamos introduzir outras definições.

Vamos assumir que k é um inteiro maior que 4.

Definição 4.4.1. *Seja* $r_1^{(k)}, r_2^{(k)}, \ldots, r_{k+15}^{(k)}$ *e* $s_1^{(k)}, s_2^{(k)}, \ldots, s_{k+14}^{(k)}$ *números reais definidos como segue:*

- $r_1^{(k)}, r_2^{(k)}, \ldots, r_k^{(k)}$ *são tais que*
  $r_1^{(k)}\frac{1}{2} = r_2^{(k)}(1 - r_1^{(k)}) = r_3^{(k)}(1 - r_2^{(k)}) = \ldots = r_k^{(k)}(1 - r_{k-1}^{(k)}) = \frac{1}{3}(1 - r_k^{(k)})$ *e* $r_1^{(k)} < \frac{4}{9}$;
- $r_{k+1}^{(k)} = \frac{1}{3},\ r_{k+2}^{(k)} = \frac{1}{4},\ \ldots,\ r_{k+15}^{(k)} = \frac{1}{17}$;
- $s_i^{(k)} = (1 - r_i^{(k)})$ *para* $i = 1, \ldots, k$;

- $s_{k+i}^{(k)} = 1 - \left(\frac{2i+4-\lfloor \frac{i+2}{3} \rfloor}{4i+10} \right)$ *para* $i = 1, \ldots, 14$.

Lema 4.4.14. *Dado* $k \geq 5$, *os números* $r_1^{(k)}, r_2^{(k)}, \ldots, r_k^{(k)}$ *existem e são tais que* $\frac{4}{9} > r_1^{(k)} > r_2^{(k)} > \cdots > r_k^{(k)} > \frac{1}{3}$. *Além disso,* $r_1^{(k)} \to \frac{4}{9}$ *à medida que* $k \to \infty$.

A seguir, demonstramos o lema 4.4.14.

Por simplicidade, omitiremos os superíndices $^{(k)}$ da notação $r_i^{(k)}$ e $s_i^{(k)}$ quando k estiver claro pelo contexto.

Considere a seguinte série de equações:

$$\frac{1}{2}r_1 = r_2(1-r_1) = r_3(1-r_2) = \ldots = r_{k-1}(1-r_{k-2}) = r_k(1-r_{k-1}) = \frac{1}{3}(1-r_k). \qquad (4.23)$$

Podemos dividir esta série em duas outras:

$$\frac{1}{2}r_1 = r_2(1-r_1) = r_3(1-r_2) = \ldots = r_{k-1}(1-r_{k-2}) = r_k(1-r_{k-1}) \qquad (4.24)$$

$$\frac{1}{2}r_1 = \frac{1}{3}(1-r_k). \qquad (4.25)$$

Note que tanto em (4.24) como em (4.25) podemos tomar r_k como uma função de r_1. Denotemos por r_k' a primeira função (usando (4.24)), e denotemos por r_k'' a segunda função (usando (4.25)). Note que (4.25) é a equação de uma linha reta (*cf.* Figura 4.7).

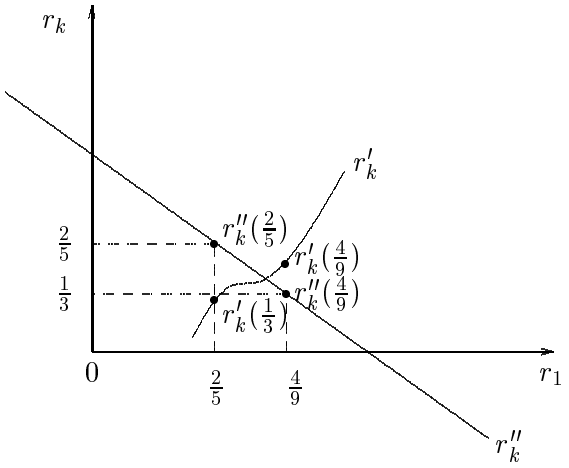


Figura 4.7: Existência dos valores $r_1, \ldots, r_k$.

A idéia de dividir a série (4.23) em duas séries e a demostração do resultado seguinte foram sugeridas por Yoshiharu Kohayakawa.

Fato 4.4.15. *Existem valores* $r_1, \ldots, r_k$, *que satisfazem a série de equações em (4.23), com* $\frac{2}{5} < r_1 < \frac{4}{9}$.

Prova. As funções r'_k e r''_k estão representadas na Figura 4.7. O propósito aqui, é mostrar a existência de um valor para r_1 entre $\frac{2}{5}$ e $\frac{4}{9}$ satisfazendo a série (4.23). Isto é o mesmo que provar que existe um valor para r_1 tal que $r'_k(r_1) = r''_k(r_1)$.

Primeiro, vamos mostrar que ambas as funções r'_k e r''_k são contínuas para valores de r_1 com $\frac{2}{5} \leq r_1 \leq \frac{4}{9}$.

A equação (4.25) é claramente contínua, pois r''_k é uma função linear.

Vamos ver agora que r'_k também é contínua. Para isto, vamos mostrar que r'_i é contínua para $\frac{2}{5} \leq r_1 \leq \frac{4}{9}$ com $0 < r'_i(r_1) < \frac{1}{2}$. Pela série (4.24), temos

$$r'_i = \frac{1}{2}\frac{r_1}{(1 - r'_{i-1})}, \tag{4.26}$$

onde $r'_1 = r_1$.

Claramente, para a função r'_2, a afirmativa é válida. Suponha que para r'_{i-1} a afirmativa seja válida. Como $0 < r'_{i-1} < \frac{1}{2}$, e r'_{i-1} é contínua, temos de (4.26) que $r'_i(r_1)$ é positiva e contínua. Por fim, temos que $r'_i(r_1) = \frac{1}{2}\frac{r_1}{(1-r'_{i-1}(r_1))} < \frac{1}{2}\frac{4/9}{1-1/2} < \frac{1}{2}$. Assim, temos que as funções r'_k e r''_k são funções contínuas para $\frac{2}{5} < r_1 < \frac{4}{9}$.

Vamos calcular o valor de $r''_k(r_1)$ para $r_1 = \frac{4}{9}$ e $r_1 = \frac{2}{5}$. Primeiro note que $r''_k(r_1) = 1 - \frac{3}{2}r_1$ e portanto, $r''_k(\frac{4}{9}) = \frac{1}{3}$ e $r''_k(\frac{2}{5}) = \frac{2}{5}$.

- Vamos estimar o valor de $r'_k(r_1)$ para $r_1 = \frac{4}{9}$. De (4.24), temos que $r_i = \frac{r_1}{2(1-r_{i-1})}$. Assim, temos a seguinte seqüência:

$$\begin{aligned} r_1 &= \tfrac{4}{9} && = 0{,}444444\ldots \\ r_2 &= \frac{\frac{4}{9}}{2(1-\frac{4}{9})} && = 0{,}4 \\ r_3 &= \frac{\frac{4}{9}}{2(1-\frac{2}{5})} && = 0{,}370370\ldots \\ r_4 &= \frac{\frac{4}{9}}{2(1-\frac{10}{27})} && = 0{,}352941\ldots \\ &\vdots \end{aligned}$$

  Suponha que sabemos que $r_{i-1} > \frac{1}{3}$. Reescrevendo a equação $r_i = \frac{2}{9}\left(\frac{1}{1-r_{i-1}}\right)$, temos que $r_{i-1} = 1 - \frac{2}{9r_i}$. Como $r_{i-1} > \frac{1}{3}$, concluímos que $r_i > \frac{1}{3}$. Portanto, como $r_1 = \frac{4}{9} > \frac{1}{3}$, segue que $r_i > \frac{1}{3}$ para todo $i \geq 1$. Ou seja, $r'_k(\frac{4}{9}) > \frac{1}{3}$.

- Vamos estimar o valor de $r'_k(r_1)$ para $r_1 = \frac{2}{5}$.

  Note que considerando a série (4.24), temos $r_i = \frac{r_1}{2(1-r_{i-1})}$. Assim temos a seguinte seqüência:

$$r_1 = \tfrac{2}{5} \quad = 0{,}4$$

$$\begin{aligned} r_2 &= \frac{\frac{2}{5}}{2(1-\frac{2}{5})} &= 0{,}333333\ldots \\ r_3 &= \frac{\frac{2}{5}}{2(1-\frac{1}{3})} &= 0{,}3 \\ r_4 &= \frac{\frac{2}{5}}{2(1-\frac{3}{10})} &= 0{,}285714\ldots \\ &\vdots \end{aligned}$$

Suponha que sabemos que $r_{i-1} < \frac{2}{5}$. Reescrevendo a equação $r_i = \frac{1}{5}\left(\frac{1}{1-r_{i-1}}\right)$ temos que $r_{i-1} = 1 - \frac{1}{5r_i}$. Como $r_{i-1} < \frac{2}{5}$ concluímos que $r_i < \frac{1}{3}$. Portanto, como $r_2 = \frac{1}{3} < \frac{2}{5}$ então $r_i < \frac{1}{3}$ para $i \geq 3$. Ou seja, $r'_k(\frac{2}{5}) < \frac{1}{3}$, para $k \geq 3$.

As considerações acima, juntamente com o fato de r'_k ser uma função contínua, $k \geq 3$, mostram que para algum r_1 entre $\frac{2}{5}$ e $\frac{4}{9}$ temos que $r'_k(r_1) = r''_k(r_1)$. □

Fato 4.4.16. *Os valores de $r_1, \ldots, r_k$, são tais que $\frac{4}{9} > r_1 > r_2 > \cdots > r_k > \frac{1}{3}$.*

Prova. Do fato anterior, temos que $r_1 < \frac{4}{9}$. Note que isto automaticamente mostra, usando (4.25), que $r_k > \frac{1}{3}$. Vamos mostrar agora que $r_1 > r_2 > \cdots > r_k$. Faremos isto por indução em k.

Pela série (4.23) temos que $r_2 = \frac{1}{2}\left(\frac{r_1}{1-r_1}\right)$. Assim, $r_2 < r_1$ se e somente se $\frac{1}{2}\left(\frac{r_1}{1-r_1}\right) < r_1$. Esta última inequação vale quando $0 < r_1 < \frac{1}{2}$, o que de fato ocorre.

Agora, suponha que $r_1 > r_2 > \cdots > r_{i-1}$ para $i \geq 3$. Vamos mostrar que $r_{i-1} > r_i$. Por (4.23) temos que

$$r_{i-1}(1 - r_{i-2}) = r_i(1 - r_{i-1}),$$

ou seja,

$$\begin{aligned} r_i &= r_{i-1}\left(\frac{1 - r_{i-2}}{1 - r_{i-1}}\right) \\ &< r_{i-1}\left(\frac{1 - r_{i-1}}{1 - r_{i-1}}\right) \\ &= r_{i-1}\,. \end{aligned}$$

□

Fato 4.4.17. *O valor de r_1 se aproxima de $\frac{4}{9}$ à medida que k cresce.*

Prova. Note que provar este fato é equivalente a provar que o valor de r_k se aproxima de $\frac{1}{3}$ à medida que k cresce.

Primeiramente, vamos provar que $\left(\frac{r_i}{r_{i-1}}\right) < \left(\frac{r_{i+1}}{r_i}\right) < 1$. De (4.23) obtemos que $r_i(1-r_{i-1}) = r_{i+1}(1-r_i)$, ou seja,

$$\left(\frac{1-r_{i-1}}{1-r_i}\right) = \left(\frac{r_{i+1}}{r_i}\right) .$$

Por outro lado, temos que $\left(\frac{r_i}{r_{i-1}}\right) < \left(\frac{1-r_{i-1}}{1-r_i}\right)$. Para ver isto, note que para $0 < r_i < \frac{1}{2}$, temos que $r_i - r_i^2 < r_{i-1} - r_{i-1}^2$. Observamos que a função $f(x) = x - x^2$ é crescente no intervalo $\left[0, \frac{1}{2}\right]$ e como $r_{i-1} > r_i$, temos que $f(r_{i-1}) > f(r_i)$.

Assim, temos que

$$\left(\frac{r_i}{r_{i-1}}\right) < \left(\frac{1-r_{i-1}}{1-r_i}\right) = \left(\frac{r_{i+1}}{r_i}\right) < 1.$$

Isto mostra que para $k \to \infty$, temos que $\frac{r_k}{r_{k-1}} \to 1$.

Em (4.23) temos que

$$r_k(1-r_{k-1}) = \frac{1}{3}(1-r_k).$$

Mais ainda, por esta equação, temos que se r_{k-1} se aproxima de r_k à medida que k cresce, então r_k deve se aproximar de $\frac{1}{3}$ à medida que k cresce. □

A demonstração do Lema 4.4.14 segue dos fatos provados acima.

Apesar do Lema 4.4.14 mostrar que $r_1^{(k)}$ converge para $\frac{4}{9}$, isto nada diz o quão rápido isto ocorre. A Tabela 4.3 mostra que o valor de r_1 se torna bem próximo de $\frac{4}{9} = 0{,}44444444\ldots$ com k ainda pequeno.

Definição de conjuntos usando os valores $r_1, \ldots, r_{k+14}$

Tendo provado a existência dos números mencionados na Definição 4.4.1, podemos agora definir alguns conjuntos.

$$\begin{aligned} \mathcal{A}_{k,i}^{xy} &= \mathcal{C}^{xy}\left[r_{i+1}, r_i \ ; \ \frac{1}{2}, s_i\right], \\ \mathcal{B}_{k,i}^{xy} &= \mathcal{C}^{xy}\left[\frac{1}{2}, s_i \ ; \ r_{i+1}, r_i\right], \end{aligned}$$

k	$r_1^{(k)}$
4	0,43844718...
5	0,44129761...
6	0,44281294...
7	0,44360837...
8	0,44401983...
9	0,44423013...
10	0,44433669...
11	0,44439040...
12	0,44441737...
13	0,44443089...
14	0,44443766...
15	0,44444105...
20	0,44444433...
25	0,44444444...

Tabela 4.3: Valores de $r_1^{(k)}$ para alguns valores de k.

$$\mathcal{A}_k^{xy} = \bigcup_{i=1}^{k+14} \mathcal{A}_{k,i}^{xy}, \qquad \mathcal{B}_k^{xy} = \bigcup_{i=1}^{k+14} \mathcal{B}_{k,i}^{xy},$$

$$\mathcal{A}_{1-k}^{xy} = \bigcup_{i=1}^{k} \mathcal{A}_{k,i}^{xy}, \qquad \mathcal{B}_{1-k}^{xy} = \bigcup_{i=1}^{k} \mathcal{B}_{k,i}^{xy},$$

$$\mathcal{A}_j^{xy'} = \mathcal{A}_{1-k}^{xy} \bigcap \mathcal{X}[0, 1-s_j] \quad \mathcal{A}_j^{xy''} = \mathcal{A}_{1-k}^{xy} \bigcap \mathcal{X}[1-s_j, 1],$$

$$\mathcal{B}_j^{xy'} = \mathcal{B}_{1-k}^{xy} \bigcap \mathcal{Y}[0, 1-s_j], \quad \mathcal{B}_j^{xy''} = \mathcal{B}_{1-k}^{xy} \bigcap \mathcal{Y}[1-s_j, 1].$$

Definimos a seguir uma lista de posições $p_{i,j}, q_{i,j}, p'_j, q'_j, p''_j$ e q''_j que serão usadas para *combinar* elementos de $\mathcal{A}_{k,i}^{xy}$ e $\mathcal{B}_{k,i}^{xy}$. Essas posições são definidas de tal maneira que a área ocupada pelos retângulos que serão empacotados nas placas (exceto talvez em um número constante delas) seja pelo menos $\frac{27}{56}$. Um algoritmo usando esta lista de posições será detalhado logo depois. Os conjuntos acima foram definidos considerando-se o plano xy. Podemos também definir conjuntos de forma análoga para os planos yz e zx, apenas trocando xy acima por yz e zx, respectivamente.

Definição 4.4.2. *Definição das posições $p_{i,j}, q_{i,j}, p'_j, q'_j, p''_j$ e q''_j usadas para combinar sublistas $\mathcal{A}_{k,i}^{xy}$ e $\mathcal{B}_{k,j}^{xy}$.*

Definimos estas posições apenas para os valores de $i \leq j$. O caso em que $i > j$ é simétrico (veja a Figura 4.11 para visualizar estas sublistas e a Figura 4.13 para visualizar um exemplo de empacotamento usando estas posições). Dada uma lista L, seja $A_i := \mathcal{A}_{k,j}^{xy}$ e $B''_i = \mathcal{B}_{k,j}^{xy}$.

- *Para combinar as listas A_i, $1 \leq i \leq k$, e B_j, $i \leq j \leq k$, considere*

  $p_{i,j} = \left[(0,0), \left(\frac{1}{2}, 0\right)\right]$ *e* $q_{i,j} = [(0, s_i)]$.

  Note que neste caso, temos uma garantia de área de pelo menos $\frac{1}{2}$.

- *Para combinar a lista $A_{[1-k]} = (A_1 \cup \ldots \cup A_k)$ com B_j, $k+1 \leq j \leq k+14$, consideramos duas fases. Dividimos $A_{[1-k]}$ em A' e A'' tomando $A' = \{b \in A_{[1-k]} : x(b) \leq 1 - s_j\}$ e $A'' = A_{[1-k]} \setminus A'$.*

  - ⋆ *Para combinar A' com B_j tome*

    $p'_j = [(s_j, 0)]$ *e*

    $q'_j = \left[(0,0), \left(0, \frac{1}{j-k+2}\right), \left(0, \frac{2}{j-k+2}\right), \ldots, \left(0, \frac{j-k+1}{j-k+2}\right)\right]$.

    Neste caso, temos uma garantia de área de pelo menos $\frac{13}{24}$. Este mínimo é atingido quando $j = k+1$.

  - ⋆ *Para combinar A'' com B_j tome*

    $p''_j = \left[(0,0), (\frac{1}{2}, 0)\right]$ *e*

    $q''_j = \left[\left(0, \frac{2}{3}\right), \left(0, \frac{2}{3} + \frac{1}{j-k+2}\right), \left(0, \frac{2}{3} + \frac{2}{j-k+2}\right), \ldots, \left(0, \frac{2}{3} + \left(\lfloor \frac{j-k+2}{3} \rfloor - 1\right) \frac{1}{j-k+2}\right)\right]$.

    Aqui obtemos uma garantia de área de pelo menos $\frac{27}{56}$. De fato, os valores s_j (que determinam A' e A''), $k+1 \leq j \leq k+14$, foram escolhidos de tal modo que os empacotamentos dos retângulos de $L \cap \wp_3 \setminus L_B$ possam ter boa garantia de área apenas usando algoritmos de estratégia NF. O valor $\frac{27}{56}$ é atingido quando $j = k+1$ (um retângulo de B_{k+1} com área $\frac{1}{8}$ e dois retângulos de A'', cada um com área de pelo menos $\frac{5}{28}$).

- *Para combinar as listas A_i, $k+1 \leq i \leq k+14$, e B_j, $i \leq j \leq k+14$, considere*

  $p_{i,j} = \left[(s_j, 0), \left(s_j + \frac{1}{i-k+2}, 0\right), \left(s_j + \frac{2}{i-k+2}, 0\right), \ldots, \left(s_j + (\lfloor (1-s_j) \cdot (i-k+2) \rfloor - 1) \frac{1}{i-k+2}, 0\right)\right]$ e

  $q_{i,j} = \left[(0,0), \left(0, \frac{1}{j-k+2}\right), \left(0, \frac{2}{j-k+2}\right), \ldots, \left(0, \frac{j-k+1}{j-k+2}\right)\right]$.

  Neste caso, obtemos também uma garantia de área de pelo menos $\frac{27}{56}$.

Estes conjuntos críticos serão muito importantes, sendo que iremos apresentar vários algoritmos fazendo uso destes conjuntos.

4.5 Caso Com Rotações

Nesta seção, apresentamos três algoritmos usando de rotações ortogonais. Dois para o caso geral, chamados $\text{BI}_{k,\epsilon}$ e OBI; o primeiro *off-line* com um limite de desempenho assintótico que pode

se tornar tão próximo de 2,6396055... quanto se queira e o segundo *on-line* com um limite de desempenho assintótico 3,25. O outro algoritmo é para o caso *on-line* de empacotamento em placas quadradas e tem um limite de desempenho assintótico que pode se tornar tão próximo de 2,6875 quanto se queira.

4.5.1 Algoritmo $\mathbf{BI}_{k,\epsilon}$

Apresentamos nesta seção o Algoritmo $\mathrm{BI}_{k,\epsilon}$ *(Bidimensional packing algorithm)* para o PEP^r. Antes de apresentarmos este algoritmo, vamos descrever dois procedimentos, o FFC (*First Fit Combine*) e o COMBINE-AB_k^{xy}, que são usados como subrotinas para empacotar os retângulos críticos de L.

Os parâmetros de entrada do Algoritmo FFC são uma lista de retângulos L, dois conjuntos de retângulos $\mathcal{T}_1$ e $\mathcal{T}_2$ e duas listas de coordenadas p_1 e p_2 associadas a esses conjuntos, cada lista p_i com n_i pontos. Este algoritmo gera um empacotamento, onde cada placa terá n_1+n_2 retângulos, exceto talvez a última placa. No contexto referente ao PEP^r, diremos que um retângulo r é do tipo $\mathcal{T}_i$ se $r \in \mathcal{T}_i$ ou $\rho(r) \in \mathcal{T}_i$.

Algoritmo FFC.

Entrada: $(L, \mathcal{T}_1, \mathcal{T}_2, [p_1], [p_2])$

Saída: Empacotamento $\mathcal{P}$ onde, ou todos os retângulos do tipo $\mathcal{T}_1$ ou todos os retângulos do tipo $\mathcal{T}_2$ são totalmente empacotados.

1 Enquanto todos os retângulos de um dos tipos não forem totalmente empacotados, faça

1.1 Tente empacotar o próximo retângulo r' do tipo $\mathcal{T}_i$ (sem perda de generalidade, considere $r' \in \mathcal{T}_i$, caso contrário, gire r' previamente) para algum $i \in \{1, 2\}$ em alguma posição de p_i, na última placa sendo empacotada desde que nenhum retângulo tenha sido empacotado nesta posição desta placa. Caso não exista, considere uma nova placa (ainda vazia), e empacote o retângulo r' na primeira posição de p_1.

2 Seja L' o conjunto de retângulos empacotados em $\mathcal{P}$.

3 Retorne $(\mathcal{P}, L')$.

Fim algoritmo.

Usando o Algoritmo FFC, podemos definir um algoritmo para empacotar (conjuntamente) retângulos de $\mathcal{A}_k^{xy}$ com retângulos de $\mathcal{B}_k^{xy}$. Isto será feito pelo Algoritmo COMBINE-AB_k^{xy}. Este algoritmo será bastante usado como subrotina em outros algoritmos para combinar conjuntos críticos.

Na descrição deste e de outros algoritmos, usamos uma função denominada *Atualize* que tem como parâmetro de entrada listas $L_1, \ldots, L_m$. Esta função remove das listas L_i, $1 \leq i \leq m$, todas as caixas que já foram empacotadas até o momento de sua chamada.

O Algoritmo COMBINE-AB$_k^{xy}$ será descrito de forma genérica para ser usado tanto no PEP^r como nos problemas PET, PET^r e PEC^r. Primeiramente, vamos dar uma idéia de como é feita a combinação de itens de tipos $\mathcal{A}_k^{xy}$ e $\mathcal{B}_k^{xy}$ pelo Algoritmo COMBINE-AB$_k^{xy}$.

Este algoritmo é chamado com cinco parâmetros: $(L, type, \mathcal{T}_A, \mathcal{T}_B, \text{COMBINE})$.

A função *type* indica quais itens de L são de certo tipo; esta função poderá ser uma entre ftype, rtype e xy-type.

Os conjuntos $\mathcal{T}_A$ e $\mathcal{T}_B$ servem para restringir os itens de entrada.

A rotina COMBINE é chamada para fazer a combinação de partes do empacotamento retornado pelo Algoritmo COMBINE-AB$_k^{xy}$.

O Algoritmo COMBINE-AB$_k^{xy}$ retorna um empacotamento que contém todos os itens de $type(L, \mathcal{T}_A \cap \mathcal{A}_k^{xy})$ ou todos os itens de $type(L, \mathcal{T}_B \cap \mathcal{B}_k^{xy})$.

No passo 3 do algoritmo, é feita a combinação entre os itens de $type(L, \mathcal{T}_A \cap \mathcal{A}_{k,i}^{xy})$ e $type(L, \mathcal{T}_B \cap \mathcal{B}_{k,j}^{xy})$, $1 \leq i, j \leq k$. Começa-se fazendo a combinação entre os itens de $type(L, \mathcal{T}_A \cap \mathcal{A}_{k,1}^{xy})$ e $type(L, \mathcal{T}_B \cap \mathcal{B}_{k,1}^{xy})$. Se todos os itens de $type(L, \mathcal{T}_A \cap \mathcal{A}_{k,i}^{xy})$ forem empacotados, então faz-se a combinação entre os itens de $type(L, \mathcal{T}_A \cap \mathcal{A}_{k,i+1}^{xy})$ e $type(L, \mathcal{T}_B \cap \mathcal{B}_{k,j}^{xy})$; caso contrário, de $type(L, \mathcal{T}_A \cap \mathcal{A}_{k,i}^{xy})$ e $type(L, \mathcal{T}_B \cap \mathcal{B}_{k,j+1}^{xy})$.

Depois de terminado o passo 3, temos que ou todos os itens de $type(L, \mathcal{T}_A \cap \mathcal{A}_{[1-k]}^{xy})$ foram empacotados (vai para o passo 4) ou $type(L, \mathcal{T}_B \cap \mathcal{B}_{[1-k]}^{xy})$ foram empacotados (vai para o passo 5). Os passos 4 e 5 são simétricos, assim, considere que todos os itens de $type(L, \mathcal{T}_B \cap \mathcal{B}_{[1-k]}^{xy})$ foram empacotados. Neste caso, os itens de $type(\mathcal{T}_A \cap \mathcal{A}_{[1-k]}^{xy})$ são combinados com os itens de $type(\mathcal{T}_B \cap \mathcal{B}_{k,j}^{xy})$, $k+1 \leq j \leq k+14$. Para isto, o conjunto $type(L, \mathcal{T}_A \cap \mathcal{A}_{[1-k]}^{xy})$ é dividido em duas partes: o conjunto $type(L, \mathcal{T}_A \cap \mathcal{A}_j^{xy'})$ e $type(L, \mathcal{T}_A \cap \mathcal{A}_j^{xy''})$ e só depois os itens de cada um destes tipos são combinados com os de $type(L, \mathcal{T}_B \cap \mathcal{B}_{k,j}^{xy})$. Ao fim do passo 4, temos que ou todos os itens de $type(L, \mathcal{T}_A \cap \mathcal{A}_{[1-k]}^{xy})$ ou todos os itens de $type(L, \mathcal{T}_B \cap \mathcal{B}_k^{xy})$ foram empacotados. Caso ainda haja itens de $type(L, \mathcal{T}_B \cap \mathcal{B}_k^{xy})$, estes são empacotados com os itens restantes de $type(L, \mathcal{T}_A \cap \mathcal{A}_k^{xy})$ da mesma forma como feito no passo 3.

Algoritmo COMBINE-AB$_k^{xy}$ $(L, type, \mathcal{T}_A, \mathcal{T}_B, \text{COMBINE})$

Entrada: Listas de itens L e tipos $\mathcal{A}_k^{xy}$ e $\mathcal{B}_k^{xy}$.

Saída: Empacotamento parcial $\mathcal{P}_{AB}$ de L, tal que ou todos os itens de $type(L, \mathcal{T}_A)$ ou todos os itens de $type(L, \mathcal{T}_B)$ são empacotados.

1 Sejam $p_{i,j}, q_{i,j}$, $(1 \leq i, j \leq k+14)$, e p'_j, p''_j, q'_j, q''_j, $(k+1 \leq j \leq k+14)$, as posições dadas na Definição 4.4.2.

2 $i \leftarrow 1$;
$j \leftarrow 1$;
$\mathcal{P}_{AB} \leftarrow \emptyset$.

3 Enquanto ($i \leq k$ e $j \leq k$) faça

3.1 $\mathcal{P}' \leftarrow$ COMBINE$(L, \mathcal{T}_A \cap \mathcal{A}^{xy}_{k,i}, \mathcal{T}_B \cap \mathcal{B}^{xy}_{k,j}, p_{ij}, q_{ij})$.

3.2 Atualize(L).

3.3 $\mathcal{P}_{AB} \leftarrow \mathcal{P}_{AB} \cup \mathcal{P}'$.

3.4 Se $type(L, \mathcal{T}_A \cap \mathcal{A}^{xy}_{k,i}) = \emptyset$ então $i \leftarrow i+1$ senão $j \leftarrow j+1$.

4 Se $type(L, \mathcal{T}_B \cap \mathcal{B}^{xy}_{[1-k]}) = \emptyset$

4.1 Enquanto ($j \leq k+14$ e $type(L, \mathcal{T}_A \cap \mathcal{A}^{xy}_{k,i}) \neq \emptyset$) faça

4.1.1 $\mathcal{P}' \leftarrow$ COMBINE$(L, \mathcal{T}_A \cap \mathcal{A}^{xy'}_{j}, \mathcal{T}_B \cap \mathcal{B}^{xy}_{k,j}, p'_j, q'_j)$;

4.1.2 $\mathcal{P}'' \leftarrow$ COMBINE$(L, \mathcal{T}_A \cap \mathcal{A}^{xy''}_{j}, \mathcal{T}_B \cap \mathcal{B}^{xy}_{k,j}, p''_j, q''_j)$;

4.1.3 $\mathcal{P}_{AB} \leftarrow \mathcal{P}_{AB} \| \mathcal{P}' \| \mathcal{P}''$.

4.1.4 Atualize(L).

4.1.5 Se $type(L, \mathcal{T}_B \cap \mathcal{B}^{xy}_{k,j}) = \emptyset$ então $j \leftarrow j+1$.

4.2 $i \leftarrow k+1$.

5 Senão (Todos os itens de $type(L, \mathcal{T}_A \cap \mathcal{A}^{xy}_{[1-k]})$ foram empacotados) execute os passos simétricos aos indicados no passo 4.

6 Enquanto ($i \leq k+14$ e $j \leq k+14$) faça

6.1 $\mathcal{P}' \leftarrow$ COMBINE$(L, \mathcal{T}_A \cap \mathcal{A}^{xy}_{k,i}, \mathcal{T}_B \cap \mathcal{B}^{xy}_{k,j}, p_{ij}, q_{ij})$.

6.2 Atualize(L).

6.3 $\mathcal{P}_{AB} \leftarrow \mathcal{P}_{AB} \cup \mathcal{P}'$.

6.4 Se $type(L, \mathcal{T}_A \cap \mathcal{A}^{xy}_{k,i}) = \emptyset$ então $i \leftarrow i+1$ senão $j \leftarrow j+1$.

7 Retorne $\mathcal{P}_{AB}$.

Fim algoritmo.

Na definição deste algoritmo usamos o plano xy, mas podemos definir algoritmos análogos usando outros planos. Assim poderemos usar este mesmo algoritmo como COMBINE-AB$^{yz}_k$ e COMBINE-AB$^{zx}_k$ para usar os planos yz e zx, respectivamente.

As idéias básicas do Algoritmo $\mathrm{BI}_{k,\epsilon}(L)$ são apresentadas a seguir. Procuramos aqui ilustrar o conceito de *conjuntos críticos*, pois estes são fundamentais no entendimento deste e de outros algoritmos descritos nesta tese.

Este algoritmo divide a lista de entrada L em sublistas e aplica um algoritmo apropriado para cada uma (ou uma combinação) destas sublistas. O empacotamento final é obtido como uma concatenação destes empacotamentos.

Apenas para apresentar a idéias por trás deste algoritmo, considere o PEP$(1,1)$. Considere os retângulos de L divididos em quatro partes: P_1, P_2, P_3 e P_4, onde $P_i := L \cap \wp_i$, $i = 1, \ldots, 4$ (veja a Figura 4.8).

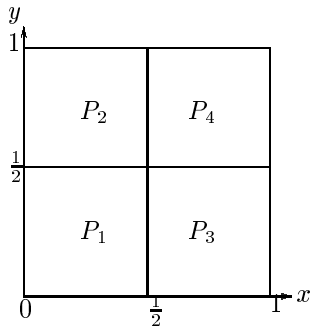


Figura 4.8: Subdivisão de L em partes P_1, P_2, P_3 e P_4.

Vamos analisar a garantia de área que podemos assegurar para um empacotamento dos itens em cada uma das partes P_i, $i = 1, \ldots, 4$. Para isto, seja L'_i, $i = 1, \ldots, 4$, subconjuntos dos itens nessas partes, como apresentado na Figura 4.9.

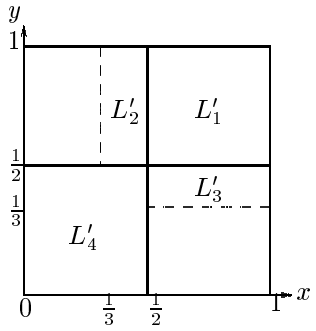


Figura 4.9: Listas $L'_1, \ldots, L'_4$.

Para a lista L'_1, o melhor que podemos garantir é $\frac{1}{4}$. Para a lista L'_2, o melhor que podemos fazer é empacotar dois retângulos por placa. Assim, só podemos ter uma garantia de $\frac{1}{3}$ para L'_2. Por simetria, só podemos ter uma garantia de $\frac{1}{3}$ para a lista L'_3. Para a lista L'_4, usando o Algoritmo BI_2, podemos ter uma garantia de $\frac{4}{9}$. Ou seja, considerando esses empacotamentos,

temos as seguintes garantias de área: $\frac{1}{4}$ para a parte P_4, $\frac{1}{3}$ para as partes P_2 e P_3 e $\frac{4}{9}$ para a parte P_1 (veja Figura 4.10).

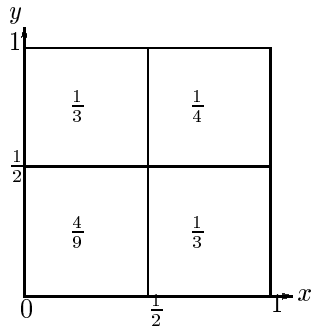


Figura 4.10: Garantia de área para as partes P_1, P_2, P_3 e P_4.

Assim, com respeito à garantia de área, vamos classificar os empacotamentos da parte P_1 como sendo *bom*, das partes P_2 e P_3 como *regular* e da parte P_4 como *ruim*. Chamaremos um empacotamento de uma sublista como sendo *bom* se ele tem uma garantia de área perto daquela da parte P_1. A idéia do Algoritmo $\mathrm{BI}_{k,\epsilon}$ é refinar a subdivisão das partes de modo que as sublistas obtidas forneçam uma área combinada melhor ou uma garantia de área melhor. Para isto, vamos chamar de *retângulos críticos* alguns dos retângulos que geram empacotamentos com pouca garantia de área.

Primeiramente é feita a combinação dos retângulos de tipos $\mathcal{A}_k^{xy}$ e $\mathcal{B}_k^{xy}$ usando-se o Algoritmo COMBINE-AB_k^{xy}. O empacotamento gerado a partir desta combinação contém ou todos os retângulos do tipo $\mathcal{A}_k^{xy}$ ou todos os retângulos do tipo $\mathcal{B}_k^{xy}$. Vamos chamar de $\mathcal{P}_{AB}$ este empacotamento combinado e L_{AB} o conjunto de retângulos empacotados em $\mathcal{P}_{AB}$. Uma informação importante sobre o empacotamento $\mathcal{P}_{AB}$ é que este tem uma boa garantia de área.

Suponhamos que todos os retângulos do tipo $\mathcal{B}_k^{xy}$ foram empacotados. Considerando os retângulos restantes, é possível considerar duas partes: a parte $(P_2 \cup P_4) \setminus L_{AB}$ e a parte $(P_1 \cup P_3) \setminus L_{AB}$. Dividindo os retângulos de $(P_1 \cup P_3) \setminus L_{AB}$ em sublistas $L_1, \ldots, L_{25}$ (veja Figura 4.12), e aplicando algoritmos do tipo NF, é possível ter uma boa garantia de área para os empacotamentos gerados desta forma, ou seja, próxima de $\frac{4}{9}$. Observe que é possível ter um empacotamento bem próximo do ótimo para os retângulos de $(P_2 \cup P_4) \setminus L_{AB}$. Como dois retângulos de $(P_2 \cup P_4) \setminus L_{AB}$ não podem ser empacotados lado a lado no eixo y, podemos considerar o empacotamento destes retângulos como uma instância para o problema de empacotamento unidimensional. Assim, usando o Algoritmo VL_ϵ sobre esta lista, podemos ter um empacotamento assintoticamente tão próximo do ótimo quanto se queira.

Por fim, considerando todos estes empacotamentos, podemos considerar o empacotamento final dividido em dois tipos: um com garantia de área próxima de $\frac{1}{4}$, mas assintoticamente

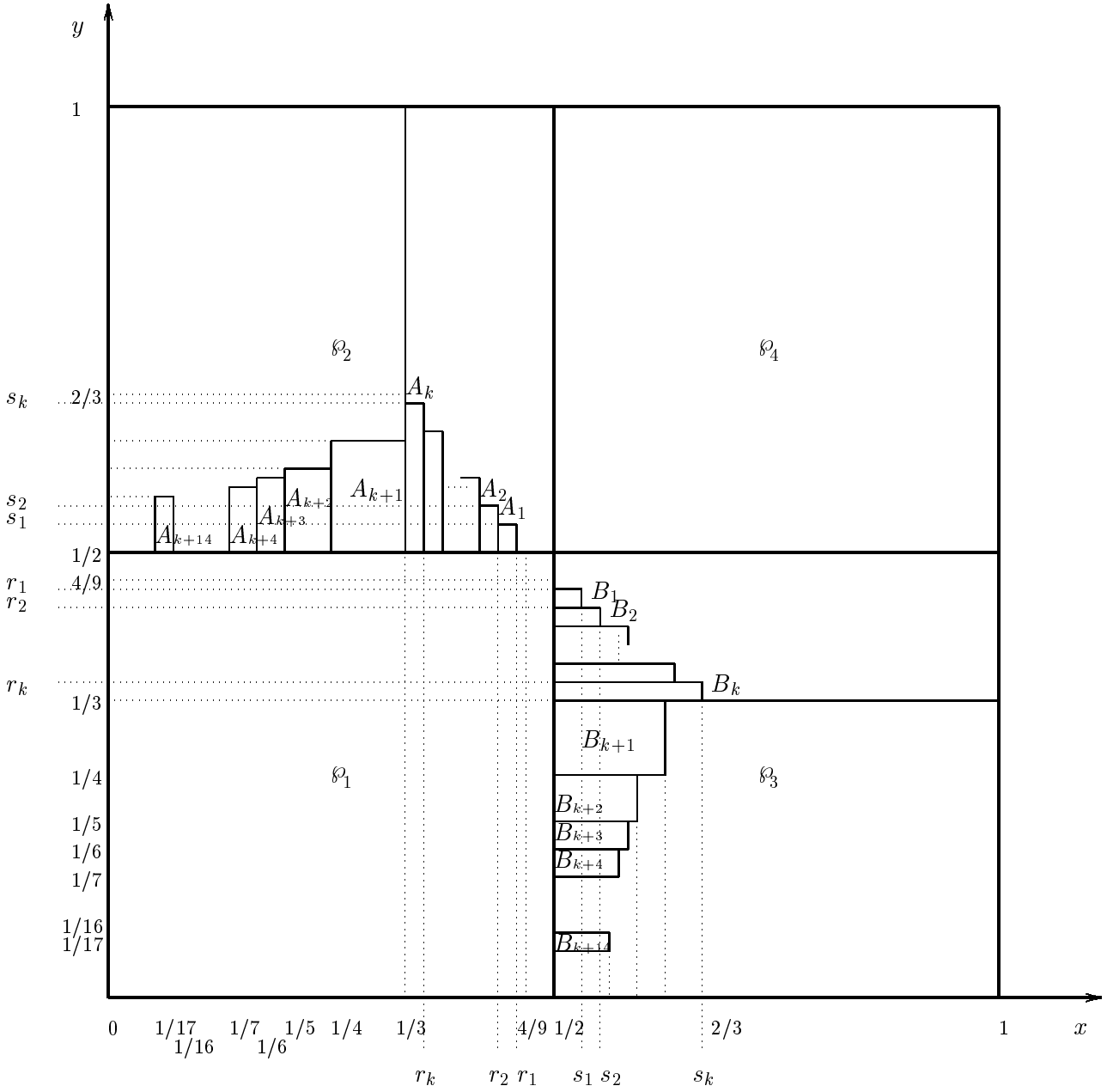


Figura 4.11: Sublistas A_i e B_j.

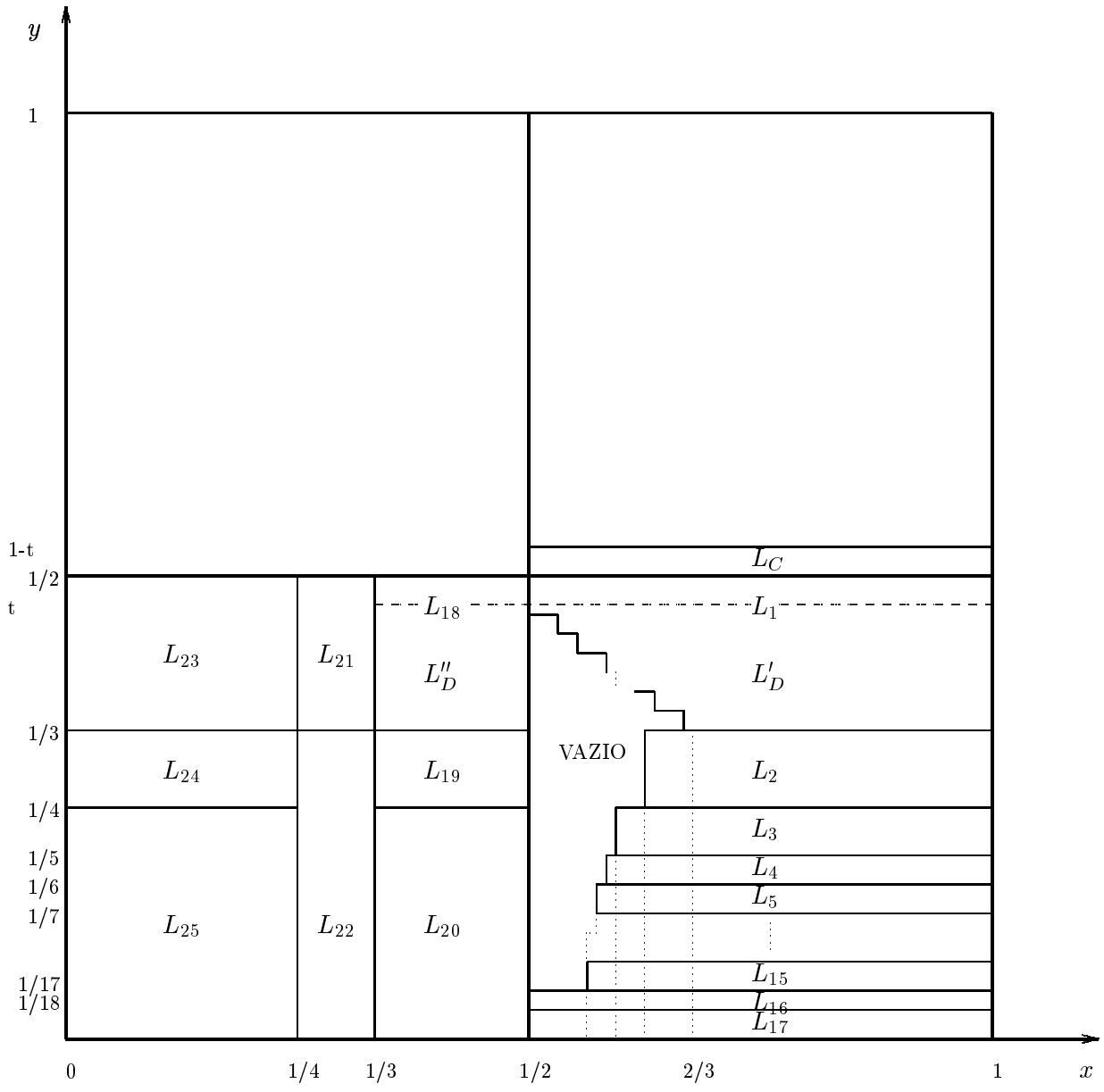


Figura 4.12: Sublista após o empacotamento da lista $L_B = (B_1 \cup \ldots \cup B_{k+14})$.

bem próximo do ótimo, e outro com garantia de área próxima de $\frac{4}{9}$. Isto nos dá um limite de desempenho assintótico próximo de 2,6875 para este algoritmo. Mas não vamos parar por aqui. É possível combinar mais sublistas críticas, de forma a melhorar um pouco mais este limite.

Considere novamente as listas $L_1, \ldots, L_{25}$ indicadas na Figura 4.12. Os empacotamentos das listas L_1 e L_{18} gerados pelo Algoritmo NF, têm garantia próxima de $\frac{4}{9}$, mas para as listas $L_2, \ldots, L_{17}, L_{19}, \ldots, L_{25}$ a estratégia NF gera empacotamentos com garantia de área melhor que $\frac{4}{9}$. Assim, definimos como $L_D := L'_D \cup L''_D$ os retângulos críticos de $L_1 \cup L_{18}$. Agora observe que os retângulos de $(P_2 \cup P_4) \setminus L_{AB}$ para os quais são gerados empacotamentos com garantia de área próxima de $\frac{1}{4}$ são os que estão P_4. Assim, definimos como L_C os retângulos críticos de P_4 (veja a Figura 4.12). Estes novos conjuntos críticos (L_C e L_D) foram definidos de forma que no empacotamento combinado destes um deles é totalmente empacotado. Chamaremos este empacotamento combinado e o conjunto de retângulos empacotados de $\mathcal{P}_{CD}$ e L_{CD}, respectivamente.

Garantimos que os empacotamentos $\mathcal{P}_{AB}$ e $\mathcal{P}_{CD}$ têm garantia de área melhor que $\frac{4}{9}$. Dependendo de qual conjunto é totalmente empacotado em $\mathcal{P}_{CD}$, podemos melhorar ou a garantia de área de $\frac{1}{4}$, do empacotamento de $(P_2 \cup P_4) \setminus (L_{AB} \cup L_{CD})$, ou a garantia de área de $\frac{4}{9}$, do empacotamento de $(P_1 \cup P_3) \setminus (L_{AB} \cup L_{CD})$. Desta forma, é possível obter um algoritmo para o PEP$(1,1)$ com limite de desempenho assintótico próximo de 2,64.

Quando consideramos o PEP$^r(a,b)$, alguns retângulos são girados de modo que ao aplicar o Algoritmo VL$_\epsilon$ sobre os retângulos de $(P_2 \cup P_4) \setminus (L_{AB} \cup L_{CD})$ ainda possamos garantir um empacotamento próximo do ótimo.

Diversos algoritmos que desenvolvemos para os demais problemas seguem esta mesma abordagem. Assim é importante entender este algoritmo.

A seguir, apresentamos o Algoritmo BI$_{k,\epsilon}$.

Algoritmo BI$_{k,\epsilon}(L)$

Entrada: Lista de retângulos L

Saída: Empacotamento $\mathcal{P}$ de L em placas $R = (a,b)$, permitindo rotações ortogonais.

1 Gire todos os retângulos de b que estão em $\wp_4$ de tal forma que $\rho(b) \in \wp_1 \cup \wp_2 \cup \wp_3$.
/* Seja $R_1 \leftarrow \{b \in L \cap \wp_4 : \rho(b) \in \wp_1 \cup \wp_2 \cup \wp_3\}$; $L \leftarrow (L \setminus R_1) \bigcup \rho(R_1)$. */

2 $t \leftarrow 0{,}4574271$.

3 $\mathcal{P}_{AB} \leftarrow$ COMBINE$-$AB$^{xy}(L, xy\text{-type}, \mathcal{C}_1, \mathcal{C}_1, \text{FFC})$.
Atualize(L).

4 Se todas os retângulos de xy-type$(L, \mathcal{B}_k^{xy})$ foram empacotadas então

4.1 Gire os retângulos de $L \cap \wp_2$ que se encaixam em $\wp_1 \cup \wp_3$.

4.2 Gire os retângulos de $L \cap (\wp_2 \cup \wp_4)$ tal que se $b \in L \cap (\wp_2 \cup \wp_4)$ então $x(b) \leq y(b)$ ou $\rho(b) \notin \mathcal{C}_1$.

4.3 Subdivida a lista L em $L_1, \ldots, L_{25}$ como segue (veja a Figura 4.12).

$L_i \leftarrow L \bigcap \mathcal{C}\left[\frac{1}{2}, 1\ ;\ \frac{1}{i+2}, \frac{1}{i+1}\right]$, para $i = 1, \ldots, 16$ $\quad L_{17} \leftarrow L \bigcap \mathcal{C}\left[\frac{1}{2}, 1\ ;\ 0, \frac{1}{18}\right]$,
$L_{18} \leftarrow L \bigcap \mathcal{C}\left[\frac{1}{3}, \frac{1}{2}\ ;\ \frac{1}{3}, \frac{1}{2}\right]$, $\quad L_{19} \leftarrow L \bigcap \mathcal{C}\left[\frac{1}{3}, \frac{1}{2}\ ;\ \frac{1}{4}, \frac{1}{3}\right]$,
$L_{20} \leftarrow L \bigcap \mathcal{C}\left[\frac{1}{3}, \frac{1}{2}\ ;\ 0, \frac{1}{4}\right]$, $\quad L_{21} \leftarrow L \bigcap \mathcal{C}\left[\frac{1}{4}, \frac{1}{3}\ ;\ \frac{1}{3}, \frac{1}{2}\right]$,
$L_{22} \leftarrow L \bigcap \mathcal{C}\left[\frac{1}{4}, \frac{1}{3}\ ;\ 0, \frac{1}{3}\right]$, $\quad L_{23} \leftarrow L \bigcap \mathcal{C}\left[0, \frac{1}{4}\ ;\ \frac{1}{3}, \frac{1}{2}\right]$,
$L_{24} \leftarrow L \bigcap \mathcal{C}\left[0, \frac{1}{4}\ ;\ \frac{1}{4}, \frac{1}{3}\right]$, $\quad L_{25} \leftarrow L \bigcap \mathcal{C}_4$,
$L_C \leftarrow L \bigcap \mathcal{C}\left[\frac{1}{2}, 1\ ;\ \frac{1}{2}, 1-t\right]$, $\quad L'_D \leftarrow L_1 \bigcap \mathcal{C}\left[0, t\ ;\ 0, 1\right]$,
$L''_D \leftarrow L_{18} \bigcap \mathcal{C}\left[0, t\ ;\ 0, 1\right]$, $\quad L_D \leftarrow L'_D \bigcup L''_D$.

4.4 Gere empacotamento $\mathcal{P}_{CD}$ como segue.

$(\mathcal{P}_{CD'}, L_{CD'}) \leftarrow \mathrm{FFC}(L, L_C, [(0,0)], L'_D, [(0, 1-t)])$;
$(\mathcal{P}_{CD''}, L_{CD''}) \leftarrow \mathrm{FFC}(L, L_C \setminus L_{CD'}, [(0,0)], L''_D, [(0, 1-t), (\frac{1}{2}, 1-t)])$;
$\mathcal{P}_{CD} \leftarrow \mathcal{P}_{CD'} \| \mathcal{P}_{CD''}$;
$L_{CD} \leftarrow L_{CD'} \bigcup L_{CD''}$;
$L_1 \leftarrow L_1 \setminus L_{CD}$;
$L_{18} \leftarrow L_{18} \setminus L_{CD}$.

4.5 Gere empacotamentos $\mathcal{P}_1, \ldots, \mathcal{P}_{25}$ como segue.

$\mathcal{P}_i \leftarrow \mathrm{NFDH}^{(p)}(L_i)$ para $i = 1, \ldots, 22$;
$\mathcal{P}_i \leftarrow \mathrm{NFDH}^{(p)}(L_i)$ para $i = 23, 24$;
$\mathcal{P}_{25} \leftarrow \mathrm{BI}_4(L_{25}, 4)$.

4.6 Atualize L removendo os retângulos empacotadas. Note que $L \subseteq \wp_2 \cup \wp_4$.

4.7 Considere cada retângulo de L como uma barra de comprimento $x(r)$ e cada placa como uma barra de comprimento a.
Aplique o Algoritmo VL_ϵ em L; seja $\mathcal{P}_{\mathrm{VL}_\epsilon}$ este empacotamento.
Seja $\mathcal{P}_{FFD}$ o empacotamento $\mathrm{FFD}(L, \cap\mathcal{X}[0, \frac{1}{3}]) \| \mathrm{FFD}(L, \cap\mathcal{X}[\frac{1}{3}, \frac{1}{2}]) \| \mathrm{FFD}(L, \cap\mathcal{X}[\frac{1}{2}, 1])$.
Seja $\mathcal{P}_{UNI}$ um menor empacotamento em $\{\mathcal{P}_{\mathrm{VL}_\epsilon}, \mathcal{P}_{FFD}\}$.

4.8 $\mathcal{P}_{aux} \leftarrow \mathcal{P}_{AB} \| \mathcal{P}_{CD} \| \mathcal{P}_1 \| \ldots \| \mathcal{P}_{25}$;

4.9 $\mathcal{P} \leftarrow \mathcal{P}_{UNI} \| \mathcal{P}_{aux}$.

5 Se todos os retângulos de xy-type$(L, \mathcal{A}_k^{xy})$ foram empacotados então gere um empacotamento $\mathcal{P}$ de L como no passo 4 (de forma simétrica).

6 Retorne $\mathcal{P}$.

Fim algoritmo.

O próximo teorema nos dá um limite de desempenho assintótico para o Algoritmo $\mathrm{BI}_{k,\epsilon}$ quando $k \to \infty$ e $\epsilon \to 0$.

Teorema 4.5.1. *Para qualquer instância L do* PEP^r*, temos que*

$$\mathrm{BI}_{k,\epsilon}(L) \leq \alpha_{k,\epsilon} \cdot \mathrm{OPT}(L) + \mathcal{O}\left(k + \frac{1}{\epsilon}\right),$$

onde $\alpha_{k,\epsilon} \to 2{,}639605\ldots$ à medida que $k \to \infty$ e $\epsilon \to 0$.

Prova. Apresentamos a prova para o caso onde todos os retângulos de xy-type$(L, \mathcal{B}_k^{xy})$ foram empacotados no passo 4. A prova para o outro caso (passo 5) é análoga. Esta prova é dividida em 2 casos, de acordo com o passo 4.4 ($L_C \subseteq L_{CD}$).

Cada um dos empacotamento $\mathcal{P}_i$, $i \in \{1, \ldots, 25\} \setminus \{1, 18\}$, têm uma garantia de área que é pelo menos $\frac{17}{36} \cdot a \cdot b$, este mínimo sendo atingido quando $i \in \{16, 17\}$. Assim, aplicando o Lema 4.2.1 e o Lema 4.4.3 podemos concluir que

$$\#(\mathcal{P}_i) \leq \frac{36}{17}\frac{S(L_i)}{ab} + 1, \quad \text{para} \quad i \in \{1, \ldots, 25\} \setminus \{1, 18\}. \tag{4.27}$$

Agora, para cada um dos empacotamentos $\mathcal{Q} \in \left\{\mathcal{P}_{i,j}, \tilde{\mathcal{P}'_{i,j}}, \tilde{\mathcal{P}''_{i,j}}\right\}$ que são usados para gerar o empacotamento $\mathcal{P}_{AB}$ no final do passo 5 e construídos pelo Algoritmo COMBINE-AB$_k^{xy}$, temos $\#(\mathcal{Q}) \leq \frac{56}{27}\frac{S(\mathcal{Q})}{ab} + 1$. Para ver isto, note que para cada um destes empacotamentos, a área garantida em cada placa, exceto talvez na última, é de pelo menos $\frac{27}{56}ab$. Como existe um máximo de $(2k-1) + 28 + 14 = 2k + 41$ empacotamentos gerados de L_A e L_B, podemos ver que $\#(\mathcal{P}_{AB}) \leq \frac{56}{27}\frac{S(L_{AB})}{ab} + (2k + 41)$. Assim, a seguinte inequação vale:

$$\#(\mathcal{P}_{AB}) \leq \frac{36}{17}\frac{S(L_{AB})}{ab} + (2k + 41). \tag{4.28}$$

Pelo Lema 4.4.14, temos que quando $k \to \infty$ o valor de $r_1^{(k)}$ tende a $\frac{4}{9}$, $r_1 = r_1^{(k)} < \frac{4}{9}$ (veja a Definição 4.4.1).

Para os empacotamentos $\mathcal{P}_{CD'}$ e $\mathcal{P}_{CD''}$ (no passo 4.4), a área combinada é pelo menos $(\frac{1}{4} + \frac{r_1}{2})ab$, com isso, segue do Lema 5.3.8 que

$$\#(\mathcal{P}_{CD}) \leq \frac{1}{\left(\frac{1}{4} + \frac{r_1}{2}\right)}\frac{S(L_{CD})}{ab} + 2. \tag{4.29}$$

Vamos agora analisar os dois casos possíveis (veja passo 5.4).

Caso 1. $L_C \subseteq L_{CD}$.

Para os empacotamentos $\mathcal{P}_1$ e $\mathcal{P}_{18}$ temos:

$$\#(\mathcal{P}_1) \leq \frac{1}{r_1}\frac{S(L_1)}{ab} + 1, \tag{4.30}$$

$$\#(\mathcal{P}_{18}) \leq \frac{1}{\frac{4}{9}}\frac{S(L_{18})}{ab} + 1. \tag{4.31}$$

Pelo Teorema 2.4.3,

$$\#(\mathcal{P}_{UNI}) \leq \#(\mathcal{P}_{\mathrm{VL}_\epsilon}) \leq (1+\epsilon)\cdot \mathrm{OPT}(L_{UNI}) + \beta_\epsilon. \tag{4.32}$$

Note também que com o Algoritmo FFD aplicado à lista L_{UNI} obtemos a seguinte desigualdade

$$\#(\mathcal{P}_{UNI}) \leq \#(\mathcal{P}_{FFD}) \leq \frac{1}{(1-t)\cdot\frac{1}{2}} \cdot \frac{S(L_{UNI})}{ab} + 3. \tag{4.33}$$

Agora, para o empacotamento $\mathcal{P}_{aux} = \mathcal{P}_{AB}\|\mathcal{P}_1\|\ldots\|\mathcal{P}_{25}$, usando as inequações (4.28),...,(4.31) e o fato que $r_1 = \min\left\{\frac{17}{36}, r_1, \frac{1}{4}+\frac{r_1}{2}, \frac{4}{9}\right\}$, obtemos

$$\#(\mathcal{P}_{aux}) \leq \frac{1}{r_1}\frac{S(L_{aux})}{ab} + (2k+68), \tag{4.34}$$

onde L_{aux} denota o conjunto de retângulos no empacotamento $\mathcal{P}_{aux}$.

Sejam

$$\mathcal{H}_1 := \#(\mathcal{P}_{UNI}) - \beta_\epsilon, \tag{4.35}$$

$$\mathcal{H}_2 := \#(\mathcal{P}_{aux}) - (2k+68). \tag{4.36}$$

Da inequação (4.32) temos $\mathcal{H}_1 \leq (1+\epsilon)\cdot \mathrm{OPT}(L_{UNI})$, e portanto,

$$\mathrm{OPT}(L_{UNI}) \geq \frac{\mathcal{H}_1}{(1+\epsilon)}.$$

Assim,

$$\mathrm{OPT}(L) \geq \mathrm{OPT}(L_{UNI}) \geq \frac{\mathcal{H}_1}{(1+\epsilon)}. \tag{4.37}$$

Note que das inequações (4.34) e (4.36) podemos concluir que

$$\frac{S(L_{aux})}{ab} \geq r_1 \cdot \mathcal{H}_2. \tag{4.38}$$

Substituindo a definição de $\mathcal{H}_1$ em (4.33), temos

$$\frac{S(L_{UNI})}{ab} \geq \frac{(1-t)}{2}\mathcal{H}_1. \tag{4.39}$$

Como $S(L) = S(L_{aux}) + S(L_{UNI})$, usando (4.38) e (4.39) obtemos

$$\frac{S(L)}{ab} \geq r_1 \cdot \mathcal{H}_2 + \frac{(1-t)}{2}\mathcal{H}_1.$$

Assim,

$$\mathrm{OPT}(L) \geq \frac{S(L)}{ab} \geq r_1 \cdot \mathcal{H}_2 + \frac{(1-t)}{2}\mathcal{H}_1.$$

Combinando (4.37) e a inequação acima, segue que

$$\mathrm{OPT}(L) \geq \max\left\{\frac{1}{1+\epsilon}\mathcal{H}_1, \frac{1-t}{2}\mathcal{H}_1 + r_1 \cdot \mathcal{H}_2\right\}.$$

Como $\#(\mathcal{P}) = \#(\mathcal{P}_{aux}) + \#(\mathcal{P}_{UNI})$; usando (4.35) e (4.36), temos

$$\begin{aligned}\#(\mathcal{P}) &= (\mathcal{H}_2 + (2k+68) + \mathcal{H}_1 + \beta_\epsilon) \\ &= \mathcal{H}_1 + \mathcal{H}_2 + \left(2k + \beta'_\epsilon\right),\end{aligned}$$

onde $\beta'_\epsilon = \beta_\epsilon + 68$. Assim,

$$\mathrm{BI}_{k,\epsilon}(L) \leq \alpha'_k(r_1) \cdot \mathrm{OPT}(L) + \left(2k + \beta'_\epsilon\right),$$

onde $\alpha'_k(r_1) = \left[\frac{1}{r_1} - \frac{(1-t)}{2r_1} + (1+\epsilon)\right]$. Para provar isto, basta notar que $\frac{\mathcal{H}_1+\mathcal{H}_2}{\max\left\{\frac{1}{1+\epsilon}\mathcal{H}_1, \frac{(1-t)}{2}\mathcal{H}_1 + r_1\cdot\mathcal{H}_2\right\}} \leq \alpha'_k(r_1)$, analisando-se os dois casos onde o denominador é máximo.

Caso 2. $L_D \subseteq L_{CD}$.

Como os retângulos de L'_D foram empacotados em $\mathcal{P}_{CD}$, temos uma garantia de área de pelo menos t para as placas de $\mathcal{P}_1$, exceto talvez para a última. O mesmo pode ser verificado para o empacotamento $\mathcal{P}_{18}$. Com isto, vale que

$$\#(\mathcal{P}_i) \leq \frac{1}{t} \cdot \frac{S(L_i)}{ab} + 1 \quad \text{para} \quad i \in \{1, 18\}.$$

Pelo Teorema 2.4.3.

$$\#(\mathcal{P}_{UNI}) \leq (1+\epsilon) \cdot \mathrm{OPT}(L_{UNI}) + \beta_\epsilon.$$

Como o empacotamento $\mathcal{P}_{FFD}$ tem garantia de área de pelo menos $\frac{1}{4}$ em todas as placas, exceto talvez em três delas, e $\#(\mathcal{P}_{UNI}) \leq \#(\mathcal{P}_{FFD})$, temos

$$\#(\mathcal{P}_{UNI}) \leq \frac{1}{1/4}\frac{S(L_{UNI})}{ab} + 3.$$

Como $t = \min\left\{t, \frac{1}{4} + \frac{r_1}{2}, \frac{17}{36}\right\}$, temos

$$\#(\mathcal{P}_{aux}) \leq \frac{1}{t}\frac{S(L_{aux})}{ab} + (2k+68).$$

Sejam

$$\begin{aligned}\mathcal{H}_1 &:= \#(\mathcal{P}_{UNI}) - \beta_\epsilon,\\ \mathcal{H}_2 &:= \#(\mathcal{P}_{aux}) - (2k+68).\end{aligned}$$

Então

$$\mathrm{OPT}(L) \geq \frac{1}{1+\epsilon}\mathcal{H}_1.$$

Por outro lado,

$$\begin{aligned}\frac{S(L_{aux})}{ab} &\geq t\cdot\mathcal{H}_2 \text{ e}\\ \frac{S(L_{UNI})}{ab} &\geq \frac{1}{4}\mathcal{H}_1,\end{aligned}$$

e portanto,

$$\mathrm{OPT}(L) \geq \frac{S(L)}{ab} \geq t\cdot\mathcal{H}_2 + \frac{1}{4}\mathcal{H}_1\ .$$

Assim,

$$\mathrm{OPT}(L) \geq \max\left\{\frac{1}{1+\epsilon}\mathcal{H}_1, \frac{1}{4}\mathcal{H}_1 + t\mathcal{H}_2\right\}.$$

Portanto, $\mathrm{BI}_{k,\epsilon}(L) \leq \alpha''_k(r_1)\cdot\mathrm{OPT}(L)+(2k+\beta'_\epsilon)$, onde $\alpha''_k(r_1) = \left[\frac{1}{t} - \frac{(1+\epsilon)}{4t} + (1+\epsilon)\right]$. A última inequação segue mostrando que $\frac{\mathcal{H}_1+\mathcal{H}_2}{\max\left\{\frac{1}{1+\epsilon}\mathcal{H}_1, \frac{1}{4}\cdot\mathcal{H}_1+t\cdot\mathcal{H}_2\right\}} \leq \alpha''_k(r_1)$.

A partir dos dois casos, concluímos que para $k\to\infty$ e $\epsilon\to 0$ o teorema segue. □

4.5.2 Empacotamento *On-Line* em Placas Quadradas, com Rotações Ortogonais

Apresentamos a seguir um algoritmo *on-line* para empacotamento de retângulos, para o caso particular onde as placas são quadradas e os retângulos podem sofrer rotações ortogonais. Veremos que o limite de desempenho assintótico desse algoritmo pode se tornar tão próximo de 2,6875 quanto se queira.

Este algoritmo ilustra um caso em que, ao permitir rotações, podemos obter um algoritmo com melhor limite de desempenho assintótico (veja citação de Chung *et al.* [6] na página 18). Observe que o PEP (orientado) para o caso geral e o PEP para o caso de empacotamento em quadrados são equivalentes, uma vez que podemos reparametrizar o caso geral do PEP para o PEP(1,1).

Este algoritmo será importante na descrição do Algoritmo $\mathrm{RR}^{(3)}_{k,p}$ para o problema de empacotamento tridimensional mencionado no Capítulo 5. Consideramos sem perda de generalidade o PEP$(1,1)$. Antes de apresentar o algoritmo em si, vamos descrever outros algoritmos, chamados Pack$\mathcal{A}^{xy}_{k,i}$, $\mathrm{FF}^{(2)}$ e NF^{xy}_p. O Algoritmo Pack$\mathcal{A}^{xy}_{k,i}$ é para empacotar retângulos do tipo $\mathcal{A}^{xy}_k$;

o Algoritmo $\mathrm{FF}^{(2)}$, desenvolvido por Li e Cheng [43], será usado para empacotar retângulos de dimensões pequenas. E finalmente, o Algoritmo NF_p^{xy} é um refinamento da estratégia NF adaptada para a versão com rotações ortogonais de certos tipos de retângulos.

Primeiramente, vamos descrever o Algoritmo Pack$\mathcal{A}_{k,i}^{xy}$.

Algoritmo Pack$\mathcal{A}_{k,i}^{xy}(A)$

Entrada: Lista de retângulos $A \subseteq \mathcal{A}_{k,i}^{xy}$

Saída: Empacotamento de A em placas $R = (1,1)$, permitindo rotações ortogonais.

1 Sejam p_{ii} e q_{ii} posições apresentadas para os conjuntos $\mathcal{A}_{k,i}^{xy}$ e $\mathcal{B}_{k,i}^{xy}$ na Definição 4.4.2. Seja P a seqüência das posições p_{ii} e q_{ii}.

2 Para cada $r \in A$ faça

2.1 Considere a última placa gerada por este algoritmo até o momento. Caso exista, seja p a primeira posição em P associada a uma placa onde nenhum retângulo foi empacotado nesta posição desta placa.

2.2 Caso não seja encontrada uma posição p no passo 2.1, considere o primeiro ponto p de P em uma placa nova.

2.3 Se $p \in p_{ii}$ então empacote r nesta posição. Caso contrário, empacote $\rho(r)$ na posição p.

3 Retorne o empacotamento gerado pelo passo 2.

Fim algoritmo.

Para se ter uma idéia de como este empacotamento é gerado, considere as sublistas de retângulos $A_1, \ldots, A_k$ (veja a Figura 4.15). Vamos mostrar como seria a lista de pontos para se empacotar a sublista A_i, $1 \leq i \leq k$. Note que os retângulos $b \in A_i$ são tais que $\frac{1}{3} < x(b) \leq r_i \leq \frac{1}{2}$ e $\frac{1}{2} < y(b) \leq s_i = (1 - r_i)$. Assim, é possível empacotar três retângulos de A_i em cada placa. Dois retângulos nas posições $(0,0)$ e $(\frac{1}{2},0)$ e um terceiro retângulo, previamente girado, na posição $(0, s_i)$ (veja a Figura 4.13.)

Note que a área ocupada pelos retângulos empacotados é pelo menos $\frac{1}{2}$. Para as demais listas o empacotamento é análogo, sendo que o único cuidado tomado foi em obter uma garantia de área de pelo menos $\frac{27}{56}$. Com isso, temos o seguinte resultado.

Lema 4.5.2. *Se $\mathcal{P}$ é um empacotamento de uma lista de retângulos L gerado pelo Algoritmo Pack$\mathcal{A}_{k,i}^{xy}$, então*

$$\#(\mathcal{P}) \leq \frac{56}{27} S(L) + 1.$$

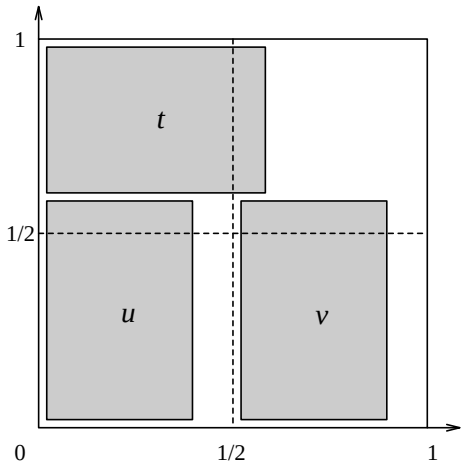


Figura 4.13: Empacotamento de três retângulos de $\mathcal{A}_{k,i}^{xy}$ em uma placa.

Vamos agora descrever o Algoritmo $\text{FF}^{(2)}$ que é uma generalização do algoritmo FF, descrito para o problema de empacotamento unidimensional.

Sejam $\mathcal{S}_1 = \{1\}$, $\mathcal{S}_2 = \{\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^k}, \ldots\}$ e $\mathcal{S}_3 = \{\frac{1}{3}, \frac{1}{6}, \ldots, \frac{1}{3\cdot 2^k}, \ldots\}$. Primeiramente o algoritmo $\text{FF}^{(2)}$ usa um esquema de arredondamento, de forma que cada retângulo tenha sua largura arredondada para cima para o valor mais próximo em $\mathcal{S} := \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3$. Note que $\mathcal{S}_2$ e $\mathcal{S}_3$ são seqüências fortemente divisíveis, i.e., todo elemento na seqüência divide o elemento anterior. Vamos chamar de $\overline{r}$ o retângulo r após arredondamento.

Dizemos que um retângulo r é um (s,i)-*retângulo* se s é a largura do retângulo $\overline{r}$ e, $s \in \mathcal{S}_i$. Uma (s,i)-*faixa* é uma faixa de largura s, onde $s \in \mathcal{S}_i$. O Algoritmo $\text{FF}^{(2)}$ mantém sempre três tipos de placas. As 1-*placas* são as que contêm apenas 1-faixas. As 2-*placas* são as que contêm apenas $(s,2)$-faixas. As 3-*placas* são as que contêm apenas $(s,3)$-faixas. Cada (s,i)-retângulo é empacotado em uma (s,i)-faixa usando o algoritmo FF e cada (s,i)-faixa, $s \in \mathcal{S}_i$, é empacotada em uma i-placa usando também o algoritmo FF.

Algoritmo $\text{FF}^{(2)}$

Entrada: Lista de retângulos $L = (r_1, \ldots, r_n)$.

Saída: Empacotamento de L em placas $R = (1, 1)$.

1 Para $i \leftarrow 1$ até n faça

1.1 Seja s tal que r_i é um (s,k)-retângulo.

1.2 Considere o retângulo r_i como uma barra de comprimento $x(r_i)$ e considere as (s,k)-faixas criadas até o momento como barras de comprimento 1. Use a estratégia FF para empacotar r_i nas s-faixas.

1.3 Se uma nova (s,k)-faixa foi criada no passo 1.2, então use novamente a estratégia FF

para empacotar esta (s,k)-faixa dentro das k-placas.

2 Retorne o empacotamento gerado no passo 1.

Fim algoritmo.

Note que este algoritmo gera as faixas na direção do eixo x. Poderíamos ter uma variante onde as faixas são geradas na direção do eixo y. Assim, quando for necessário especificar a direção, denotaremos por $\mathrm{FF}^{(x2)}$ e $\mathrm{FF}^{(y2)}$ os respectivos algoritmos.

Li e Cheng provaram que o Algoritmo $\mathrm{FF}^{(2)}$ tem um limite de desempenho assintótico que pode se tornar tão próximo de 2,975 quanto se queira. Para nosso uso, provamos o seguinte resultado para este algoritmo.

Lema 4.5.3. *Seja L uma lista de retângulos tal que para todo retângulo $r \in L$ temos $y(r) \leq \frac{1}{2}$ e $x(r) \leq \frac{1}{m}$, $m \geq 2$. Então*

$$\mathrm{FF}^{(2)}(L) \leq \frac{3}{2}\left(\frac{m}{m-1}\right) S(L) + 6.$$

Prova. Seja L_s, $s \in \mathcal{S}$, a sublista de todos os (s,i)-retângulos de L. Seja n_s, $s \in \mathcal{S}_i$, o número de (s,i)-faixas e N_i, $i = 1,2,3$, o número de i-placas criadas pelo Algoritmo $\mathrm{FF}^{(2)}$. Note que com as restrições impostas pelo lema, o Algoritmo $\mathrm{FF}^{(2)}$ não gera empacotamentos em 1-placas.

Para $s \in \mathcal{S}_2 \cup \mathcal{S}_3$ temos que

$$\begin{aligned} S(L_s) &= \sum_{r \in L_s} x(r) \cdot y(r) \\ &> \frac{2}{3} \cdot s \cdot \sum_{r \in L_s} x(r) \\ &\geq \frac{2}{3} \cdot s \cdot \left(1 - \frac{1}{m}\right) \cdot (n_s - 1) \\ &= \frac{2}{3} \cdot s \cdot \left(\frac{m-1}{m}\right) \cdot (n_s - 1). \end{aligned}$$

Assim,

$$\begin{aligned} S(L) &= \sum_{i=2}^{3} \sum_{s \in \mathcal{S}_i} S(L_s) \\ &\geq \frac{2}{3}\left(\frac{m-1}{m}\right) \sum_{i=2}^{3} \sum_{s \in \mathcal{S}_i} (s \cdot n_s - s). \end{aligned}$$

Note que para cada tipo, o espaço não ocupado por faixas em todas as placas, exceto na última, tem soma de largura no máximo 1 e portanto,

$$S(L) \geq \frac{2}{3}\left(\frac{m-1}{m}\right) \sum_{i=2}^{3} ((N_i - 2) - 1)$$

$$
\begin{aligned}
&= \frac{2}{3}\left(\frac{m-1}{m}\right)(N_2 + N_3 - 6) \\
&= \frac{2}{3}\left(\frac{m-1}{m}\right)(\mathrm{FF}^{(2)} - 6).
\end{aligned}
$$

Isolando $\mathrm{FF}^{(2)}$, obtemos,

$$\mathrm{FF}^{(2)}(L) \leq \frac{3}{2}\left(\frac{m}{m-1}\right) S(L) + 6.$$

□

Outro algoritmo que iremos usar será o Algoritmo NF_p^{xy}. Este algoritmo empacota os retângulos de L em placas $R = (1, 1)$ da seguinte maneira.

O Algoritmo NF_p^{xy} começa a empacotar os retângulos de L na ordem em que aparecem na lista L. O empacotamento gerado em cada placa é dividido em duas partes. Na primeira parte o algoritmo empacota os retângulos na região $[0, 1) \times [0, p)$ usando o Algoritmo NF^x. Quando não puder mais empacotar nesta região, o algoritmo empacota os retângulos na região $[0, 1) \times [1-p, 1)$ usando o Algoritmo NF^y sobre os próximos itens de L, girando previamente cada item. Quando não puder mais empacotar nesta região, o Algoritmo NF_p^{xy} deixa de empacotar nesta placa, para começar todo este processo sobre os itens restantes em uma nova placa R. O processo continua até que todos os retângulos de L tenham sido empacotados (veja Figura 4.14).

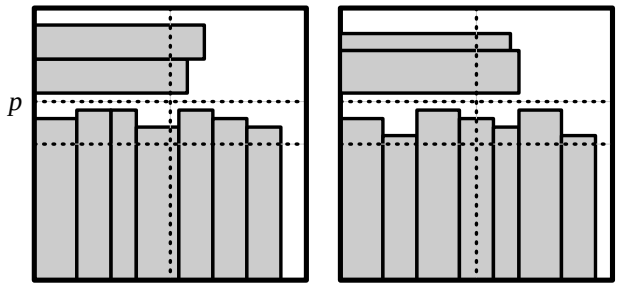


Figura 4.14: Exemplo de empacotamento gerado pelo Algoritmo NF_p^{xy}.

O valor de p e a lista de retângulos L devem ser escolhidos de tal forma a viabilizar o empacotamento gerado pelo algoritmo.

Agora podemos apresentar o algoritmo desta seção.

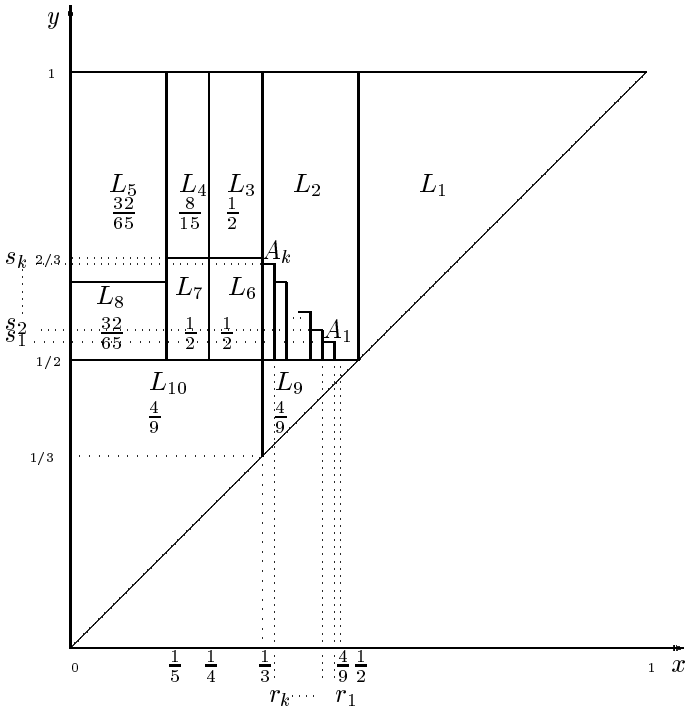


Figura 4.15: Partição da lista L gerada pelo Algoritmo RR_k

Algoritmo RR_k

Entrada: Lista de retângulos $L = (r_1, \ldots, r_n)$.

Saída: Empacotamento de L em placas $R = (1, 1)$, permitindo rotações ortogonais.

1 Gire todos os retângulos $r \in L$, de forma a obter $y(r) \geq x(r)$.

2 Divida a lista L em sublistas da seguinte forma (veja a Figura 4.15).

$$
\begin{aligned}
&A_i \leftarrow L \cap \mathcal{A}^{xy}_{k,i}, \quad i = 1, \ldots, k,\\
&L' \leftarrow L \setminus \textstyle\bigcup_{i=1}^{k} A_i,\\
&L_i \leftarrow L' \cap \mathcal{C}^{xy}\left[\tfrac{1}{i+1}, \tfrac{1}{i}\ ;\ \tfrac{1}{2}, 1\right] \quad 1, 2,\\
&L_3 \leftarrow L' \cap \mathcal{C}^{xy}\left[\tfrac{1}{4}, \tfrac{1}{3}\ ;\ \tfrac{2}{3}, 1\right],\\
&L_4 \leftarrow L' \cap \mathcal{C}^{xy}\left[\tfrac{1}{5}, \tfrac{1}{4}\ ;\ \tfrac{2}{3}, 1\right],\\
&L_5 \leftarrow L' \cap \mathcal{C}^{xy}\left[0, \tfrac{1}{5}\ ;\ \tfrac{8}{13}, 1\right],\\
&L_6 \leftarrow L' \cap \mathcal{C}^{xy}\left[0, \tfrac{1}{5}\ ;\ \tfrac{8}{13}, 1\right],\\
&L_7 \leftarrow L' \cap \mathcal{C}^{xy}\left[\tfrac{1}{5}, \tfrac{1}{4}\ ;\ \tfrac{1}{2}, \tfrac{2}{3}\right],\\
&L_8 \leftarrow L' \cap \mathcal{C}^{xy}\left[0, \tfrac{1}{5}\ ;\ \tfrac{1}{2}, \tfrac{8}{13}\right],\\
&L_9 \leftarrow L' \cap \mathcal{C}^{xy}\left[\tfrac{1}{4}, \tfrac{1}{3}\ ;\ \tfrac{1}{2}, \tfrac{2}{3}\right],\\
&L_{10} \leftarrow L' \cap \mathcal{C}^{xy}\left[0, \tfrac{1}{3}\ ;\ 0, \tfrac{1}{2}\right].
\end{aligned}
$$

3 $\mathcal{P}^A_i \leftarrow \mathrm{Pack}\mathcal{A}^{xy}_{k,i}(A_i)$, para $i = 1, \ldots, k$;

4 $\mathcal{P}_i \leftarrow \mathrm{NF}^x(L_i)$, para $i = 1, \ldots, 5$;

5 $\mathcal{P}_6 \leftarrow \mathrm{NF}^{xy}_{\frac{2}{3}}(L_6)$;

6 $\mathcal{P}_7 \leftarrow \mathrm{NF}^{xy}_{\frac{2}{3}}(L_7)$;

7 $\mathcal{P}_8 \leftarrow \mathrm{NF}^{xy}_{\frac{8}{13}}(L_8)$;

8 $\mathcal{P}_i \leftarrow \mathrm{FF}^{(2)}(L_i)$, para $i = 9, 10$;

9 $\mathcal{P}_{aux} \leftarrow \mathcal{P}_1^A \| \ldots \| \mathcal{P}_k^A \| \mathcal{P}_2 \| \ldots \| \mathcal{P}_{10}$;

10 $\mathcal{P} \leftarrow \mathcal{P}_1 \cup \mathcal{P}_{aux}$.

11 Retorne $\mathcal{P}$.

Fim algoritmo.

O Algoritmo RR_k, como descrito, é *off-line*. Mas sua transformação para *on-line* é direta e simples, pois todos os algoritmos usados como subrotina são algoritmos *on-line*, ou podem ser escritos de forma *on-line*. Com isso, podemos considerá-lo *on-line*.

Este algoritmo tem um limite de desempenho assintótico tão próximo de 2,6875 quanto se queira. É o que provamos a seguir.

Lema 4.5.4. *Seja L uma instância para o* $\mathrm{PEP}^r(1,1)$. *Então* $\mathrm{RR}_k(L) \leq \alpha_k \cdot \mathrm{OPT}(L) + k + 14$, *onde α_k tende a* 2,6875 *à medida que k se torna grande.*

Prova. Note que para o empacotamento da lista L_1, temos

$$\begin{aligned} \#(\mathcal{P}_1) &= \mathrm{OPT}(L_1) \\ &\leq \mathrm{OPT}(L) \end{aligned}$$

e

$$\#(\mathcal{P}_1) \leq \frac{1}{1/4} S(L_1).$$

Vamos analisar as relações válidas para os demais empacotamentos. Pelo Lema 4.5.2,

$$\#(\mathcal{P}_i^A) \leq \frac{1}{4/9} S(A_i) + 1, \quad i = 1, \ldots, k.$$

Aplicando o Lema 4.2.1 sobre o empacotamento das listas $L_2, \ldots, L_9$, temos

$$\begin{aligned} \#(\mathcal{P}_2) &\leq \frac{1}{r_1} S(L_2) + 1; \\ \#(\mathcal{P}_i) &\leq \frac{1}{4/9} S(L_i) + 1, \quad i = 3, \ldots, 9. \end{aligned}$$

Pelo Lema 4.5.3,

$$\#(\mathcal{P}_{10}) \quad \leq \quad \frac{1}{4/9} S(L_{10}) + 6.$$

Como $r_1 < \frac{4}{9}$, temos

$$\#(\mathcal{P}_{aux}) \leq \frac{1}{r_1} S(L_{aux}) + k + 14. \tag{4.40}$$

Tomando $h_1 = \#(\mathcal{P}_1)$ e $h_2 = \#(\mathcal{P}_{aux}) - (k+14)$, temos que $\mathrm{RR}_k(L) \leq \alpha_k \cdot \mathrm{OPT}(L) + k + 14$, onde $\alpha_k \leq \frac{h_1+h_2}{\max\{h_1, \frac{1}{4}h_1 + r_1 \cdot h_2\}}$. Como $\lim_{k\to\infty} r_1 = \frac{4}{9}$ temos que $\lim_{k\to\infty} \alpha_k \leq 2{,}6875$. □

O seguinte resultado mostra que o Algoritmo RR_k apresenta um limite de desempenho assintótico bem próximo de 2,6875, para valores pequenos de k.

Corolário 4.5.5. *Para toda lista de retângulos L para o* PEP^r *temos*

$$\mathrm{RR}_{13}(L) \leq 2{,}6876 \cdot \mathrm{OPT}(L) + 27.$$

4.5.3 Algoritmo OBI

Apresentamos nesta seção um algoritmo para o empacotamento *on-line* de retângulos em placas usando rotações ortogonais. Este algoritmo é uma adaptação do algoritmo desenvolvido por Coppersmith e Raghavan [12] para este mesmo problema, sem usar rotações. O limite de 3,25 do algoritmo desses autores é mantido para o PEP^r.

A idéia por trás deste algoritmo é construir um empacotamento dividido em duas partes. Uma parte é constituída de retângulos, 'grandes', de $\wp_4$. A outra parte, é constituída de retângulos de $\wp_1 \cup \wp_2 \cup \wp_3$. Para estes últimos, é gerado um empacotamento com garantia de área $\frac{1}{3}$. Para os retângulos de $\wp_4$ é construído um empacotamento ótimo com garantia de área $\frac{1}{4}$. Para assegurar que este empacotamento é ótimo, alguns retângulos de $L \cap \wp_4$ são girados previamente.

Algoritmo OBI

Entrada: Lista de retângulos $L = (r_1, \ldots, r_n)$.

Saída: Empacotamento *on-line* de L em placas $R = (a, b)$, permitindo rotações ortogonais.

1 Gire todos os retângulos $r \in L$, tais que $\rho(r) \in \wp_1 \cup \wp_2 \cup \wp_3$.

2 Divida a lista L em sublistas $L_1, \ldots, L_6$ da seguinte forma (veja a Figura 4.16).

$$
\begin{aligned}
L_1 &\leftarrow L \cap \wp_4, \\
L_2 &\leftarrow L \cap \mathcal{C}^{xy}\left[\tfrac{1}{3}, \tfrac{1}{2}\ ;\ \tfrac{1}{2}, 1\right], \\
L_3 &\leftarrow L \cap \mathcal{C}^{xy}\left[0, \tfrac{1}{3}\ ;\ \tfrac{1}{2}, 1\right], \\
L_4 &\leftarrow L \cap \mathcal{C}^{xy}\left[\tfrac{1}{2}, 1\ ;\ \tfrac{1}{3}, \tfrac{1}{2}\right], \\
L_5 &\leftarrow L \cap \mathcal{C}^{xy}\left[\tfrac{1}{2}, 1\ ;\ 0, \tfrac{1}{3}\right], \\
L_6 &\leftarrow L \cap \wp_1.
\end{aligned}
$$

3 Construa empacotamentos $\mathcal{P}_1, \ldots, \mathcal{P}_6$ da seguinte maneira:

3.1 $\mathcal{P}_i \leftarrow \mathrm{FF}^{(x2)}(L_i)$, para $i = 1, 2, 3, 6$;

3.2 $\mathcal{P}_i \leftarrow \mathrm{FF}^{(y2)}(L_i)$, para $i = 4, 5$;

4 $\mathcal{P} \leftarrow \mathcal{P}_1 \| \ldots \| \mathcal{P}_6$.

5 Retorne $\mathcal{P}$.

Fim algoritmo.

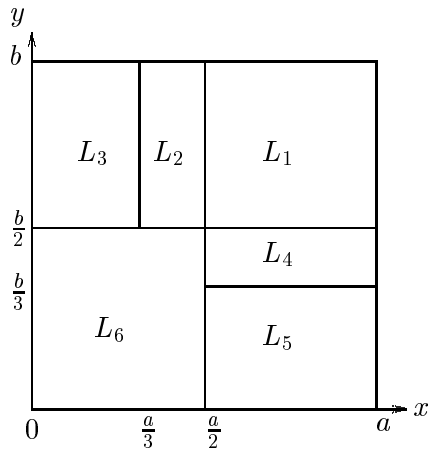


Figura 4.16: Subdivisão dos itens de L pelo Algoritmo OBI.

Da forma como este algoritmo foi descrito, ele é *off-line*. Mas sua transformação em algoritmo *on-line* é fácil, já que todos os algoritmos usados como subrotinas são também *on-line*. Deixamos esta transformação para o leitor interessado. Assim, consideramos que este algoritmo é *on-line*.

O seguinte resultado é valido para o Algoritmo OBI.

Teorema 4.5.6. *Para qualquer instância L do* PEP^r*, temos que*

$$\text{OBI}(L) \leq 3{,}25 \cdot \text{OPT}(L) + 10.$$

Prova. Primeiramente, note que (pelo passo 1) a lista L_1 contém retângulos tais que dois destes não podem ser empacotados em uma mesma placa. Assim,

$$\begin{aligned} \text{OPT}(L) &\geq \text{OPT}(L_1) \\ &= \#(\mathcal{P}_1). \end{aligned} \tag{4.41}$$

Como cada retângulo de L_1 tem área de pelo menos $\frac{ab}{4}$, temos

$$\#(\mathcal{P}_1) \leq \frac{S(L_1)}{ab/4}. \tag{4.42}$$

Considere a lista L_2. O Algoritmo $\text{FF}^{(x2)}$ empacota dois retângulos de L_2 em cada placa, exceto talvez na última. Como cada retângulo de L_2 tem pelo menos uma área de $\frac{ab}{6}$, concluímos a partir do Lema 4.2.1 que

$$\#(\mathcal{P}_2) \leq \frac{S(L_2)}{ab/3} + 1.$$

Este mesmo raciocínio pode ser feito para as listas $L_3, \ldots, L_5$. Assim, temos

$$\#(\mathcal{P}_i) \leq \frac{S(L_i)}{ab/3} + 1, \quad \text{para} \quad i = 2, \ldots, 5. \tag{4.43}$$

Pelo Lema 4.5.3, temos

$$\#(\mathcal{P}_6) \leq \frac{S(L_6)}{ab/3} + 6. \tag{4.44}$$

Definindo $n_1 := \#(\mathcal{P}_1)$ e $n_2 := \#(\mathcal{P}_2) + \cdots + \#(\mathcal{P}_6) - 10$, e usando (4.43) e (4.44) temos

$$\frac{S(L \setminus L_1)}{ab} \geq \frac{1}{3} n_2. \tag{4.45}$$

Por outro lado,

$$\begin{aligned} \text{OPT}(L) &\geq \frac{S(L)}{ab} \\ &\geq \frac{S(L_1)}{ab} + \frac{S(L \setminus L_1)}{ab} \\ &\geq \frac{1}{4} n_1 + \frac{1}{3} n_2. \end{aligned} \tag{4.46}$$

Portanto, podemos obter que

$$\text{OBI}(L) \leq \alpha \cdot \text{OPT}(L) + 10,$$

onde $\alpha = \frac{n_1 + n_2}{\max\{n_1, \frac{1}{4} n_1 + \frac{1}{3} n_2\}} \leq 3{,}25$. ∎

4.6 Resumo dos Algoritmos

A seguir, apresentamos duas tabelas com os algoritmos para os problemas de empacotamento em placas. Uma para o PEP e outra para o PEP^r. Alguns dos algoritmos desenvolvidos para o PEP foram apresentados nos algoritmos do PEP^r, pois estes algoritmos tiveram suas análises baseadas apenas em argumentos envolvendo área. Assim, os limites provados para estes algoritmos servem tanto para o PEP como para o PEP^r.

Os algoritmos desenvolvidos nesta tese estão marcados com $\star$ na coluna da referência.

Algoritmos para o PEP

Algoritmo	**Tipo**	α	β	**Ref.**	**Condição**
STP_m	*off-line*	$\alpha(\mathrm{STP}_m)$	$\mathcal{O}(m)$	$\star$	Retângulos de L têm dimensões pequenas
IOFF_m	*on-line*	$\left(\frac{m+2}{m+1}\right)^2 + \frac{2}{m(m+1)}$	$\mathcal{O}(m^2)$	$\star$	Retângulos de L têm dimensões pequenas
HFF	*off-line*	2,125	5	[6]	Caso Geral
$\mathrm{FF}^{(2)}$	*on-line*	$\asymp 2{,}86$	$\mathcal{O}\left(\frac{1}{1-r}\frac{1}{1-s}\right)$	[39, 14]	Caso Geral

Algoritmos para o PEP^r

Algoritmo	**Tipo**	α	β	**Ref.**	**Condição**
$\mathrm{BI}_{\mathbf{m}}$	*off-line*	$\left(\frac{m+1}{m}\right)^2$	6	$\star$	Retângulos de L têm dimensões pequenas
$\mathrm{OFF}_m^{(2)}$	*on-line*	$\left(\frac{m+1}{m}\right)^2$	$\mathcal{O}(m^2)$	$\star$	Retângulos de L têm dimensões pequenas
RR_k	*on-line*	$\asymp 2{,}6875$	$\mathcal{O}(k)$	$\star$	Empacotamento em Placas quadradas
$\mathrm{BI}_{k,\epsilon}$	*off-line*	$\asymp 2{,}639\ldots$	$\mathcal{O}\left(k+\frac{1}{\epsilon}\right)$	$\star$	Caso Geral
OBI	*on-line*	3,25	10	$\star$	Caso Geral

Capítulo 5

Problema de Empacotamento Tridimensional

5.1 Introdução

Apresentamos neste capítulo algoritmos de aproximação para o Problema de Empacotamento Tridimensional (PET). Consideramos duas versões, uma em que todas as caixas são orientadas e outra em que as caixas de L podem ser giradas ortogonalmente em torno do eixo z.

Li e Cheng [40] foram os primeiros a estudar este problema sob a ótica de algoritmos de aproximação. Em 1990, esses autores apresentaram vários algoritmos para o PET: para o caso geral, um algoritmo com limite de desempenho assintótico 3,25; e para o caso especial onde todas as caixas têm fundo quadrado, um algoritmo com limite de desempenho assintótico 2,6875. Em [44] eles melhoraram este limite apresentando um algoritmo *on-line* com limite de desempenho que pode se tornar tão próximo de 2,89 quanto se queira. Apresentamos um algoritmo que melhora este limite para menos que 2,67. Este resultado responde a uma questão levantada por Li e Cheng [44], sobre a existência de um algoritmo polinomial com limite de desempenho assintótico menor que 2,89. Desenvolvemos também algoritmos particulares para casos especiais deste problema. Para o caso em que as caixas de L têm fundo quadrado apresentamos um algoritmo com limite de desempenho assintótico não maior que 2,543. E para o caso especial onde a caixa B e todas as caixas de L têm fundo quadrado, melhoramos o limite (devido a Li e Cheng) para menos que 2,361 .

Para o caso em que as caixas de L podem sofrer rotações em torno do eixo z apresentamos uma aplicação interessante em um problema de escalonamento de processos em computadores paralelos. Este problema foi introduzido por Li e Cheng [41], que foram os primeiros a desenvolver um algoritmo de aproximação para este problema. Esses autores apresentaram um algoritmo com um limite de desempenho assintótico 4,572. Obtivemos um algoritmo para este

problema, adaptando o algoritmo para o caso orientado (com limite de desempenho assintótico 2,67), e mantendo o mesmo limite de desempenho assintótico. Descrevemos também um algoritmo para o caso especial em que a caixa B tem fundo quadrado e mostramos que seu limite de desempenho assintótico é menor que 2,528.

Para o caso *on-line* apresentamos um algoritmo com um limite de desempenho assintótico que pode se tornar tão próximo de 3,25 quanto se queira, e também outro algoritmo para o caso especial onde a caixa B tem fundo quadrado, com um limite de desempenho assintótico que pode se tornar tão próximo de 2,6875 quanto se queira.

Também consideramos o caso especial, *on-line* e *off-line*, onde as caixas têm fundo de dimensões pequenas.

Os melhores limites descritos para instâncias especiais não são aplicações diretas dos algoritmos para o caso geral. Cada um deles provém de um algoritmo particular, que é resultado de análises específicas para as instâncias em consideração.

5.2 Definições

Denotamos uma caixa b_i como uma tripla $b_i = (x_i, y_i, z_i)$, onde x_i, y_i e z_i são seus *comprimento*, *largura* e *altura*, respectivamente.

O fundo da caixa B será sempre representado no plano xy e sua altura é relativa ao eixo z. Assim, usaremos a notação $S(b)$ para designar a área de fundo de uma caixa b.

Note que, usando um sistema de coordenadas tridimensional, uma caixa $B = (a, b, c)$ pode ser vista como a região $[0, a) \times [0, b) \times [0, c)$, e podemos definir um *empacotamento tridimensional orientado* de L em B como $\mathcal{P} : L \to [0, a) \times [0, b) \times [0, c)$, tal que

$$\mathcal{P}^x(b_i) + x(b_i) \leq a, \mathcal{P}^y(b_i) + y(b_i) \leq b \quad \text{e} \quad \mathcal{P}^z(b_i) + z(b_i) \leq c,$$

onde $\mathcal{P}(b_i) = (\mathcal{P}^x(b_i), \mathcal{P}^y(b_i), \mathcal{P}^z(b_i))$, $i = 1, \ldots, n$.

Mais ainda, se $\mathcal{R}(b_i)$ é definido como

$$\mathcal{R}(b_i) = [\mathcal{P}^x(b_i), \mathcal{P}^x(b_i) + x_i) \times [\mathcal{P}^y(b_i), \mathcal{P}^y(b_i) + y_i) \times [\mathcal{P}^z(b_i), \mathcal{P}^z(b_i) + z_i),$$

então

$$\mathcal{R}(b_i) \cap \mathcal{R}(b_j) = \emptyset \quad \forall i, j, \ 1 \leq i \neq j \leq n.$$

As condições acima significam que cada caixa em L deve estar inteiramente contida dentro da caixa B e deve ser empacotada ortogonalmente e orientada em todas as três dimensões (veja a Figura 5.1). Mais ainda, duas caixas de L não podem se sobrepor neste empacotamento.

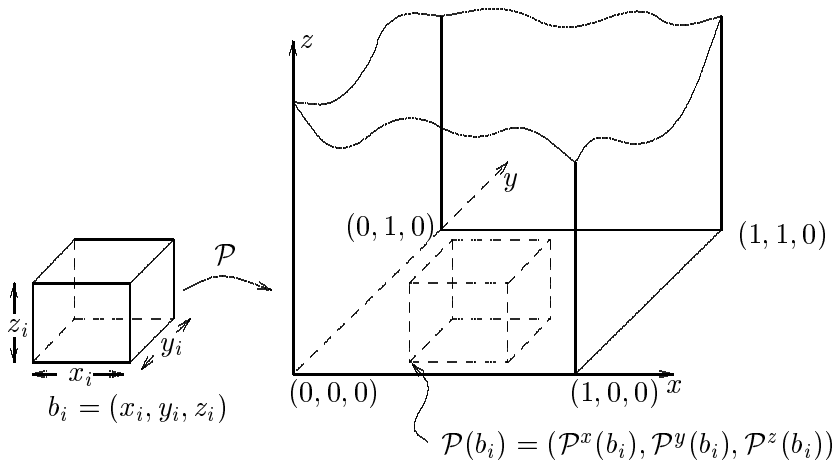


Figura 5.1: Empacotamento de uma caixa $b_i = (x_i, y_i, z_i)$ dentro da caixa $B = (1, 1, \infty)$.

Dado um empacotamento $\mathcal{P}$ de L, denotamos por $H(\mathcal{P})$ a *altura* do empacotamento $\mathcal{P}$, *i.e.*, $H(\mathcal{P}) := \max\{\mathcal{P}^z(b) + z(b) : \ b \in L\}$.

Problema. *Problema de Empacotamento Tridimensional (PET): Dados uma lista de caixas L e uma caixa $B = (a, b, \infty)$, encontrar um empacotamento tridimensional orientado de L em B cuja altura seja a menor possível.*

Assumiremos para o PET que a caixa B tem dimensões $(1, 1, \infty)$, já que, se este não for o caso e tivermos $B = (a, b, \infty)$, $a > 0$, $b > 0$, podemos dividir o comprimento $x(b_i)$ e a largura $y(b_i)$ de cada caixa b_i por a e b, respectivamente.

Problema. *Problema de Empacotamento Tridimensional z-Orientado* (PET^r)*: Dados uma lista de caixas L e uma caixa $B = (a, b, \infty)$, encontrar um empacotamento tridimensional orientado de uma lista $L' \in \Gamma^z(L)$ em B cuja altura seja a menor possível.*

Quando estivermos referindo ao PET^r, dado um conjunto $\mathcal{T}$ de caixas, diremos que uma caixa b é do tipo $\mathcal{T}$ se $b \in \mathcal{T}$ ou $\rho(b) \in \mathcal{T}$.

Sejam $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_v$ empacotamentos de listas disjuntas, $L_1, L_2, \ldots, L_v$, respectivamente. Definimos a *concatenação* desses empacotamentos como um empacotamento $\mathcal{P} = \mathcal{P}_1 \| \mathcal{P}_2 \| \ldots \| \mathcal{P}_v$ de $L = L_1 \cup L_2 \cup \ldots \cup L_v$, onde $\mathcal{P}(b) = (\mathcal{P}_i^x(b), \mathcal{P}_i^y(b), \sum_{j=1}^{i-1} H(\mathcal{P}_j) + \mathcal{P}_i^z(b))$, para todo $b \in L_i$, $1 \leq i \leq v$.

Definimos um *nível* em um empacotamento $\mathcal{P}$ como uma região

$$N = [0, a) \times [0, b) \times [Z_1, Z_2),$$

na qual há um conjunto S de caixas tal que

$$\forall c \in S : \mathcal{P}^z(c) = Z_1 \text{ e } Z_2 - Z_1 \geq \max_{c \in S}(z(c)).$$

Lembramos que $\mathcal{P}^z(c)$ é a posição relativa ao eixo z, onde c é empacotado. Neste caso, $Z_2 - Z_1$ é a *altura* do nível N. Quando a altura de um nível não for definida previamente pelo algoritmo, consideramos que a sua altura é $Z_2 - Z_1 = \max_{c \in S}(z(c))$. Dizemos que o nível N é ortogonal ao eixo z. De forma análoga, podemos ter níveis ortogonais aos eixos x ou y.

Uma *faixa em um nível* N, como definido acima, é uma região

$$F = [0, a) \times [Y_1, Y_2) \times [Z_1, Z_2)$$

na qual há um conjunto S de caixas tal que

$$\forall c \in S : \mathcal{P}^y(c) = Y_1 \text{ e } \mathcal{P}^z(c) = Z_1 \text{ ; } Y_2 - Y_1 \geq \max_{c \in S}(y(c)) \text{ e } Z_2 - Z_1 \geq \max_{c \in S}(z(c)).$$

Dizemos que a faixa F tem largura $Y_2 - Y_1$. Quando a largura de uma faixa não for definida previamente pelo algoritmo, consideramos que sua largura é $Y_2 - Y_1 = \max_{c \in S}(y(c))$.

Dizemos que a faixa é ortogonal ao eixo z e paralela ao eixo y. Faixas em outras direções são definidas analogamente.

Seja S_p um conjunto de arredondamento, $S_p := \{1, p, p^2, \ldots\}$, $0 < p < 1$. Toda caixa c em L, tal que $p^{i+1} < z(c) \leq p^i$, é submetida a um arredondamento na altura p^i. O arredondamento pode ser tão pequeno quanto se queira, bastando para isto tomar p suficientemente próximo de 1. Cada caixa tal que $p^{i+1} < z(c) \leq p^i$ é denominada uma i-caixa. Toda i-caixa é empacotada em um nível de altura p^i, tal nível é denominado de i-nível.

Consideraremos aqui que Z é um limite superior para a altura de uma caixa de L.

Denotaremos por $PET(a, b)$ o problema do empacotamento tridimensional onde a caixa B é do tipo (a, b, ∞).

O lema seguinte, é um resultado referente à altura de um empacotamento que consiste de níveis com certas propriedades. Este lema é uma generalização de resultados provados para algoritmos particulares apresentados em [40].

Lema 5.2.1. *Seja L uma instância do* PET *e $\mathcal{P}$ um empacotamento de L consistindo de níveis $N_1, \ldots, N_v$ onde $\min\{z(b) : b \in N_i\} \geq \max\{z(b) : b \in N_{i+1}\}$, e $S(N_i) \geq s$ para uma dada constante $s > 0$, $i = 1, \ldots, v-1$. Então, $H(\mathcal{P}) \leq \frac{1}{s}V(L) + Z$.*

Prova. Seja h_i a altura do nível N_i, $i = 1, \ldots, v$.

$$\begin{aligned} V(L) &\geq S(N_1) \cdot h_2 + S(N_2) \cdot h_3 + \cdots + S(N_{v-1}) \cdot h_v \\ &\geq s \cdot h_2 + s \cdot h_3 + \cdots + s \cdot h_v \\ &= s \cdot (\sum_{i=1}^{v} h_i - h_1) \\ &= s \cdot (H(\mathcal{P}) - Z). \end{aligned}$$

□ A constante s mencionada no Lema 5.2.1 será denominada de *garantia de área* do empacotamento $\mathcal{P}$.

5.3 Caso Orientado

Nesta seção apresentamos alguns algoritmos de estratégias simples, generalizações de algoritmos do caso unidimensional, já conhecidos na literatura e algoritmos desenvolvidos nesta tese. Apresentamos algoritmos para empacotamento de caixas de fundo de dimensões pequenas, *on-line* e *off-line*, para o empacotamento de caixas com fundo quadrado e para o caso geral. Estes últimos são *off-line*.

5.3.1 Estratégias *Next Fit* (NF), *First Fit* (FF) e Outros

Em [40], Li e Cheng apresentaram dois algoritmos que aplicam a estratégia NF e FF para o PET. Eles são denominados de NFDH$^{(t)}$ (*Next Fit Decreasing Height*) e FFDH (*First Fit Decreasing Height*).

Primeiramente, descreveremos o Algoritmo NFDH$^{(t)}$. Este algoritmo tem duas variantes: NFDH^x e NFDH^y. A notação NFDH$^{(t)}$ é usada para referir a qualquer uma destas variantes.

O Algoritmo NFDH^x primeiro ordena as caixas de L em ordem não-crescente de altura: $b_1, b_2, \ldots, b_n$. A primeira caixa b_1 é empacotada na posição $(0, 0, 0)$, a próxima caixa é empacotada na posição $(x(b_1), 0, 0)$, assim por diante, lado a lado, até que uma caixa não possa ser empacotada nesta faixa. Neste momento, a próxima caixa b_k é empacotada na posição $(0, y(b^*), 0)$, onde $y(b^*) = \max\{y(b_i), i = 1, \ldots, k-1\}$. Este procedimento é repetido até que uma caixa b_l não possa ser empacotada no primeiro nível. Então o algoritmo empacota esta caixa em um novo nível na altura $z(b_1)$ de B. O algoritmo procede desta maneira até que todas as caixas de L tenham sido empacotadas.

O Algoritmo NFDH^y é análogo ao Algoritmo NFDH^x, exceto que ele gera as faixas na direção do eixo y.

Como foi visto no Capítulo 3 o Algoritmo $\text{NFDH}^{(f)}$ que usa a estratégia NF para o caso bidimensional em faixa, tem limite de desempenho assintótico 2. No entanto, como mostra o teorema a seguir, o Algoritmo $\text{NFDH}^{(t)}$ tem desempenho de pior caso ilimitado [40].

Teorema 5.3.1. *Para cada inteiro $M > 1$, existe uma instância L para o $PET(a, b)$ tal que*

$$\text{NFDH}^{(t)}(L) > M \cdot \text{OPT}(L).$$

No Algoritmo $\text{NFDH}^{(t)}$, sempre que se começa uma nova faixa, todas as faixas anteriormente criadas são esquecidas. Um meio de se evitar possíveis "desperdícios" seria usar a mesma idéia do algoritmo de empacotamento bidimensional em faixa FFDH, no qual são sempre revisitadas as faixas anteriores. Esta estratégia pode ter boas conseqüências na prática, mas não faz com que o limite de desempenho deste algoritmo melhore. De fato, Li e Cheng [40] provaram que o Algoritmo FFDH também tem limite de desempenho de pior caso ilimitado.

Embora no caso geral o Algoritmo $\text{NFDH}^{(t)}$ tenha desempenho de pior caso ilimitado, restringindo-se os tamanhos das caixas contidas na lista L, é possível conseguir bons limites de desempenho nestes casos. O seguinte resultado, provado por Li e Cheng [40], pode ser derivado como um corolário do Lema 5.2.1.

Lema 5.3.2. *Se $L \subset \mathcal{C}^{xy}\left[\frac{1}{m+1}, \frac{1}{m}\ ;\ 0, \frac{1}{m}\right]$ então $\text{NFDH}^{y}(L) \leq \left(\frac{m+1}{m-1}\right) V(L) + Z$. O mesmo resultado vale para o Algoritmo NFDH^{x} quando aplicado à lista $L \in \mathcal{C}^{xy}\left[0, \frac{1}{m}\ ;\ \frac{1}{m+1}, \frac{1}{m}\right]$.*

Prova. Seja $\mathcal{P}$ um empacotamento gerado pelo Algoritmo NFDH^{y} sobre uma lista $L \subset \mathcal{C}^{xy}\left[\frac{1}{m+1}, \frac{1}{m}\ ;\ 0, \frac{1}{m}\right]$. Observe que em cada nível de $\mathcal{P}$, exceto talvez o último, há exatamente m faixas. Cada faixa contém caixas com largura total maior que $\left(1 - \frac{1}{m}\right)$ e cada caixa tem comprimento maior que $\frac{1}{m+1}$. Portanto a garantia de área de $\mathcal{P}$ é pelo menos $m \cdot \left(1 - \frac{1}{m}\right) \cdot \frac{1}{m+1} = \frac{m-1}{m+1}$. O resultado segue usando o Lema 5.2.1. □

Um outro algoritmo que será usado como subrotina de algoritmos do caso geral do PET e PET^{r} é o Algoritmo **OC** (*One Column*). Dada uma lista de caixas, digamos $L = (b_1, \ldots, b_n)$, este algoritmo empacota cada caixa b_{i+1} no topo da caixa b_i, para $i = 1, \ldots, n-1$. Assim, a primeira caixa é empacotada na posição $(0, 0, 0)$, a segunda é empacotada na posição $(0, 0, z(b_1))$, assim por diante. É fácil verificar o seguinte resultado.

Lema 5.3.3. *Se $\mathcal{P}$ é um empacotamento gerado pelo Algoritmo* OC *quando aplicado à lista L e s é uma constante tal que $S(b) \geq s$ para cada caixa b em L, então $H(\mathcal{P}) \leq \frac{V(L)}{s}$.*

Lema 5.3.4. *Seja $\mathcal{P}$ um empacotamento gerado pelo Algoritmo* OC *quando aplicado a uma lista L tal que $x(b) > \frac{1}{2}$ e $y(b) > \frac{1}{2}$ para cada caixa b em L. Então $H(\mathcal{P}) = \text{OPT}(L)$.*

5.3.2 Empacotamento de Caixas com Fundo Pequeno

Apresentamos nesta seção um outro algoritmo que gera um empacotamento em níveis, como descrevemos no Lema 5.2.1, e usa a estratégia NF. Este algoritmo foi apresentado por Li e Cheng em [41] e é usado para empacotar lista de caixas $L \in \mathcal{C}^{xy}\left[0, \frac{1}{m}\ ;\ 0, \frac{1}{m}\right]$, $m \geq 3$. Denominaremos este algoritmo de LL_m.

Vamos descrever informalmente este algoritmo usando o Algoritmo $\mathrm{NFDH}^{(p)}$ para empacotamento de retângulos pequenos em placas.

Inicialmente, este algoritmo ordena as caixas de L em ordem não-crescente de altura. A seguir, divide a lista L em sublistas $L_1, \ldots, L_v$, obtendo $L = L_1 \| L_2 \| \ldots \| L_v$, sendo que cada sublista preserva a ordem (não-crescente) das caixas, e

$$\begin{array}{ll} S(L_i) \leq \left(\frac{m-2}{m}\right) + \left(\frac{1}{m}\right)^2 & \text{para } \ i = 1, \ldots, v, \\ S(L_i) + S(\mathit{first}(L_{i+1})) > \left(\frac{m-2}{m}\right) + \left(\frac{1}{m}\right)^2 & \text{para } \ i = 1, \ldots, v-1. \end{array}$$

Então, o Algoritmo LL_m usa o algoritmo de empacotamento bidimensional em placas $\mathrm{NFDH}^{(p)}$ para empacotar cada lista L_i em apenas um nível, digamos N_i (veja o Lema 4.4.2). O empacotamento final é a concatenação de cada um destes níveis. Como a área de cada caixa é no máximo $\frac{1}{m^2}$, temos que cada nível N_i (exceto talvez o último) é tal que $S(N_i) \geq \frac{m-2}{m}$. Assim, o seguinte resultado (dado em [41]) pode ser obtido aplicando-se o Lema 5.2.1.

Lema 5.3.5. *Se $\mathcal{P}$ é um empacotamento gerado pelo Algoritmo LL_m para uma instância $L \subseteq \mathcal{C}_m$, então $H(\mathcal{P}) \leq \left(\frac{m}{m-2}\right) V(L) + Z$.*

Usando o Algoritmo BI_m, apresentado na Seção 4.4.4 para o PEP, é fácil desenvolver sua versão para o PET (PET^r). Vamos chamar esta versão adaptada ao PET como $\mathrm{BI}_m^{(t)}$. Considerando os passos do Algoritmo $\mathrm{BI}_m^{(t)}$ vamos descrever as alterações necessárias para adaptá-lo ao PET. O Algoritmo $\mathrm{BI}_m^{(t)}$ faz a mesma partição da lista de entrada em sublistas como feito no Algoritmo BI_m. Nos passos 2 e 3, o Algoritmo $\mathrm{BI}_m^{(t)}$ aplica o Algoritmo NFDH. No passo 4 a lista L_6 é ordenada em ordem não-crescente de altura, e só depois é feita sua divisão em sublistas. Cada sublista é empacotada em apenas um nível usando-se o Algoritmo $\mathrm{NFDH}^{(p)}$, sendo que o empacotamento de L_6 é a concatenação dos níveis gerados desta forma. A concatenação dos empacotamentos é feita conforme definida para o PET. Com isso, é fácil provar os seguintes resultados, apresentados em [46].

Lema 5.3.6. *Para toda lista de caixas $L \subset \mathcal{C}^{xy}\left[0, \frac{1}{m}\ ;\ 0, \frac{1}{m}\right]$, $m \geq 2$, onde nenhuma caixa tem altura maior que Z, tem-se que*

$$\mathrm{BI}_m^{(t)}(L) \leq \left(\frac{m+1}{m}\right)^2 V(L) + 6Z.$$

Prova. Pelas inequações (4.3)—(4.6), temos que os empacotamentos parciais de L têm garantia de área de pelo menos $\left(\frac{m}{m+1}\right)^2$. O lema segue usando esta garantia de área com o Lema 5.2.1. □

Corolário 5.3.7. *Para toda lista de caixas* $L \subset \mathcal{C}^{xy}\left[0, \frac{1}{m}\ ;\ 0, \frac{1}{m}\right]$, $m \geq 2$, *onde nenhuma caixa tem altura maior que* Z, *tem-se que*

$$\mathrm{BI}_m^{(t)}(L) \leq \left(\frac{m+1}{m}\right)^2 \mathrm{OPT}(L) + 6Z.$$

Um algoritmo que terá um papel importante, no desenvolvimento dos algoritmos do caso geral do PET, é o Algoritmo COLUMN. Este algoritmo gera um empacotamento parcial de duas listas de caixas L_1 e L_2. O empacotamento consiste de várias pilhas de caixas, as quais referimos como *colunas*. Cada coluna é construída colocando-se uma caixa no topo de outra, e cada coluna consiste de caixas de somente uma das listas L_1 ou L_2.

O Algoritmo COLUMN é chamado com os parâmetros $(L_1, L_2, [p^1], [p^2])$, onde $[p^1] = p_1^1$, $p_2^1, \ldots, p_{n_1}^1$ indica as posições no fundo da caixa B onde as colunas de caixas de L_1 podem começar e $[p^2] = p_1^2, p_2^2, \ldots, p_{n_2}^2$ indica as posições no fundo da caixa B onde as colunas de caixas de L_2 podem começar. Cada ponto $p_j^i = (x_j^i, y_j^i)$ representa as coordenadas nos eixos x e y onde a primeira caixa (se existir) de cada coluna da respectiva lista deve ser empacotada. Note que a coordenada do eixo z não precisa ser especificada, já que podemos assumir como 0 (correspondendo ao fundo da caixa B). Vamos assumir que $[p^1]$, $[p^2]$ e as listas L_1 e L_2 são escolhidas de tal modo que elas não geram empacotamentos inviáveis.

Vamos chamar de *altura da coluna* a soma das alturas de todas as caixas dessa coluna.

Inicialmente, todas as $n_1 + n_2$ colunas começam vazias no fundo da caixa B. Em cada iteração, o algoritmo escolhe uma coluna com menor altura e empacota a próxima caixa da correspondente lista no topo dessa coluna. O processo termina quando todas as caixas da lista L_1 ou todas as caixas da lista L_2 tiverem sido empacotadas (o que ocorrer primeiro). Neste momento, o algoritmo retorna um par $(\mathcal{P}, L')$, onde L' consiste das caixas de $L_1 \cup L_2$ que foram empacotadas, e $\mathcal{P}$ é o empacotamento de L' gerado pelo algoritmo. Dizemos que o Algoritmo COLUMN *combina* as listas L_1 e L_2.

Se cada caixa da lista L_i tem área de fundo pelo menos s_i $(i = 1, 2)$, a soma $n_1 s_1 + n_2 s_2$ é chamada de *área combinada* do empacotamento gerado pelo Algoritmo COLUMN. Denotaremos por $\mathrm{COLUMN}^{(t)}$ o Algoritmo COLUMN chamado com os parâmetros $(L \cap \mathcal{T}_1, L \cap \mathcal{T}_2, [p^1], [p^2])$.

O seguinte resultado sobre este algoritmo será usado.

Lema 5.3.8. *Seja* $\mathcal{P}$ *um empacotamento de* $L' \subseteq L_1 \cup L_2$ *gerado pelo Algoritmo COLUMN quando aplicado às listas* L_1 *e* L_2, *e lista de posições* $p_1^i, p_2^i, \ldots, p_{n_i}^i$ $(i = 1, 2)$. *Se* $S(b) \geq s_i$, *para cada caixa* b *em* L_i $(i = 1, 2)$, *então* $H(\mathcal{P}) \leq \frac{1}{s_1 n_1 + s_2 n_2} V(L') + Z$.

Prova. Note que a diferença de altura de quaisquer duas colunas não é maior que Z. Assim, $V(L') \geq (H(\mathcal{P}) - Z)(s_1 n_1 + s_2 n_2)$. □

Usando o Algoritmo COLUMN para combinar listas de caixas críticas, podemos adaptar o Algoritmo STP_m (do PEP) para o PET. Vamos chamar esta adaptação de $\mathrm{STP}_m^{(t)}$. A adaptação segue como feito na adaptação do Algoritmo BI_m para o Algoritmo $\mathrm{BI}_m^{(t)}$. O Algoritmo COLUMN é usado para combinar as caixas críticas da mesma forma como foi feito no Algoritmo COMB (para gerar o empacotamento $\mathcal{P}_{ABC}$); e o empacotamento $\mathcal{P}_1$ (gerado no passo 5) é construído gerando-se m^2 colunas, empacotando uma caixa em uma coluna mais baixa em cada iteração. Desta forma, teremos que $\mathcal{P}_1 \| \mathcal{P}_{ABC}$ é um empacotamento assintoticamente ótimo. Com isto, podemos provar o seguinte resultado para este algoritmo.

Teorema 5.3.9. *Para qualquer lista L de retângulos, com dimensões no fundo menores ou iguais a $\frac{1}{m}$,*

$$\mathrm{STP}_m^{(t)}(L) \leq \alpha_m \cdot \mathrm{OPT}(L) + (4k - 2m + 8)Z,$$

onde $\alpha_m = (2m^3 + 5m^2 + 5m + 2 + \sqrt{9m^4 + 34m^3 + 41m^2 + 20m + 4})/(2m(m+1)^2)$.

Este resultado melhora os limites anteriores de $\frac{m+1}{m-1}$ de Li e Cheng [40] e $\left(\frac{m+1}{m}\right)^2$ de Miyazawa [46]. A Tabela 4.1 apresenta os valores de α_m para m entre 1 e 10.

5.3.3 Empacotamento *On-Line* de Caixas com Fundo Pequeno

Sem perda de generalidade, iremos considerar que o empacotamento é feito em uma caixa $B = (1, 1, \infty)$. Se este não for o caso, basta reparametrizar a instância.

Além do valor de m, este algoritmo também dependerá de um valor p, $0 < p < 1$, que será usado para *arredondar* a altura das caixas de forma a restringir os diferentes valores de alturas possíveis. As caixas de alturas iguais serão empacotadas usando um algoritmo de empacotamento bidimensional *on-line*.

Este algoritmo tem um limite de desempenho assintótico $\frac{1}{p}\left(\frac{m+1}{m}\right)^2$. O fator $\frac{1}{p}$ é devido ao arredondamento feito na altura das caixas e o fator $\left(\frac{m+1}{m}\right)^2$ é devido à garantia de área associada aos empacotamentos gerados pelo algoritmo de empacotamento bidimensional em placas.

Algoritmo $\mathrm{SRR}^{(3)}_{m,p}$

Entrada: Lista de caixas $L = (b_1, \ldots, b_n)$, $b_i \in \mathcal{C}^{xy}\left[0, \frac{1}{m}\ ;\ 0, \frac{1}{m}\right]$.

Saída: Empacotamento das caixas de L em uma caixa $B = (1, 1, \infty)$.

1 $\mathcal{P} \leftarrow \emptyset$.

2 Para $i = 1$ até n faça

2.1 Seja $j \geq 0$ tal que b_i é uma j-caixa.

2.2 Seja $\mathcal{N}_j$ o conjunto de j-níveis gerados até o momento.

2.3 Use o algoritmo $\mathrm{OFF}^{(2)}_m$ para empacotar b_i nos níveis $\mathcal{N}_j$, visualizando cada nível de $\mathcal{N}_j$ como uma placa $(1, 1)$ e b_i como um retângulo $(x(b_i), y(b_i))$. Caso necessário, crie um novo j-nível em $\mathcal{P}$ para empacotar b_i.

3 Retorne $\mathcal{P}$.

Fim algoritmo.

Teorema 5.3.10. *Seja m um inteiro $m \geq 2$, e p um racional positivo menor que 1. Seja L uma lista de caixas $L \subset \mathcal{C}_m$ a serem empacotadas em $B = (1, 1, \infty)$. Então, o Algoritmo $\mathrm{SRR}^{(3)}_{m,p}$ para o PET é tal que*

$$\mathrm{SRR}^{(3)}_{m,p}(L) \leq \frac{1}{p}\left(\frac{m+1}{m}\right)^2 \cdot \mathrm{OPT}(L) + \mathcal{O}\left(\frac{m^2}{1-p}\right).$$

Prova. Seja L_i as i-caixas de L e N_i o número de i-níveis em um empacotamento $\mathcal{P}$ gerado pelo Algoritmo $\mathrm{SRR}^{(3)}_{m,p}$. Então temos que

$$\begin{aligned} V(L) &\geq \sum_{i\geq 0} V(L_i) \\ &\geq \sum_{i\geq 0} p^{i+1} S(L_i) \\ &\geq p\sum_{i\geq 0} p^i \left[\left(\frac{m}{m+1}\right)^2 N_i - \mathcal{O}(m^2)\right] \text{ (pelo Lema 4.4.7)} \\ &= p\left[\left(\frac{m}{m+1}\right)^2 \sum_{i\geq 0} p^i N_i - \mathcal{O}(m^2)\sum_{i\geq 0} p^i\right]. \end{aligned}$$

Ou seja,

$$H(\mathcal{P}) \leq \frac{1}{p}\left(\frac{m+1}{m}\right)^2 V(L) + \mathcal{O}\left(\frac{m^2}{1-p}\right). \tag{5.1}$$

□

As análises feitas para este algoritmo se baseiam apenas em comparação do empacotamento ótimo com o volume. Observe que rotações não são efetuadas e portanto, temos um algoritmo para o PET^r com o mesmo limite de desempenho assintótico.

Usando-se o esquema de arredondamento na altura das caixas, como feito para o Algoritmo $\mathrm{SRR}^{(3)}_{m,p}$, e considerando empacotamentos gerados da mesma forma como feito para o Algoritmo IOFF_m, do PEP, podemos construir um algoritmo *on-line* para este problema com limite de desempenho assintótico tão próximo do limite de desempenho assintótico do Algoritmo IOFF_m quanto se queira. A estratégia é empacotar as caixas de altura p^i em níveis de altura p^i. O empacotamento em níveis é feito pelo Algoritmo IOFF_m, do PEP. Com isso, o empacotamento final é constituído de duas partes, um empacotamento próximo do ótimo (assintoticamente) com garantia de área $\frac{1}{p}\left(\frac{m}{m+1}\right)^2$ e outro com garantia de área $\frac{1}{p}\left(\frac{m}{m+2}\right)$. Chamamos este algoritmo de $\mathrm{ISRR}_{m,p}$. Assim, o seguinte resultado é válido para este algoritmo.

Teorema 5.3.11. *Para qualquer lista L de caixas, com dimensões no fundo menores ou iguais a $\frac{1}{m}$, e dado p, $0 < p < 1$, temos que*

$$\mathrm{ISRR}_{m,p}(L) \leq \alpha_{m,p} \cdot \mathrm{OPT}(L) + \mathcal{O}\left(\frac{m^2}{1-p}\right),$$

onde $\lim_{p\to 1} \alpha_{m,p} = \alpha(\mathrm{ISRR}_{m,p}) \leq \left(\frac{m+2}{m+1}\right)^2 + \frac{2}{m(m+1)}$.

5.3.4 Algoritmo TRI_k

Nesta seção apresentamos um algoritmo que desenvolvemos para o PET —chamado TRI_k— com limite de desempenho assintótico não superior a 2,67.

Primeiramente, apresentaremos um outro algoritmo para o PET que será usado como subrotina no Algoritmo TRI_k. Este algoritmo é baseado no Algoritmo UD, desenvolvido por Baker *et al* [2] para o PEF.

Vejamos inicialmente a versão que chamamos de UD^x. Dada uma lista $L = (b_1, b_2, \ldots, b_n)$ de caixas $b_i = (x_i, y_i, z_i)$, o Algoritmo UD^x primeiro usa o Algoritmo UD para gerar um empacotamento $\mathcal{B}$, aplicando-o a uma lista de retângulos $R = (r_1, r_2, \ldots, r_n)$, onde $r_i = (x_i, z_i)$, $i = 1, \ldots, n$. Então ele gera um empacotamento de L empacotando as caixas correspondentes na posição 0 no eixo y e usando as mesmas coordenadas do empacotamento bidimensional $\mathcal{B}$ para os eixos x e y.

O Algoritmo UD^y é a versão simétrica a do Algoritmo UD^x.

Usando o Teorema 3.3.5 é fácil provar o seguinte resultado.

Lema 5.3.12. *Seja L uma instância para o PET tal que $y(b) > \frac{1}{2}$ (resp. $x(b) > \frac{1}{2}$) para cada caixa b em L. Então o empacotamento $\mathcal{P}$ gerado pelo Algoritmo* UD^x *(resp.* UD^y*) é tal que*

$$H(\mathcal{P}) \leq \frac{5}{4}\mathrm{OPT}(L) + \frac{53}{8}Z.$$

Daremos a seguir uma idéia do Algoritmo TRI_k. Este algoritmo é parecido com o Algoritmo $\mathrm{BI}_{k,\epsilon}$ e também depende de um parâmetro k, $k > 5$. Este é o mesmo k usado na Definição 4.4.1.

Este algoritmo divide a lista de entrada L em sublistas e aplica um algoritmo apropriado para cada uma (ou uma combinação) destas sublistas. O empacotamento final é obtido como uma concatenação destes empacotamentos.

Inicialmente, as caixas de L são divididas em quatro partes: P_1, P_2, P_3 e P_4, onde $P_i := L \cap \wp_i$, $i = 1, \ldots, 4$ (veja a Figura 5.2).

Suponha que para cada uma destas partes geramos empacotamentos consistindo de níveis. Li e Cheng [40] mostraram que é possível gerar um empacotamento de cada uma das partes P_1, P_2 e P_3 com garantia de área $\frac{1}{3}$. Note que para a parte P_4 a melhor garantia de área possível é $\frac{1}{4}$. A garantia para as partes P_1, P_2 e P_3 foram obtidas considerando a subdivisão indicada na Figura 5.2.

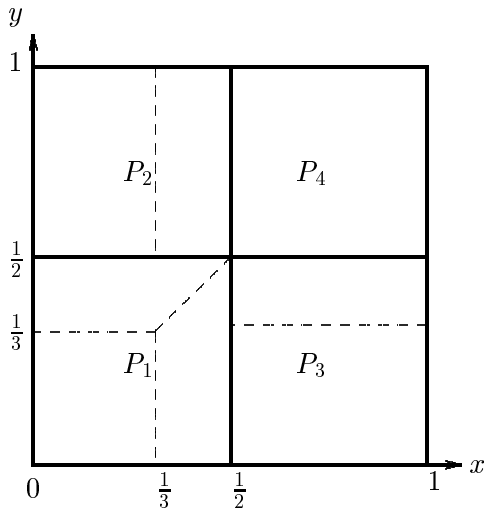


Figura 5.2: Subdivisão de P_1, P_2 e P_3.

Em geral quando temos caixas com fundo *pequeno*, podemos melhorar a garantia de área. Como vimos, usando-se o Algoritmo $\mathrm{BI}_2^{(t)}$, podemos ter um empacotamento de P_1 com garantia de área $\frac{4}{9}$, que é bem melhor que $\frac{1}{3}$. Assim, com respeito à garantia de área, vamos classificar os empacotamentos da parte P_1 como sendo *bom*, das partes P_2 e P_3 como *regular* e da parte P_4 como *ruim*. Chamaremos um empacotamento de uma sublista como sendo *bom* se ele tem uma garantia de área perto daquela da parte P_1. A idéia do algoritmo TRI_k é refinar a subdivisão

das partes de modo que as sublistas obtidas fornecam uma área combinada melhor ou uma garantia de área melhor. Para isto, vamos chamar de *caixas críticas* algumas das caixas que geram empacotamentos com pouca garantia de área.

O Algoritmo TRI_k usa o Algoritmo COMBINE-AB_k^{xy} para combinar as caixas críticas em P_2 e P_3 (estas são as caixas nos conjuntos $L_A = (A_1 \cup \ldots \cup A_{k+14})$ e $L_B = (B_1 \cup \ldots \cup B_{k+14})$ ilustradas na Figura 4.11) de tal modo que o empacotamento parcial resultante seja um empacotamento bom e as caixas críticas de P_2 e P_3 que não foram empacotadas, permaneçam em apenas uma destas partes. Mais ainda, a outra parte —agora sem caixas críticas— permita um empacotamento com boa garantia de área.

Suponha que depois deste processo, todas as caixas críticas em P_3 (caixas no conjunto L_B) foram empacotadas (veja Figura 4.12). O modo como o conjunto L_B é definido garante que as caixas restantes da parte P_3 (sublistas L_1 a L_{17}, veja a Figura 5.3) tenham boa garantia de área. Agora aplique o mesmo processo para as partes $P_1 \cup P_3$ com as caixas críticas de P_4 (estas são caixas em $L'_D \cup L''_D$ e L_C, veja a Figura 4.12). Note que a escolha das sublistas a serem combinadas deve ser cuidadosamente feita: as sublistas devem garantir boa área combinada; e uma vez que uma das sublistas é empacotada, as caixas restantes na correspondente parte P_i também devem fornecer bom empacotamento.

Suponha que L_C é totalmente empacotada (veja Figura 5.3). Agora, defina novas caixas críticas em P_2 e P_4 (estas são as caixas em $L'_F \cup L''_F$ e L_E, veja Figura 5.4) e aplique o Algoritmo COLUMN para as correspondentes sublistas. O empacotamento resultante $\mathcal{P}_{EF}$ tem uma garantia de área melhor do que se considerarmos somente caixas de P_4.

Em ambos os casos, considerando caixas ainda não empacotadas, podemos obter empacotamento que podem ser comparados com um empacotamento ótimo da sublista correspondente. Os detalhes deste procedimento ficarão claros na descrição do Algoritmo TRI_k.

As sublistas A_i e B_i que mencionamos são construídas usando os valores r_i e s_i, $(i = 1, \ldots, k+14)$, especificados na Definição 4.4.1. Estas sublistas estão indicadas na Figura 4.11, e são formalmente definidas no passo 2 do algoritmo.

As coordenadas onde as colunas de A_i e B_j são construídas são especificadas pelas posições $p_{i,j}, q_{i,j}, p'_j, q'_j, p''_j$ e q''_j, dadas na Definição 4.4.2.

Agora estamos prontos para apresentar o Algoritmo TRI_k. A descrição deste algoritmo é parecida com a do Algoritmo $\text{BI}_{k,\epsilon}$. Note que este algoritmo define mais conjuntos críticos, L_E e L_F, aumentando o número de combinações de conjuntos críticos. Note também que o valor de t foi recalculado para este algoritmo.

Algoritmo TRI_k

Entrada: Lista de caixas $L = (b_1, b_2, \ldots, b_n)$.

Saída: Empacotamento $\mathcal{P}$ de L em $B = (1, 1, \infty)$.

1 Seja $P_i \leftarrow L \cap \wp_i$, $i = 1, \ldots, 4$.

2 $\mathcal{P}_{AB} \leftarrow \mathrm{COMBINE\text{-}AB}_k^{xy}(L, \mathrm{ftype}, \mathcal{C}_1, \mathcal{C}_1, \mathrm{COLUMN}))$.

Atualize(L).

3 Se $\mathrm{ftype}(L, \mathcal{A}_k^{xy}) = \emptyset$ então

3.1 Seja L_{AB} o conjunto das caixas empacotadas em $\mathcal{P}_{AB}$.

3.2 Seja $t = 0{,}46588$. Subdivida a lista L em $L_1, \ldots, L_{25}$ como segue (veja a Figura 4.12).

$L_i \leftarrow L \bigcap \mathcal{C}^{xy}\left[\frac{1}{2}, 1\ ;\ \frac{1}{i+2}, \frac{1}{i+1}\right]$, para $i = 1, \ldots, 16$ $\quad L_{17} \leftarrow L \bigcap \mathcal{C}^{xy}\left[\frac{1}{2}, 1\ ;\ 0, \frac{1}{18}\right]$,

$L_{18} \leftarrow L \bigcap \mathcal{C}^{xy}\left[\frac{1}{3}, \frac{1}{2}\ ;\ \frac{1}{3}, \frac{1}{2}\right]$, $\quad L_{19} \leftarrow L \bigcap \mathcal{C}^{xy}\left[\frac{1}{3}, \frac{1}{2}\ ;\ \frac{1}{4}, \frac{1}{3}\right]$,

$L_{20} \leftarrow L \bigcap \mathcal{C}^{xy}\left[\frac{1}{3}, \frac{1}{2}\ ;\ 0, \frac{1}{4}\right]$, $\quad L_{21} \leftarrow L \bigcap \mathcal{C}^{xy}\left[\frac{1}{4}, \frac{1}{3}\ ;\ \frac{1}{3}, \frac{1}{2}\right]$,

$L_{22} \leftarrow L \bigcap \mathcal{C}^{xy}\left[\frac{1}{4}, \frac{1}{3}\ ;\ 0, \frac{1}{3}\right]$, $\quad L_{23} \leftarrow L \bigcap \mathcal{C}^{xy}\left[0, \frac{1}{4}\ ;\ \frac{1}{3}, \frac{1}{2}\right]$,

$L_{24} \leftarrow L \bigcap \mathcal{C}^{xy}\left[0, \frac{1}{4}\ ;\ \frac{1}{4}, \frac{1}{3}\right]$, $\quad L_{25} \leftarrow L \bigcap \mathcal{C}_4$,

$L_C \leftarrow L \bigcap \mathcal{C}^{xy}\left[\frac{1}{2}, 1\ ;\ \frac{1}{2}, 1 - t\right]$ $\quad L'_D \leftarrow \{b \in L_1 :\ y(b) \leq t\}$,

$L''_D \leftarrow \{b \in L_{18} :\ y(b) \leq t\}$. $\quad L_D \leftarrow L'_D \bigcup L''_D$.

3.3 Gere empacotamentos $\mathcal{P}_1, \ldots, \mathcal{P}_{25}$ como segue.

$(\mathcal{P}_{CD'}, L_{CD'}) \leftarrow \mathrm{COLUMN}(L_C, [(0,0)], L'_D, [(0, 1-t)])$;
$(\mathcal{P}_{CD''}, L_{CD''}) \leftarrow \mathrm{COLUMN}(L_C \setminus L_{CD'}, [(0,0)], L''_D, [(0, 1-t), (\frac{1}{2}, 1-t)])$;
$\mathcal{P}_{CD} \leftarrow \mathcal{P}_{CD'} \| \mathcal{P}_{CD''}$;
$L_{CD} \leftarrow L_{CD'} \bigcup L_{CD''}$;
$L_1 \leftarrow L_1 \setminus L_{CD}$;
$L_{18} \leftarrow L_{18} \setminus L_{CD}$;

3.4 $\mathcal{P}_i \leftarrow \mathrm{NFDH}^y(L_i)$ para $i = 1, \ldots, 22$;
$\mathcal{P}_i \leftarrow \mathrm{NFDH}^x(L_i)$ para $i = 23, 24$;
$\mathcal{P}_{25} \leftarrow \mathrm{LL}(L_{25}, 4)$;

3.5 $P'_1 \leftarrow P_1 \setminus L_{CD}$;
$P'_2 \leftarrow P_2 \setminus L_{AB}$;
$P'_3 \leftarrow P_3 \setminus (L_{AB} \cup L_{CD})$;
$P'_4 \leftarrow P_4 \setminus L_{CD}$.

3.6 Se $L_C \subseteq L_{CD}$
então (Caso 1) $p \leftarrow 0{,}441328$ /*L_C é empacotado */ (veja a Figura 5.3)
senão (Caso 2) $p \leftarrow 0{,}451600$; /*L_D é empacotado*/ (veja a Figura 5.4)

3.7 $L_E \leftarrow \{b \in P'_4 : \ x(b) \leq 1-p\}$; $\quad L'_F \leftarrow \left\{b \in P'_2 : \ \frac{1}{9} < x(b) \leq p\right\}$;
$L''_F \leftarrow \left\{b \in P'_2 : \ \frac{1}{18} < x(b) \leq \frac{1}{9}\right\}$; $\quad L_F \leftarrow L'_F \cup L''_F$;

3.8 $(\mathcal{P}_{EF'}, L_{EF'}) \leftarrow \text{COLUMN}(L_E, [(0,0)], L'_F, [(1-p,0)])$;
$(\mathcal{P}_{EF''}, L_{EF''}) \leftarrow \ \text{COLUMN}(L_E \setminus L_{EF'}, [(0,0)], L''_F, [(1-p,0), (1-p+\frac{1}{9},0),$
$\ldots, (1-p+(\lfloor 9p \rfloor - 1)\frac{1}{9}, 0)])$;
$\mathcal{P}_{EF} \leftarrow \mathcal{P}_{EF'} \| \mathcal{P}_{EF''}$;
$L_{EF} \leftarrow L_{EF'} \cup L_{EF''}$;
$P''_2 \leftarrow P'_2 \setminus L_{EF}$;
$P''_4 \leftarrow P'_4 \setminus L_{EF}$.

3.9 (Subcaso 1.) Se $L_E \subseteq L_{EF}$ então /* L_E é totalmente empacotado */

$\mathcal{P}_{UD} \leftarrow \text{UD}^x(P'_2 \cup P'_4)$;
$\mathcal{P}_{OC} \leftarrow \text{OC}(P''_4)$;
$P''_{2e} \leftarrow \left\{b \in P''_2 : \ x(b) \leq \frac{1}{3}\right\}$;
$P''_{2d} \leftarrow \left\{b \in P''_2 : \ x(b) > \frac{1}{3}\right\}$;
$\mathcal{P}_{2e} \leftarrow \text{NFDH}^x(P''_{2e})$;
$\mathcal{P}_{2d} \leftarrow \text{NFDH}^x(P''_{2d})$;
$\mathcal{P}' \leftarrow \mathcal{P}_{OC} \| \mathcal{P}_{2e} \| \mathcal{P}_{2d} \| \mathcal{P}_{EF}$;
$\mathcal{P}'' \leftarrow \{\mathcal{P} \in \{\mathcal{P}_{UD}, \mathcal{P}'\} : \ H(\mathcal{P}) \text{ é mínimo } \}$;
$\mathcal{P}_{aux} \leftarrow \mathcal{P}_{AB} \| \mathcal{P}_{CD} \| \mathcal{P}_1 \| \ldots \| \mathcal{P}_{25}$;
$\mathcal{P} \leftarrow \mathcal{P}_{aux} \| \mathcal{P}''$.

3.10 (Subcaso 2.) Se $L_F \subseteq L_{EF}$ então /* L_F é totalmente empacotado */

$\mathcal{P}_{OC} \leftarrow \text{OC}(P''_4)$;
$P''_{2e} \leftarrow \left\{b \in P''_2 : \ x(b) \leq \frac{1}{18}\right\}$;
$P''_{2d} \leftarrow \{b \in P''_2 : \ x(b) > p\}$;
$\mathcal{P}_{2e} \leftarrow \text{NFDH}^x(P''_{2e})$;
$\mathcal{P}_{2d} \leftarrow \text{NFDH}^x(P''_{2d})$;
$\mathcal{P}' \leftarrow \mathcal{P}_{OC} \| \mathcal{P}_{EF}$;
$\mathcal{P}_{aux} \leftarrow \mathcal{P}_{AB} \| \mathcal{P}_{CD} \| \mathcal{P}_{2e} \| \mathcal{P}_{2d} \| \mathcal{P}_1 \| \ldots \| \mathcal{P}_{25}$;
$\mathcal{P} \leftarrow \mathcal{P}_{aux} \| \mathcal{P}'$.

3.11 Retorne $\mathcal{P}$.

4 Senão gere um empacotamento $\mathcal{P}$ de L como no passo 3 (de forma simétrica).

5 Retorne $\mathcal{P}$.

Fim algoritmo.

O próximo teorema nos dá um limite de desempenho assintótico, α_k, do Algoritmo TRI_k quando $k \to \infty$. Depois da prova deste resultado, mostramos que para valores relativamente

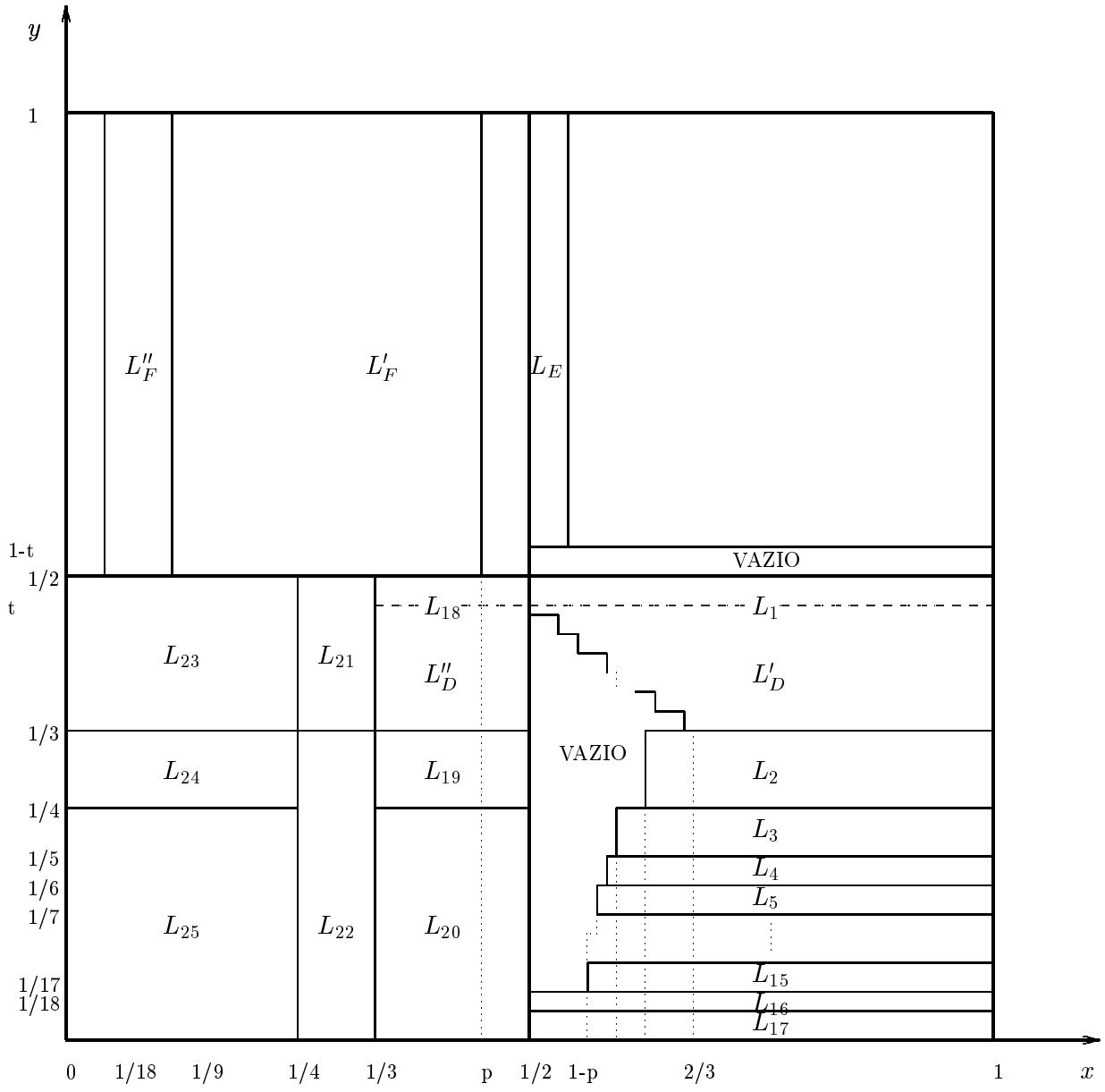


Figura 5.3: Combinação de L_C e $L'_D \cup L''_D$: L_C é totalmente empacotada.

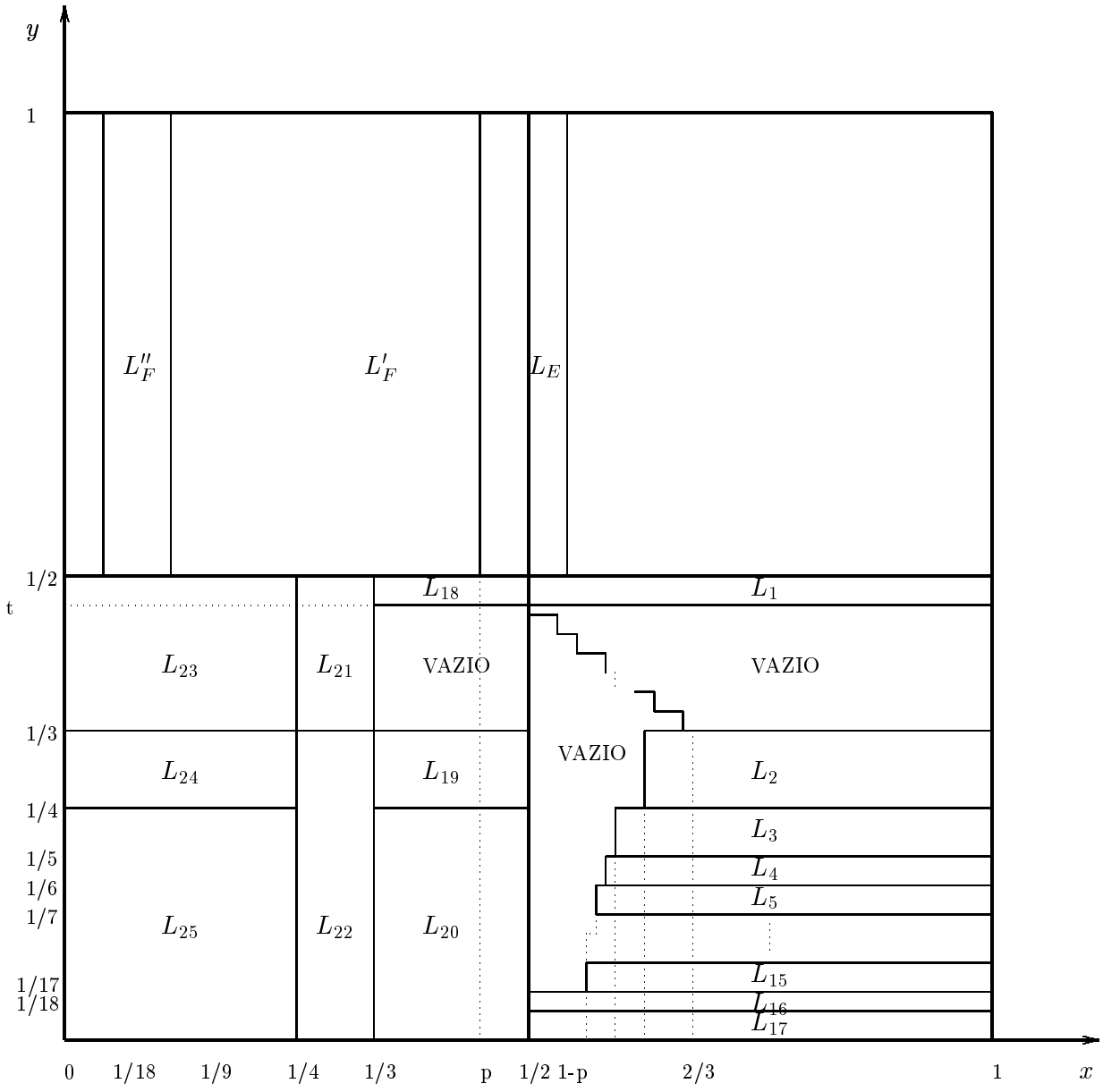


Figura 5.4: Combinação de L_C e $L'_D \cup L''_D$: $L'_D \cup L''_D$ é totalmente empacotada.

pequenos de k o valor de α_k é bem próximo de $\lim_{k\to\infty}\alpha_k$. Esta conclusão seguirá da prova do próximo teorema.

Teorema 5.3.13. *Para qualquer instância L do* PET*, temos*

$$\mathrm{TRI}_k(L) \leq \alpha_k \cdot \mathrm{OPT}(L) + \left(2k + \frac{597}{8}\right) Z,$$

onde $\alpha_k \to 2{,}6608$ à medida que $k \to \infty$.

Prova. Pelo Lema 4.4.14, temos que quando $k \to \infty$ o valor de $r_1^{(k)}$ tende a $\frac{4}{9}$, $r_1 = r_1^{(k)} < \frac{4}{9}$ (veja a Definição 4.4.1). Cada um dos empacotamentos $\mathcal{P}_i$, $i \in \{1,\ldots,25\} \setminus \{1,18\}$, tem uma garantia de área que é pelo menos t. Assim, aplicando o Lema 5.3.2 e o Lema 5.3.5 podemos concluir que

$$H(\mathcal{P}_i) \leq \frac{1}{t} V(L_i) + Z, \quad \text{para } i \in \{1,\ldots,25\} \setminus \{1,18\}. \tag{5.2}$$

Agora, para cada um dos empacotamentos $\mathcal{Q} \in \left\{\mathcal{P}_{i,j}, \tilde{\mathcal{P}'_{i,j}}, \tilde{\mathcal{P}''_{i,j}}\right\}$ que são usados para gerar o empacotamento $\mathcal{P}_{AB}$ no passo 2, $H(\mathcal{Q}) \leq \frac{56}{27} V(\mathcal{Q}) + Z$. Para ver isto, aplique o Lema 5.3.8 junto com o fato de que para cada empacotamento $\mathcal{Q}$ que combina conjuntos L_A e L_B, a área combinada é pelo menos $\frac{27}{56}$. Como existe um máximo de $(2k-1)+28+14 = 2k+41$ empacotamentos gerados de L_A e L_B, podemos ter que $H(\mathcal{P}_{AB}) \leq \frac{56}{27} V(L_{AB}) + (2k+41)Z$. Assim, a seguinte inequação é válida:

$$H(\mathcal{P}_{AB}) \leq \frac{1}{t} V(L_{AB}) + (2k+41)Z. \tag{5.3}$$

Para os empacotamentos $\mathcal{P}_{CD'}$ e $\mathcal{P}_{CD''}$ (no passo 3.3), a área combinada é pelo menos $(\frac{1}{4}+\frac{r_1}{2})$. Com isso, segue do Lema 5.3.8 que

$$H(\mathcal{P}_{CD}) \leq \frac{1}{\left(\frac{1}{4}+\frac{r_1}{2}\right)} V(L_{CD}) + 2Z. \tag{5.4}$$

Vamos agora analisar os dois casos possíveis (*cf.* passo 3.6).

Caso 1. $L_C \subseteq L_{CD}$ e $p = 0{,}441328$.

Para os empacotamentos $\mathcal{P}_1$ e $\mathcal{P}_{18}$ valem as seguintes inequações:

$$H(\mathcal{P}_1) \leq \frac{1}{r_1} V(L_1) + Z, \tag{5.5}$$

$$H(\mathcal{P}_{18}) \leq \frac{1}{4/9} V(L_{18}) + Z. \tag{5.6}$$

Como a garantia de área de cada um dos empacotamentos $\mathcal{P}_{EF'}$ e $\mathcal{P}_{EF''}$ é pelo menos $\frac{3}{10}$, podemos concluir que

$$H(\mathcal{P}_{EF}) \leq \frac{10}{3} V(L_{EF}) + 2Z. \tag{5.7}$$

Subcaso 1.1. $L_E \subseteq L_{EF}$

Pelo Lema 5.3.12,

$$H(\mathcal{P}_{UD}) \leq \frac{5}{4}\mathrm{OPT}(P_2' \cup P_4') + \frac{53}{8}Z. \tag{5.8}$$

Aplicando o Lema 5.3.3, como $S(b) \geq (1-p)(1-t)$ para $b \in P_4''$, segue que

$$H(\mathcal{P}_{OC}) \leq \frac{1}{(1-p)(1-t)}V(P_4''). \tag{5.9}$$

Para os empacotamentos $\mathcal{P}_{2e}$ e $\mathcal{P}_{2d}$, usando o Lema 5.3.2, podemos concluir que

$$H(\mathcal{P}_{2e}\|\mathcal{P}_{2d}) \leq \frac{1}{1/3}V(P_{2e}'' \cup P_{2d}'') + 2Z. \tag{5.10}$$

Das inequações (5.7), (5.9), (5.10) e da igualdade $(1-p)(1-t) = \min\{\frac{3}{10}, (1-p)(1-t), \frac{1}{3}\}$ segue que

$$\begin{aligned} H(\mathcal{P}') &= H(\mathcal{P}_{OC}\|\mathcal{P}_{2e}\|\mathcal{P}_{2d}\|\mathcal{P}_{EF}) \\ &\leq \frac{1}{(1-p)(1-t)}V\left(P_4'' \cup P_{2e}'' \cup P_{2d}'' \cup L_{EF}\right) + 4Z \\ &= \frac{1}{(1-p)(1-t)}V\left(P_2' \cup P_4'\right) + 4Z. \end{aligned} \tag{5.11}$$

Como $\mathcal{P}'' = \{\mathcal{P} \in \{\mathcal{P}_{UD}, \mathcal{P}'\} : \ H(\mathcal{P}) \text{ é mínimo }\}$, temos

$$H(\mathcal{P}'') = \min\{H(\mathcal{P}_{UD}), H(\mathcal{P}')\}. \tag{5.12}$$

Para o empacotamento $\mathcal{P}_{aux} = \mathcal{P}_{AB}\|\mathcal{P}_{CD}\|\mathcal{P}_1\|\ldots\|\mathcal{P}_{25}$, usando as inequações (5.3)—(5.6) e o fato de que $r_1 = \min\left\{t, r_1, \frac{1}{4} + \frac{r_1}{2}, \frac{4}{9}\right\}$, obtemos

$$H(\mathcal{P}_{aux}) \leq \frac{1}{r_1}V(L_{aux}) + (2k+68)Z, \tag{5.13}$$

onde L_{aux} denota o conjunto de caixas no empacotamento $\mathcal{P}_{aux}$.

Sejam

$$\mathcal{H}_1 := H(\mathcal{P}'') - \frac{53}{8}Z, \tag{5.14}$$

$$\mathcal{H}_2 := H(\mathcal{P}_{aux}) - (2k+68)Z. \tag{5.15}$$

Das inequações (5.8) e (5.12), temos

$$\mathcal{H}_1 \leq \frac{5}{4}\mathrm{OPT}(P_2' \cup P_4'),$$

e portanto,

$$\mathrm{OPT}(P_2' \cup P_4') \geq \frac{4}{5}\mathcal{H}_1.$$

Assim,

$$\mathrm{OPT}(L) \geq \mathrm{OPT}(P_2' \cup P_4') \geq \frac{4}{5}\mathcal{H}_1. \tag{5.16}$$

Note que das inequações (5.13) e (5.15) podemos concluir que

$$V(L_{aux}) \geq r_1\mathcal{H}_2. \tag{5.17}$$

Por outro lado, de (5.12) e (5.11) temos

$$\begin{aligned} H(\mathcal{P}'') &\leq H(\mathcal{P}') \\ &\leq \frac{1}{(1-p)(1-t)}V(P_2' \cup P_4') + 4Z \\ &\leq \frac{1}{(1-p)(1-t)}V(P_2' \cup P_4') + \frac{53}{8}Z, \end{aligned}$$

e portanto,

$$\mathcal{H}_1 = H(\mathcal{P}'') - \frac{53}{8}Z \leq \frac{1}{(1-p)(1-t)}V(P_2' \cup P_4'),$$

ou seja,

$$V(P_2' \cup P_4') \geq (1-p)(1-t)\mathcal{H}_1. \tag{5.18}$$

Como $V(L) = V(L_{aux}) + V(P_2' \cup P_4')$, usando (5.17) e (5.18) obtemos

$$V(L) \geq r_1\mathcal{H}_2 + (1-p)(1-t)\mathcal{H}_1.$$

Assim,

$$\mathrm{OPT}(L) \geq V(L) \geq r_1\mathcal{H}_2 + (1-p)(1-t)\mathcal{H}_1.$$

Combinando (5.16) e a inequação acima, segue que

$$\mathrm{OPT}(L) \geq \max\left\{\frac{4}{5}\mathcal{H}_1, (1-p)(1-t)\mathcal{H}_1 + r_1\mathcal{H}_2\right\}.$$

Como $H(\mathcal{P}) = H(\mathcal{P}_{aux}) + H(\mathcal{P}'')$; usando (5.14) e (5.15), temos

$$\begin{aligned} H(\mathcal{P}) &= \left(\mathcal{H}_2 + (2k+68)\,Z + \mathcal{H}_1 + \frac{53}{8}Z\right) \\ &= \mathcal{H}_1 + \mathcal{H}_2 + \left(2k + \frac{597}{8}\right)Z. \end{aligned}$$

Assim, $\mathrm{TRI}_k(L) \leq \alpha_k'(r_1) \cdot \mathrm{OPT}(L) + \left(2k + \frac{597}{8}\right)Z$, onde $\alpha_k'(r_1) = \frac{4-5(1-p)(1-t)+5r_1}{4r_1}$. Para provar isto, mostramos que $\frac{\mathcal{H}_1+\mathcal{H}_2}{\max\left\{\frac{4}{5}\mathcal{H}_1,(1-p)(1-t)\mathcal{H}_1+r_1\mathcal{H}_2\right\}} \leq \alpha_k'(r_1)$, analisando-se os dois casos onde o denominador é máximo.

Subcaso 1.2. $L_F \subseteq L_{EF}$

Neste caso,

$$H(\mathcal{P}_{OC}) \leq \frac{1}{(1-t)/2} V(P_4''). \tag{5.19}$$

Como $\mathcal{P}' = \mathcal{P}_{OC} \| \mathcal{P}_{EF'} \| \mathcal{P}_{EF''}$ e todos estes empacotamentos combinam caixas em P_4', segue que

$$\mathrm{OPT}(L) \geq \mathrm{OPT}(P_4'' \cup L_{EF}) \geq H(\mathcal{P}_{OC}) + H(\mathcal{P}_{EF}) - 2Z = H(\mathcal{P}') - 2Z. \tag{5.20}$$

Lembrando que $\mathcal{P}' = \mathcal{P}_{OC} \| \mathcal{P}_{EF}$, e usando (5.7) e (5.19) temos

$$H(\mathcal{P}') \leq \frac{1}{(1-t)/2} V(L_{EF} \cup \mathcal{P}_4'') + 2Z. \tag{5.21}$$

Usando o Lema 5.3.2 para os empacotamentos $\mathcal{P}_{2e}$ e $\mathcal{P}_{2d}$ podemos concluir que

$$H(\mathcal{P}_{2e} \| \mathcal{P}_{2d}) \leq \frac{1}{p} V(P_{2e}'' \cup P_{2d}'') + 2Z. \tag{5.22}$$

Das inequações (5.3),..., (5.6) e (5.22) e do fato de que $p = \min\{t, \frac{1}{4} + \frac{r_1}{2}, r_1, \frac{4}{9}, p\}$, temos

$$H(\mathcal{P}_{aux}) \leq \frac{1}{p} V(L_{aux}) + (2k+70)Z. \tag{5.23}$$

Sejam

$$\mathcal{H}_1 := H(\mathcal{P}') - 2Z, \tag{5.24}$$

$$\mathcal{H}_2 := H(\mathcal{P}_{aux}) - (2k+70)Z. \tag{5.25}$$

De (5.20) e (5.24), segue que

$$\mathrm{OPT}(L) \geq \mathcal{H}_1. \tag{5.26}$$

Usando (5.21) e (5.24), resp. (5.23) e (5.25), temos

$$\begin{aligned} V(L_{EF} \cup P_4'') &\geq \frac{(1-t)}{2} \mathcal{H}_1, \\ V(L_{aux}) &\geq p\mathcal{H}_2. \end{aligned}$$

Como $V(L) = V(L_{EF} \cup P_4'' \cup L_{aux})$, somando as inequações acima, obtemos

$$V(L) \geq \frac{(1-t)}{2} \mathcal{H}_1 + p\mathcal{H}_2,$$

e portanto

$$\mathrm{OPT}(L) \geq \frac{(1-t)}{2} \mathcal{H}_1 + p\mathcal{H}_2.$$

Combinando a inequação acima com (5.26) podemos provar que

$$\mathrm{TRI}_k(L) \leq \alpha''_k \cdot \mathrm{OPT}(L) + (2k+72)Z,$$

onde $\alpha''_k = \frac{1+t+2p}{2p}$. Para isto, basta provarmos que $\frac{\mathcal{H}_1+\mathcal{H}_2}{\max\{\mathcal{H}_1, \frac{(1-t)}{2}\mathcal{H}_1+p\mathcal{H}_2\}} \leq \alpha''_k$. A prova pode ser feita analogamente ao caso anterior, e portanto será omitida.

Assim, das análises dos dois subcasos, podemos concluir que

$$\mathrm{TRI}_k(L) \leq \alpha_k \cdot \mathrm{OPT}(L) + \left(2k + \frac{597}{8}\right) Z,$$

onde $\alpha_k \to \alpha'_k(\frac{4}{9}) = \alpha''_k = 2{,}6607\ldots$ à medida que $k \to \infty$.

Caso 2. $L_D \subseteq L_{CD}$ e $p = 0{,}451600$

Neste caso a prova é similar àquela apresentada no Caso 1. Assim, apenas resumiremos a prova.

Como a garantia de área dos empacotamentos $\mathcal{P}_1$ e $\mathcal{P}_{18}$ é pelo menos t, temos

$$H(\mathcal{P}_i) \leq \frac{1}{t}V(L_i) + Z \text{ para } i \in \{1, 18\}. \tag{5.27}$$

Como no caso 1,

$$H(\mathcal{P}_{EF}) \leq \frac{3}{10}V(L_{EF}) + 2Z. \tag{5.28}$$

Subcaso 2.1. $L_E \subseteq L_{EF}$

Pelo Lema 5.3.12,

$$H(\mathcal{P}_{UD}) \leq \frac{5}{4}\mathrm{OPT}(P'_2 \cup P'_4) + \frac{53}{8}Z. \tag{5.29}$$

Como a área de cada caixa de P''_4 é pelo menos $(1-p)\frac{1}{2}$, aplicando-se o Lema 5.3.3, temos

$$H(\mathcal{P}_{OC}) \leq \frac{1}{(1-p)\frac{1}{2}}V(P''_4). \tag{5.30}$$

Do mesmo modo como em (5.10),

$$H(\mathcal{P}_{2e} \| \mathcal{P}_{2d}) \leq \frac{1}{\frac{1}{3}}V(P''_{2e} \cup P''_{2d}) + 2Z. \tag{5.31}$$

De (5.28), (5.30) e (5.31), temos

$$\begin{aligned} H(\mathcal{P}') &= H(\mathcal{P}_{OC} \| \mathcal{P}_{2e} \| \mathcal{P}_{2d} \| \mathcal{P}_{EF}) \\ &\leq \frac{1}{(1-p)\frac{1}{2}}V\left(P'_2 \cup P'_4\right) + 4Z. \end{aligned} \tag{5.32}$$

Como $t = \min\left\{t, \frac{1}{4} + \frac{r_1}{2}\right\}$, de (5.27, (5.3) e (5.4) temos

$$H(\mathcal{P}_{aux}) \leq \frac{1}{t} V(L_{aux}) + (2k + 68)Z. \tag{5.33}$$

Sejam

$$\begin{aligned} \mathcal{H}_1 &:= H(\mathcal{P}'') - \frac{53}{8}Z, \\ \mathcal{H}_2 &:= H(\mathcal{P}_{aux}) - (2k + 68)Z. \end{aligned}$$

Então, de (5.29),

$$\mathrm{OPT}(L) \geq \frac{4}{5}\mathcal{H}_1.$$

Por outro lado, de (5.31) e (5.32), temos

$$\begin{aligned} V(L_{aux}) &\geq t \cdot \mathcal{H}_2 \quad \text{e} \\ V(P_2' \cup P_4') &\geq (1-p)\frac{1}{2}\mathcal{H}_1, \end{aligned}$$

e portanto,

$$\mathrm{OPT}(L) \geq V(L) \geq (1-p)\frac{1}{2}\mathcal{H}_1 + t \cdot \mathcal{H}_2.$$

Assim,

$$\mathrm{OPT}(L) \geq \max\left\{\frac{4}{5}\mathcal{H}_1, (1-p)\,\frac{1}{2}\mathcal{H}_1 + t \cdot \mathcal{H}_2\right\}.$$

Portanto, $\mathrm{TRI}_k(L) \leq \beta_k'(r_1) \cdot \mathrm{OPT}(L) + \left(2k + \frac{597}{8}\right) Z$, onde $\beta_k'(r_1) = \left[\frac{1}{t} - \frac{5(1-p)}{8t} + \frac{5}{4}\right]$. A última inequação segue mostrando que $\frac{\mathcal{H}_1 + \mathcal{H}_2}{\max\left\{\frac{4}{5}\mathcal{H}_1, (1-p)\frac{1}{2}\mathcal{H}_1 + t \cdot \mathcal{H}_2\right\}} \leq \beta_k'(r_1)$.

Subcaso 2.2. $L_F \subseteq L_{EF}$

Do mesmo modo que em (5.20), temos

$$\mathrm{OPT}(L) \geq H(\mathcal{P}') - 2Z. \tag{5.34}$$

Do Lema 5.3.3, temos

$$H(\mathcal{P}_{OC}) \leq \frac{1}{\frac{1}{4}} V(P_4'').$$

Portanto, de (5.28), temos

$$H(\mathcal{P}') \leq \frac{1}{\frac{1}{4}} V(L_{EF} \cup \mathcal{P}_4'') + 2Z. \tag{5.35}$$

Como todas as caixas de L_F foram empacotadas em $\mathcal{P}'$, temos

$$H(\mathcal{P}_{2e} \| \mathcal{P}_{2d}) \leq \frac{1}{p} V(P_{2e}'' \cup P_{2d}'') + 2Z. \tag{5.36}$$

Como $p = \min\left\{\frac{1}{4} + \frac{r_1}{2}, t, p\right\}$, de (5.36), (5.27), (5.3) e (5.4) temos

$$H(\mathcal{P}_{aux}) \leq \frac{1}{p} V(L_{aux}) + (2k + 70)Z. \tag{5.37}$$

Sejam

$$\begin{aligned}\mathcal{H}_1 &:= H(\mathcal{P}') - 2Z,\\ \mathcal{H}_2 &:= H(\mathcal{P}_{aux}) - (2k+70)Z.\end{aligned}$$

Então, de (5.34)—(5.37), temos

$$\begin{aligned}\mathrm{OPT}(L) &\geq \mathcal{H}_1,\\ V(L_{EF} \cup P_4'') &\geq \frac{1}{4}\mathcal{H}_1,\\ V(L_{aux}) &\geq p \cdot \mathcal{H}_2.\end{aligned}$$

Assim,

$$\mathrm{OPT}(L) \geq V(L) \geq \frac{1}{4}\mathcal{H}_1 + p \cdot \mathcal{H}_2.$$

Portanto,

$$\mathrm{TRI}_k(L) \leq \beta_k'' \cdot \mathrm{OPT}(L) + (2k+72)Z,$$

onde $\beta_k'' = \frac{3+4p}{4p}$. A última inequação é provada mostrando que $\frac{\mathcal{H}_1+\mathcal{H}_2}{\max\{\mathcal{H}_1, \frac{1}{4}\mathcal{H}_1 + p\cdot\mathcal{H}_2\}} \leq \frac{3+4p}{4p}$.

Assim, para os valores dados de p e t, podemos concluir que

$$\mathrm{TRI}_k(L) \leq \beta_k \cdot \mathrm{OPT}(L) + \left(2k + \frac{597}{8}\right) Z,$$

onde $\beta_k \to \beta_k'(\frac{4}{9}) = \beta_k'' = 2{,}6607\ldots$ à medida que $k \to \infty$.

Os valores de p e t que consideramos no algoritmo, foram de fato obtidos de forma a termos $\alpha_k'(\frac{4}{9}) = \alpha_k'' = \beta_k'(\frac{4}{9}) = \beta_k''$. Deixaremos ao leitor a verificação deste fato.

O teorema segue das conclusões obtidas nos casos 1 e 2. □

Corolário 5.3.14. *Para qualquer instância L do PET e $k \geq 6$ temos*

$$\mathrm{TRI}_k(L) \leq \gamma_k \cdot \mathrm{OPT}(L) + \left(2k + \frac{597}{8}\right) Z,$$

onde $\gamma_k < 2{,}67$.

Prova. A prova deste resultado segue da prova do teorema anterior. É suficiente observar que para $k \geq 6$ temos $r_1^{(k)} \geq 0{,}44281294$, e portanto todos os argumentos usados na prova continuam válidos. Note que a afirmação do corolário vale tomando-se

$$\gamma_k = \max\left\{\frac{4 - 5(1-p_1)(1-t) + 5r_1}{4r_1}, \frac{1+t+2p_1}{2p_1}, \frac{1}{t} - \frac{5(1-p_2)}{8t} + \frac{5}{4}, \frac{3+4p_2}{4p_2}\right\},$$

onde p_i corresponde ao valor de p no Caso i, $i = 1, 2$. □

Proposição 5.3.15. *O limite de desempenho assintótico do Algoritmo* TRI_k, $k \geq 6$, *está entre* 2,5 *e* 2,67.

Prova. Pelo Teorema 5.3.13 é suficiente provar que 2,5 é um limite inferior para o limite de desempenho assintótico do Algoritmo TRI_k.

Seja L uma instância para o PET, $L = L' \cup L''$, onde $L' = (b'_1, b'_2, \ldots, b'_{2N})$ e $L'' = (b''_1, b''_2, \ldots, b''_{27N})$, e N é um inteiro positivo grande.

Cada caixa b'_i em L', $i = 1, \ldots, 2N$, é definida como

$$b'_i = \left(\frac{1}{2} + \epsilon, \frac{1}{2} + \epsilon, 1\right).$$

Cada caixa b''_i em L'', $i = 1, \ldots, 27N$, é definida como

$$b''_i = \begin{cases} (\delta, \delta, 1 - (i-1)\xi_N) & \text{se } i \bmod 9 = 0 \\ \left(\frac{1}{4} - \epsilon, \frac{1}{4} - \epsilon, 1 - (i-1)\xi_N\right) & \text{caso contrário.} \end{cases}$$

Os valores de ϵ, ξ_N e δ devem ser positivos e muito pequenos, e além disso as seguintes inequações devem valer: $8\left(\frac{1}{4} - \epsilon\right)^2 + \delta^2 \leq \frac{1}{2} + \left(\frac{1}{4}\right)^2$ e $9\left(\frac{1}{4} - \epsilon\right)^2 + \delta^2 > \frac{1}{2} + \left(\frac{1}{4}\right)^2$. Para isto basta fixar um δ pequeno e tomar $\epsilon = \frac{\delta^2}{8}$.

O Algoritmo TRI_k aplicado à lista L gera um empacotamento $\mathcal{P} = \mathcal{P}' \| \mathcal{P}''$ onde $\mathcal{P}'$ (resp. $\mathcal{P}''$) é o empacotamento gerado pelo Algoritmo OC (resp. $\mathrm{LL}(L'', 4)$) aplicado à lista L' (resp. L'').

É imediato que $H(\mathcal{P}') = 2N$. Quanto ao empacotamento $\mathcal{P}''$, este é gerado como segue : $\mathcal{P}''$ consiste de $3N$ níveis, cada um consistindo de 8 caixas do tipo $\left(\frac{1}{4} - \epsilon, \frac{1}{4} - \epsilon, 1 - (i-1)\xi_N\right)$ e uma caixa do tipo $(\delta, \delta, 1 - (i-1)\xi_N)$. Portanto, $H(\mathcal{P}'') = 3N - h(\xi_N)$, onde $h(\xi_N) = 9\xi_N\left(\frac{9N^2 - 3N}{2}\right)$; e assim,

$$H(\mathcal{P}) = 2N + 3N - h(\xi_N) = 5N - h(\xi_N).$$

Um empacotamento melhor $\mathcal{P}^*$ da lista L pode ser obtido gerando:

- $2N$ níveis, cada um consistindo de uma caixa do tipo $\left(\frac{1}{2} + \epsilon, \frac{1}{2} + \epsilon, 1\right)$ e 12 caixas do tipo $\left(\frac{1}{4} - \epsilon, \frac{1}{4} - \epsilon, 1 - (i-1)\xi_N\right)$;
- um nível consistindo de todas as caixas da forma $(\delta, \delta, 1 - (i-1)\xi_N)$. Observe que isto é possível escolhendo δ convenientemente.

Assim, $H(\mathcal{P}^*) \leq 2N + 1$.

Portanto, escolhendo-se ξ_N tal que $h(\xi_N)$ tende a 0 quando $N \to \infty$, temos

$$\lim_{N\to\infty} \frac{H(\mathcal{P})}{\mathrm{OPT}(L)} \geq \lim_{N\to\infty} \frac{5N - h(\xi_N)}{2N + 1} = \frac{5}{2}.$$

□

Complexidade de Tempo

É fácil ver que todos os algoritmos que usamos no Algoritmo TRI_k —exceto para os algoritmos UD e LL— têm complexidade de tempo $\mathcal{O}(m \log m)$, onde m é o número de caixas na lista de entrada correspondente. Pode-se provar que o Algoritmo LL também tem a mesma complexidade de tempo [41]. Quanto ao Algoritmo UD, os autores afirmam (*cf.* [2]) que ele pode ser implementado de forma que sua complexidade de tempo seja $\mathcal{O}(m \log m)$. Assim, o Algoritmo TRI_k tem complexidade de tempo $\mathcal{O}(n \log n)$, onde n é o número de caixas da lista de entrada.

5.3.5 Empacotamento de Caixas de Fundo Quadrado

Nesta seção e nas próximas, aplicaremos a idéia usada no Algoritmo TRI_k, para gerar algoritmos para instâncias particulares do PET. Consideraremos o problema de empacotar uma lista L consistindo de caixas com fundo quadrado para ser empacotado em uma caixa B, não necessariamente de fundo quadrado.

Sem perda de generalidade, consideraremos que o empacotamento será feito em uma caixa B de dimensões $(1, w, \infty)$, $w \geq 1$.

Dada uma lista de caixas $L = (b_1, \ldots, b_n)$, onde $b_i = (x_i, y_i, z_i)$, consideraremos a lista de pontos (no plano xy) dados pelo conjunto $\{(x_1, y_1), \ldots, (x_n, y_n)\}$. Note que como $x_i = y_i$ todos esses pontos se encontram na reta no plano xy que passa através do ponto $(0, 0)$ e do ponto $(1, 1)$. Chamaremos esta reta de *linha de caixas*.

O algoritmo considera dois casos, de acordo com a posição no eixo x onde a linha de caixas cruza com a reta $y = \frac{1}{2}$ (i.e., na posição $xw = \left(\frac{1}{2w}\right) w$).

Algoritmo LS

Entrada: Lista de caixas $L \in \mathcal{Q}^{xy}[0, 1]$.

Saída: Empacotamento $\mathcal{P}$ de L em $B = (1, w, \infty)$.

1 Seja $p := 0{,}4791964$. Subdivida a lista L em $L_1, \ldots, L_7, L_A, L_B, L_C$ como segue (veja a Figura 5.5).

$$
\begin{array}{lll}
L_1 \leftarrow L \bigcap \mathcal{C}\left[\frac{1}{2}, 1\ ;\ \frac{1}{2}, 1\right], & L_2 \leftarrow L \bigcap \mathcal{C}\left[\frac{1}{3}, \frac{1}{2}\ ;\ \frac{1}{2}, 1\right], & L_3 \leftarrow L \bigcap \mathcal{C}\left[0, \frac{1}{3}\ ;\ \frac{1}{2}, 1\right], \\
L_4 \leftarrow L \bigcap \mathcal{C}\left[\frac{1}{3}, \frac{1}{2}\ ;\ \frac{1}{3}, \frac{1}{2}\right], & L_5 \leftarrow L \bigcap \mathcal{C}\left[\frac{1}{4}, \frac{1}{3}\ ;\ \frac{1}{3}, \frac{1}{2}\right], & L_6 \leftarrow L \bigcap \mathcal{C}\left[0, \frac{1}{4}\ ;\ \frac{1}{3}, \frac{1}{2}\right], \\
L_7 \leftarrow L \bigcap \mathcal{C}\left[0, \frac{1}{3}\ ;\ \frac{1}{4}, \frac{1}{3}\right], & L_8 \leftarrow L \bigcap \mathcal{C}\left[0, \frac{1}{4}\ ;\ 0, \frac{1}{4}\right], & L_A \leftarrow L_2 \bigcap \mathcal{C}\left[0, 1\ ;\ 0, \frac{5}{8}\right], \\
L_B \leftarrow L_3 \bigcap \mathcal{C}\left[0, 1\ ;\ 0, \frac{3}{8}\right], & L_C \leftarrow L_1 \bigcap \mathcal{C}\left[0, 1\ ;\ 0, \frac{5}{8}\right], & L_D \leftarrow L_2 \bigcap \mathcal{C}\left[0, p\ ;\ 0, 1\right], \\
L_E \leftarrow L_1 \bigcap \mathcal{C}\left[0, 1 - p\ ;\ 0, 1\right]. &&
\end{array}
$$

2 Seja $x \leftarrow \frac{1}{2w}$.

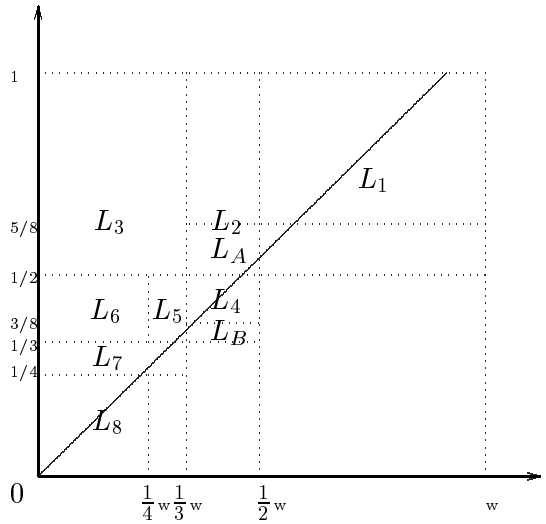


Figura 5.5: Partição da lista L gerada pelo Algoritmo LS.

3 Se $x \leq \frac{2}{5}$

então /* **(Caso 1.)** isto significa que não existem caixas em L_C */

$\mathcal{P}'_{1,2,3} \leftarrow \text{OC}(L_1) \| \text{NFDH}^x(L_2) \| \text{NFDH}^x(L_3)$;
$\mathcal{P}''_{1,2,3} \leftarrow \text{UD}(L_1 \cup L_2 \cup L_3)$;
$\mathcal{P}'' \leftarrow \big(\mathcal{P} \in \{\mathcal{P}'_{1,2,3}, \mathcal{P}''_{1,2,3}\} : \ H(\mathcal{P}) \text{ é mínimo}\big)$;
$\mathcal{P}_{aux} \leftarrow \text{NFDH}^x(L_4) \| \ldots \| \text{NFDH}^x(L_7) \| LL(L_8, 4)$;
$\mathcal{P} \leftarrow \mathcal{P}'' \| \mathcal{P}_{aux}$.
Retorne $\mathcal{P}$.

Senão /* **(Caso 2.)** isto significa que não existem caixas em $L_2 \setminus L_A$ */

4 $(\mathcal{P}_{AB}, L_{AB}) \leftarrow \text{COLUMN}(L_A, [(0,0), (\frac{1}{2}, 0)], L_B, [(0, \frac{5}{8}), (\frac{1}{2}, \frac{5}{8})])$;
5 $L_4 \leftarrow L_4 \setminus L_{AB}$;
$L_B \leftarrow L_B \setminus L_{AB}$.
6 (Caso 2.1.) Se $L_A \subseteq L_{AB}$ /* L_A é totalmente empacotada. */
$(\mathcal{P}_{BC}, L_{BC}) \leftarrow \text{COLUMN}(L_C, [(0,0)], L_B, [(0, \frac{5}{8}), (\frac{1}{2}, \frac{5}{8})])$;
$L_1 \leftarrow L_1 \setminus L_{BC}$;
$L_4 \leftarrow L_4 \setminus L_{BC}$.
(Subcaso 2.1.1.) Se $L_B \subseteq L_{BC}$
$\mathcal{P}' \leftarrow \text{OC}(L_1) \| \mathcal{P}_{BC}$;
$\mathcal{P}_{aux} \leftarrow \mathcal{P}_{AB} \| \text{NFDH}^x(L_4) \| \ldots \| \text{NFDH}^x(L_7) \| \text{LL}(L_8, 4)$.
(Subcaso 2.1.2.) Neste caso, $(L_C \subseteq L_{BC})$

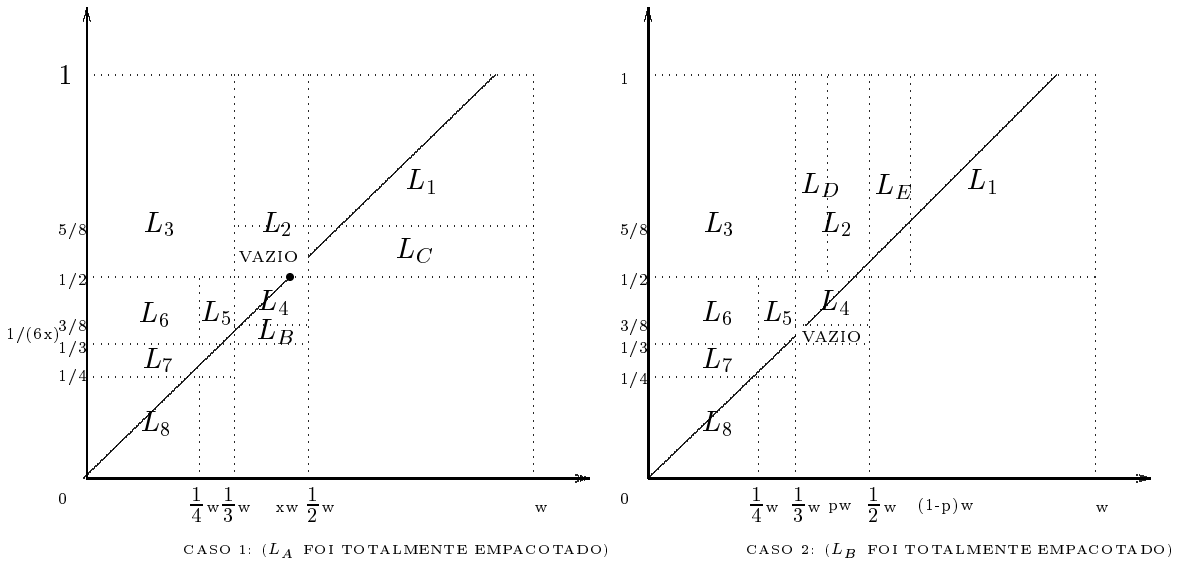


Figura 5.6: Combinando sublistas L_A e L_B.

$\mathcal{P}' \leftarrow \mathrm{OC}(L_1) \| \mathcal{P}_{BC}$;

$\mathcal{P}_{aux} \leftarrow \mathcal{P}_{BC} \| \mathrm{NFDH}^x(L_2) \| \ldots \| \mathrm{NFDH}^x(L_7) \| \mathrm{LL}(L_8, 4)$;

$\mathcal{P} \leftarrow \mathcal{P}' \| \mathcal{P}_{aux}$.

Sejam L' e L_{aux} as listas de caixas empacotadas em $\mathcal{P}'$ e $\mathcal{P}_{aux}$, respectivamente.

7 (Caso 2.2.) Se $L_B \subseteq L_{AB}$ /* L_B é totalmente empacotado. */

/* Defina duas novas sublistas (L_D e L_E) como segue. */

$(\mathcal{P}_{DE}, L_{DE}) \leftarrow \mathrm{COLUMN}(L_D, [(0,0)], L_E, [(p,0)])$;

$L_1 \leftarrow L_1 \setminus L_{DE}$;

$L_2 \leftarrow L_2 \setminus L_{DE}$.

/* Temos dois subcasos, considerando o resultado deste empacotamento. */

(Subcaso 2.2.1.) Se $L_D \subseteq L_{DE}$ ou $x \geq p$, $x = \frac{1}{2w} \in \left(p, \frac{1}{2}\right]$ então

/* Note que quando $x > p$, $L_D = \emptyset$. */

$\mathcal{P}' \leftarrow \mathrm{OC}(L_1) \| \mathcal{P}_{DE}$;

$\mathcal{P}_{aux} \leftarrow \mathcal{P}_{AB} \| \mathrm{NFDH}^x(L_2) \| \ldots \| \mathrm{NFDH}^x(L_7) \| \mathrm{LL}(L_8, 4)$.

Sejam L' e L_{aux} as listas de caixas empacotadas em $\mathcal{P}'$ e $\mathcal{P}_{aux}$, respectivamente.

$\mathcal{P} \leftarrow \mathcal{P}' \| \mathcal{P}_{aux}$.

(Subcaso 2.2.2.) Se $L_E \subseteq L_{DE}$ então

$\mathcal{P}'_{1,2} \leftarrow \mathrm{OC}(L_1) \| \mathrm{NFDH}^x(L_2) \| \mathcal{P}_{DE}$;

$\mathcal{P}''_{1,2} \leftarrow \mathrm{UD}(L_1 \cup L_2 \cup L_{DE})$;

$\mathcal{P}'' \leftarrow \left(\mathcal{P} \in \{\mathcal{P}'_{1,2}, \mathcal{P}''_{1,2}\} : \ H(\mathcal{P}) \text{ é mínimo}\right)$;

$\mathcal{P}_{aux} \leftarrow \mathcal{P}_{AB} \| \mathrm{NFDH}^x(L_4) \| \ldots \| \mathrm{NFDH}^x(L_7) \| \mathrm{LL}(L_8, 4)$.

Seja L'' e L_{aux} as lista de caixas empacotadas em $\mathcal{P}''$ e $\mathcal{P}_{aux}$, respectivamente. $\mathcal{P} \leftarrow \mathcal{P}''\|\mathcal{P}_{aux}$.

8 Retorne $\mathcal{P}$.

Fim algoritmo.

Teorema 5.3.16. *Para qualquer instância do* PET *consistindo de uma lista de caixas com fundo quadrado L, temos que*

$$\mathrm{LS}(L) \leq 2{,}5425 \cdot \mathrm{OPT}(L) + \frac{101}{8}Z.$$

Prova. Como a técnica de prova é análoga às apresentadas nas provas anteriores, vamos resumir a prova. Sugerimos ao leitor que siga as análises de cada caso, juntamente com o correspondente caso na descrição do algoritmo.

Caso 1.

Como temos uma garantia de área de pelo menos $\frac{5}{16}l \cdot w$ para os empacotamentos das listas L_1, L_2 e L_3, temos

$$H(\mathcal{P}'') \leq \frac{16}{5}\frac{V(L'')}{l \cdot w} + \frac{53}{8}Z. \tag{5.38}$$

Pelo Lema 5.3.12,

$$H(\mathcal{P}'') \leq \frac{5}{4}\mathrm{OPT}(L) + \frac{53}{8}. \tag{5.39}$$

Para os demais empacotamentos temos uma garantia de área de pelo menos $\frac{l \cdot w}{2}$, portanto

$$H(\mathcal{P}_{aux}) \leq 2\frac{V(L_{aux})}{l \cdot w} + 5Z. \tag{5.40}$$

Definindo $\mathcal{H}_1 := H(\mathcal{P}'') - \frac{53}{8}Z$ e $\mathcal{H}_2 := H(\mathcal{P}_{aux}) - 4Z$, temos que

$$\mathrm{OPT}(L) \geq \max\{\frac{4}{5}\mathcal{H}_1, \frac{5}{16}\mathcal{H}_1 + \frac{1}{2}\mathcal{H}_2\}$$

e portanto, procedendo como anteriormente, obtemos

$$H(\mathcal{P}) \leq \alpha_1 \cdot \mathrm{OPT}(L) + \frac{53}{8}Z,$$

onde $\alpha_1 \leq \frac{\mathcal{H}_1+\mathcal{H}_2}{\max\{\frac{4}{5}\mathcal{H}_1, \frac{5}{16}\mathcal{H}_1+\frac{1}{2}\mathcal{H}_2\}} \leq 2{,}5$.

Note que para os casos restantes, as listas L_3 e $L_2 \setminus L_A$ são vazias e a linha de caixas cruza a região L_C.

Subcaso 2.1.1. Note que neste caso, $L_A \cup L_B$ foi totalmente empacotado em $\mathcal{P}_{AB}\|\mathcal{P}_{BC}$. Note também que as caixas de L_C em $\mathcal{P}_{BC}$ pertencem a $\mathcal{C}^{xy}\left[\frac{1}{2}, 1\ ;\ \frac{1}{2}, 1\right]$ e portanto, obtemos a seguinte inequação:

$$H(\mathcal{P}') \leq 4\frac{V(L')}{l \cdot w} + Z.$$

Como

$$\begin{aligned} H(\mathcal{P}_{aux}) &\leq 2\frac{V(L_{aux})}{l \cdot w} + 7Z, \\ H(\mathcal{P}') &\leq \mathrm{OPT}(L) + Z, \end{aligned}$$

usando as inequações acima e definindo $\mathcal{H}_1 := H(\mathcal{P}') - Z$ e $\mathcal{H}_2 := H(\mathcal{P}_{aux}) - 7Z$ temos que

$$H(\mathcal{P}) \leq \alpha_{2,1,1} \cdot \mathrm{OPT}(L) + 8Z,$$

onde $\alpha_{2,1,1} \leq \frac{\mathcal{H}_1+\mathcal{H}_2}{\max\{\mathcal{H}_1, \frac{1}{4}\mathcal{H}_1+\frac{1}{2}\mathcal{H}_2\}} \leq 2{,}5$.

Subcaso 2.1.2. Neste caso, temos que todas as caixas de $L_A \cup L_C$ foram totalmente empacotadas em $\mathcal{P}_{AB} \| \mathcal{P}_{BC}$. Temos também que $x = \frac{1}{2w}$ e $x \in (\frac{2}{5}, \frac{1}{2}]$ (note que $x \cdot w$ é a posição no eixo x onde a linha de caixas cruza a reta $y = \frac{1}{2}$). Neste caso, temos as seguintes inequações com respeito a x.

$$\begin{aligned} H(\mathcal{P}') &\leq \frac{32}{25x}\frac{V(L')}{l \cdot w} + Z, \\ H(\mathcal{P}_{aux}) &\leq \frac{1}{\min\{\frac{2}{9x}, \frac{1}{2}\}}\frac{V(L_{aux})}{l \cdot w} + 8Z, \\ H(\mathcal{P}') &\leq \mathrm{OPT}(L) + Z, \end{aligned}$$

e portanto,

$$H(\mathcal{P}) \leq \alpha_{2,1,2} \cdot \mathrm{OPT}(L) + 9Z,$$

onde $\alpha_{2,1,2} \leq \frac{\mathcal{H}_1+\mathcal{H}_2}{\max\{\mathcal{H}_1, \frac{25x}{32}\mathcal{H}_1+\min\{\frac{2}{9x}, \frac{1}{2}\}\mathcal{H}_2\}}$. Analisando o valor de $\alpha_{2,1,2}$, quando $x \geq \frac{4}{9}$ e quando $x < \frac{4}{9}$, podemos obter que $\alpha_{2,1,2} \leq 2{,}5$.

Subcaso 2.2.1. Neste caso, temos que $L_B \cup L_D$ é totalmente empacotado em $\mathcal{P}_{AB} \| \mathcal{P}_{DE}$. Lembramos que $x = \frac{1}{2w}$. Dividimos a análise em dois casos, considerando primeiro quando $x \in (p, \frac{1}{2}]$. Temos aqui que

$$\begin{aligned} H(\mathcal{P}') &\leq 8x\frac{V(L')}{l \cdot w} + Z, \\ H(\mathcal{P}_{aux}) &\leq \frac{1}{x}\frac{V(L_{aux})}{l \cdot w} + 7Z, \\ H(\mathcal{P}') &\leq \mathrm{OPT}(L) + Z, \end{aligned}$$

e portanto,

$$H(\mathcal{P}) \leq \alpha_{2,2,1} \cdot \mathrm{OPT}(L) + 8Z,$$

onde $\alpha_{2,2,1} \leq \frac{\mathcal{H}_1+\mathcal{H}_2}{\max\{\mathcal{H}_1, \frac{1}{8x}\mathcal{H}_1+x\mathcal{H}_2\}} \leq \frac{\mathcal{H}_1+\mathcal{H}_2}{\max\{\mathcal{H}_1, \frac{1}{8p}\mathcal{H}_1+p\mathcal{H}_2\}} \leq 2{,}5424\ldots$

Se $x \in (\frac{2}{5}, p]$, a análise é similar e será omitida.

Subcaso 2.2.2. Temos aqui que $L_B \cup L_E$ é totalmente empacotado em $\mathcal{P}_{AB} \| \mathcal{P}_{DE}$. Seja $x = \frac{1}{2w}$, $x \in (\frac{2}{5}, p]$. Então,

$$\begin{aligned} H(\mathcal{P}'') &\leq \frac{2x}{(1-p)^2}\frac{V(L')}{l \cdot w} + \frac{53}{8}Z, \\ H(\mathcal{P}_{aux}) &\leq 2\frac{V(L_{aux})}{l \cdot w} + 5Z, \\ H(\mathcal{P}'') &\leq \frac{5}{4}\mathrm{OPT}(L) + \frac{53}{8}Z, \end{aligned}$$

e portanto,

$$H(\mathcal{P}) \leq \alpha_{2,2,2} \cdot \mathrm{OPT}(L) + \frac{93}{8}Z,$$

onde $\alpha_{2,2,2} \leq \frac{\mathcal{H}_1+\mathcal{H}_2}{\max\{\frac{4}{5}\mathcal{H}_1, \frac{(1-p)^2}{2x}\mathcal{H}_1+\frac{1}{2}\mathcal{H}_2\}} \leq \frac{\mathcal{H}_1+\mathcal{H}_2}{\max\{\frac{4}{5}\mathcal{H}_1, \frac{(1-p)^2}{2p}\mathcal{H}_1+\frac{1}{2}\mathcal{H}_2\}} \leq 2{,}5424\ldots.$

De fato, o valor p foi tomado de tal maneira que os dois subcasos acima (2.2.1 e 2.2.2) conduzissem ao mesmo limite.

O teorema segue considerando todos os casos analisados acima. □

O próximo lema será usado para provar limites inferiores para os limites de desempenho assintótico de alguns algoritmos apresentados aqui.

Lema 5.3.17. *Seja $\mathcal{A}$ um algoritmo para o* PET *(*PET^r*) que particiona a lista de entrada L em duas sublistas $L_1 \subset \wp_4$ e $L_2 \in \mathcal{Q}_m$, $m \geq 3$, e gera um empacotamento $\mathcal{P} = \mathcal{P}_1 || \mathcal{P}_2$, onde $\mathcal{P}_1$ é qualquer empacotamento de L_1 e $\mathcal{P}_2$ é um empacotamento de L_2 usando o Algoritmo LL. Então o limite de desempenho assintótico $r(\mathcal{A})$ de $\mathcal{A}$ é tal que $r(\mathcal{A}) \geq \frac{7m-8}{4m-8}$.*

Prova. A prova é análoga à do Lema 4.4.11, usando o valor $\left(\frac{m-2}{m}\right)$ ao invés de $\left(\frac{m-1}{m}\right)^2$ □

Proposição 5.3.18. *O limite de desempenho assintótico do Algoritmo LS está entre* 2,5 *e* 2,5425.

Prova. Segue diretamente do Teorema 5.3.16 e do Lema 5.3.17 (tomando $m = 4$). □

5.3.6 Empacotamento de Caixas com Fundo Quadrado em Caixa com Fundo Quadrado

Considere agora o caso especial do PET onde todas as caixas de L e B têm fundo quadrado. Em 1990, Li e Cheng [40] apresentaram um algoritmo com limite de desempenho assintótico 2,6875 para este problema. Nesta seção, apresentaremos um algoritmo para o PET chamado $\mathrm{SS}_m^{(t)}$, que melhora o resultado de Li e Cheng e tem um limite de desempenho assintótico 2,36048.... Sem perda de generalidade, consideraremos o PET$(1, 1)$.

O algoritmo para empacotar caixas de fundo quadrado em caixa B de fundo quadrado vem da adaptação do Algoritmo SS_m do PEP para o PET. Chamaremos este algoritmo de $\mathrm{SS}_m^{(t)}$. Não apresentaremos o algoritmo explicitamente, já que este pode ser obtido fazendo-se as mesmas adaptações feitas para se construir o Algoritmo $\mathrm{STP}_m^{(t)}$ a partir do Algoritmo STP_m. O limite obtido para o Algoritmo $\mathrm{SS}_m^{(t)}$ também melhora o limite $\left(\frac{m+1}{m}\right)$ de Li e Cheng [40]

Teorema 5.3.19. *Para qualquer lista L para o* PET*, onde todas as caixas têm fundo quadrado,*

$$\mathrm{SS}_1^{(t)}(L) \leq 2{,}36048602\ldots \cdot \mathrm{OPT}(L) + 4Z.$$

Os mesmos limites para valores de m entre 1 e 10 apresentados na Tabela 4.2 valem para o Algoritmo $\mathrm{SS}_m^{(t)}$.

5.4 Caso com Rotação em Torno do Eixo z

Observamos que neste problema não podemos reparametrizar a caixa $B = (a, b, \infty)$ para $B = (1, 1, \infty)$. Uma reparametrização deste tipo poderia fazer com que caixas que não poderiam ser giradas, possam ser em caixas $B = (1, 1, \infty)$.

Em 1990, Li e Cheng [41] apresentaram o PET^r como um modelo para um *problema de escalonamento de tarefas em sistemas de malha conexa particionável*. Neste problema, um conjunto de tarefas $J_1, J_2, \ldots, J_n$ deve ser processado em um sistema de malha conexa particionável que consiste de $l \times w$ elementos de processamento ligados como uma malha retangular. Cada tarefa J_i é especificada por uma tripla $J_i = (x_i, y_i, t_i)$ indicando que uma submalha de tamanho (x_i, y_i) ou (y_i, x_i) é requerida pela tarefa J_i, e t_i é seu tempo de processamento (veja Figura 5.7). O objetivo é atribuir as tarefas para as submalhas de forma a minimizar o tempo total de processamento.

Apresentamos um algoritmo *off-line* com um limite de desempenho assintótico, para o caso geral, que pode se tornar tão próximo de 2,67 quanto se queira. Para o caso *on-line* apresentamos um algoritmo com um limite de desempenho assintótico que pode se tornar tão próximo de 3,25 quanto se queira.

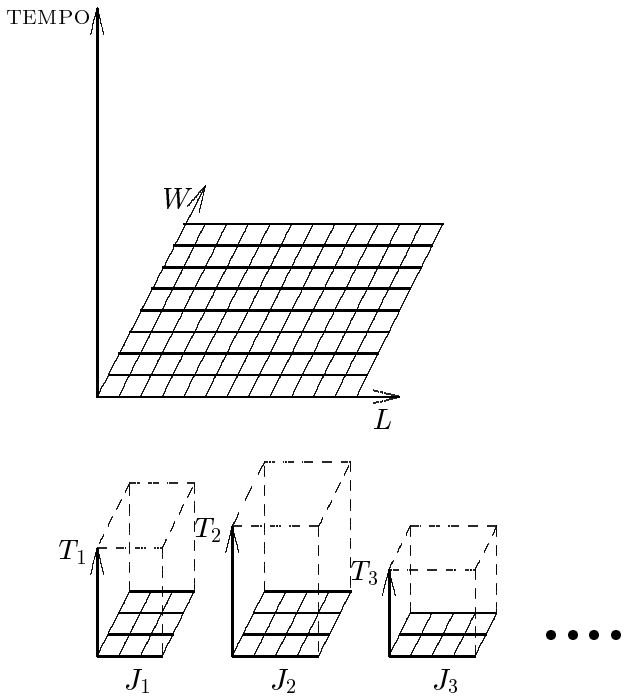


Figura 5.7: Escalonamento de processos $J_1, J_2, J_3, \ldots$ em computador paralelo.

5.4.1 Algoritmo R_k

Nesta seção, adaptamos o Algoritmo TRI_k de forma que este se torne um algoritmo para o PET^r e mantenha o mesmo limite de desempenho assintótico de 2,67 do Algoritmo TRI_k.

Primeiramente, vamos apresentar o Algoritmo COLUMN^r que será usado como subrotina. Este algoritmo é uma versão modificada do Algoritmo COLUMN apresentado anteriormente e gera um empacotamento parcial de uma lista L.

O empacotamento gerado pelo Algoritmo COLUMN^r também consistirá de várias pilhas de caixas, referidas como *colunas*. Cada coluna é construída colocando-se uma caixa no topo de outra, e cada coluna consiste apenas de caixas do tipo $\mathcal{T}^1$ ou $\mathcal{T}^2$.

O Algoritmo COLUMN^r é também chamado com os parâmetros $(L, \mathcal{T}^1, \mathcal{T}^2, [p^1], [p^2])$, e difere do Algoritmo COLUMN no momento de empacotar uma nova caixa. Em cada iteração, o algoritmo escolhe uma coluna com a menor altura, digamos uma coluna dada pela posição p^i_j, e empacota uma caixa não empacotada $b \in L$ do tipo $\mathcal{T}^i$ girando-a previamente de forma a ter $b \in \mathcal{T}^i$; atualizando a lista L depois de cada iteração. Se não existir caixa b deste tipo, então o algoritmo termina retornando o empacotamento parcial $\mathcal{P}$ de L.

O seguinte lema sobre este algoritmo é válido. A prova é análoga àquela feita para o Algoritmo COLUMN.

Lema 5.4.1. *Seja $\mathcal{P}$ o empacotamento de $L' \subseteq L$ gerado pelo Algoritmo* COLUMN *quando aplicado às listas de tipos $\mathcal{T}^1$ e $\mathcal{T}^2$ e lista de posições $p_1^i, p_2^i, \ldots, p_{n_i}^i$, $i = 1, 2$. Se $S(b) \geq s_i \cdot l \cdot w$, para todas as caixas b em $\mathcal{T}^i$, $i = 1, 2$, então $H(\mathcal{P}) \leq \frac{1}{s_1 n_1 + s_2 n_2} \frac{V(L')}{l \cdot w} + Z$.*

Os seguintes resultados referentes aos algoritmos OC, UD^x e UD^y são adaptações do PET para o PET^r.

Lema 5.4.2. *Se $\mathcal{P}$ é o empacotamento gerado pelo Algoritmo* OC *quando aplicado à lista L e s é uma constante, tal que $S(b) \geq s \cdot l \cdot w$ para todas as caixas b em L, então $H(\mathcal{P}) \leq \frac{V(L)}{s \cdot l \cdot w}$.*

Lema 5.4.3. *Se $\mathcal{P}$ é o empacotamento gerado pelo Algoritmo* OC *quando aplicado à lista L tal que $b \in P_4$ e ($\rho(b) \in P_4$ ou $\rho(b) \notin \mathcal{C}^{xy}[0, 1\ ;\ 0, 1]$) então $H(\mathcal{P}) = \mathrm{OPT}(L)$.*

Lema 5.4.4. *Seja L uma instância para o PET^r tal que $b \in \mathcal{C}\left[\frac{1}{2}, 1\ ;\ 0, 1\right]$ (resp. $b \in \mathcal{C}^{xy}\left[0, 1\ ;\ \frac{1}{2}, 1\right]$) e ou $x(b) \leq y(b)$ (resp. $y(b) \leq x(b)$) ou $\rho(b) \notin \mathcal{C}^{xy}[0, 1\ ;\ 0, 1]$ (resp. $\rho(b) \notin \mathcal{C}[0, 1\ ;\ 0, 1]$) para todas as caixas b em L. Isto significa que duas caixas não podem ser empacotadas lado a lado na direção do eixo x. (resp. direção do eixo y). Então o empacotamento $\mathcal{P}$ gerado pelo Algoritmo UD^x (resp. UD^y) é tal que $H(\mathcal{P}) \leq \frac{5}{4}\mathrm{OPT}(L) + \frac{53}{8}Z$.*

Prova. Este resultado segue diretamente do Lema 5.3.12 e do fato que duas caixas não podem ser empacotadas lado a lado na direção do eixo x. (resp. direção do eixo y), mesmo se rotações são permitidas. □

Agora podemos apresentar a descrição do Algoritmo R_k. Basicamente, os passos do algoritmo são análogos àqueles do Algoritmo TRI_k apresentado anteriormente para o PET. A principal diferença reside no tratamento de algumas caixas, já que agora estas podem sofrer rotações e temos de tratá-las diferentemente para obter inequações válidas com respeito ao empacotamento ótimo.

Algoritmo R_k

Entrada: Lista de caixas L.

Saída: Empacotamento $\mathcal{P}$ de L em $B = (l, w, \infty)$.

1 Gire todas as caixas b que estão em $\wp_4$ de tal forma que $\rho(b) \in \wp_1 \cup \wp_2 \cup \wp_3$.
/* I.e., Seja $R_1 \leftarrow \{b \in L \cap \wp_4 : \rho(b) \in \wp_1 \cup \wp_2 \cup \wp_3\}$. $L \leftarrow (L \setminus R_1) \bigcup \rho(R_1)$. */

2 Gire todas as caixas b de L que estão $\wp_2 \cup \wp_3$ tal que $\rho(b) \in \wp_1$.

3 $\mathcal{P}_{AB} \leftarrow \mathrm{COMBINE\text{-}AB}_k^{xy}(L, xy\text{-type}, \mathcal{C}_1, \mathcal{C}_1, \mathrm{COLUMN}^r))$.

Atualize(L).

4 Se $xy\text{-type}(L, \mathcal{A}_k^{xy}) = \emptyset$ então

4.1 Gire as caixas de $L \cap \wp_2$ que se encaixam em $\wp_3$.

4.2 Gire as caixas de $L \cap (\wp_2 \cup \wp_4)$ tal que se $b \in L \cap (\wp_2 \cup \wp_4)$ então $x(b) \leq y(b)$ ou $\rho(b) \notin \mathcal{C}_1$.

4.3 Subdivida a lista L em $L_1, \ldots, L_{25}$ como segue (veja a Figura 4.12).

$$L_i \leftarrow L \bigcap \mathcal{C}\left[\tfrac{1}{2}, 1\ ;\ \tfrac{1}{i+2}, \tfrac{1}{i+1}\right],\ \text{para}\ \ i = 1, \ldots, 16 \qquad L_{17} \leftarrow L \bigcap \mathcal{C}\left[\tfrac{1}{2}, 1\ ;\ 0, \tfrac{1}{18}\right],$$
$$L_{18} \leftarrow L \bigcap \mathcal{C}\left[\tfrac{1}{3}, \tfrac{1}{2}\ ;\ \tfrac{1}{3}, \tfrac{1}{2}\right], \qquad L_{19} \leftarrow L \bigcap \mathcal{C}\left[\tfrac{1}{3}, \tfrac{1}{2}\ ;\ \tfrac{1}{4}, \tfrac{1}{3}\right],$$
$$L_{20} \leftarrow L \bigcap \mathcal{C}\left[\tfrac{1}{3}, \tfrac{1}{2}\ ;\ 0, \tfrac{1}{4}\right], \qquad L_{21} \leftarrow L \bigcap \mathcal{C}\left[\tfrac{1}{4}, \tfrac{1}{3}\ ;\ \tfrac{1}{3}, \tfrac{1}{2}\right],$$
$$L_{22} \leftarrow L \bigcap \mathcal{C}\left[\tfrac{1}{4}, \tfrac{1}{3}\ ;\ 0, \tfrac{1}{3}\right], \qquad L_{23} \leftarrow L \bigcap \mathcal{C}\left[0, \tfrac{1}{4}\ ;\ \tfrac{1}{3}, \tfrac{1}{2}\right],$$
$$L_{24} \leftarrow L \bigcap \mathcal{C}\left[0, \tfrac{1}{4}\ ;\ \tfrac{1}{4}, \tfrac{1}{3}\right], \qquad L_{25} \leftarrow L \bigcap \mathcal{C}_4,$$
$$L_C \leftarrow L \bigcap \mathcal{C}\left[\tfrac{1}{2}, 1\ ;\ \tfrac{1}{2}, \tfrac{1}{t}\right] \qquad L'_D \leftarrow L_1 \bigcap \mathcal{C}\left[0, t\ ;\ 0, 1\right],$$
$$L''_D \leftarrow L_{18} \bigcap \mathcal{C}\left[0, t\ ;\ 0, 1\right] \qquad L_D \leftarrow L'_D \bigcup L''_D.$$

onde $t = 0{,}46588$.

4.4 Gere empacotamento $\mathcal{P}_{CD}$ como segue.

$(\mathcal{P}_{CD'}, L_{CD'}) \leftarrow \text{COLUMN}(L_C, [(0,0)], L'_D, [(0, 1-t)])$;
$(\mathcal{P}_{CD''}, L_{CD''}) \leftarrow \text{COLUMN}(L_C \setminus L_{CD'}, [(0,0)], L''_D, [(0, 1-t), (\tfrac{1}{2}, 1-t)])$;
$\mathcal{P}_{CD} \leftarrow \mathcal{P}_{CD'} \| \mathcal{P}_{CD''}$.
$L_{CD} \leftarrow L_{CD'} \bigcup L_{CD''}$;
$L_1 \leftarrow L_1 \setminus L_{CD}$;
$L_{18} \leftarrow L_{18} \setminus L_{CD}$.

4.5 Gere empacotamentos $\mathcal{P}_1, \ldots, \mathcal{P}_{25}$ como segue.

$\mathcal{P}_i \leftarrow \text{NFDH}^y(L_i) \quad \text{para} \quad i = 1, \ldots, 22$;
$\mathcal{P}_i \leftarrow \text{NFDH}^x(L_i) \quad \text{para} \quad i = 23, 24$;
$\mathcal{P}_{25} \leftarrow \text{LL}(L_{25}, 4)$.

4.6 Atualize L removendo as caixas empacotadas. Note que $L \subseteq \wp_2 \cup \wp_4$.

4.7 Se $L_C \subseteq L_{CD}$

então (Caso 1) $p \leftarrow 0{,}441328$; /* L_C é totalmente empacotado */ (veja Figura 5.3)
senão (Caso 2) $p \leftarrow 0{,}451600$; /* L_D é totalmente empacotado */ (veja Figura 5.4)

4.8 $L_E \leftarrow L \cap \mathcal{C}\left[\tfrac{1}{2}, 1-p\ ;\ \tfrac{1}{2}, 1\right]$;
$L'_F \leftarrow L \cap \mathcal{C}\left[\tfrac{1}{9}, p\ ;\ \tfrac{1}{2}, 1\right]$;
$L''_F \leftarrow L \cap \mathcal{C}\left[\tfrac{1}{18}, \tfrac{1}{9}\ ;\ \tfrac{1}{2}, 1\right]$;
$L_F \leftarrow L'_F \cup L''_F$;

4.9 $(\mathcal{P}_{EF'}, L_{EF'}) \leftarrow \text{COLUMN}(L_E, [(0,0)], L'_F, [(1-p, 0)])$;
$(\mathcal{P}_{EF''}, L_{EF''}) \leftarrow \text{COLUMN}(L_E \setminus L_{EF'}, [(0,0)], L''_F, [(0, 1-p), (0, 1-p+\tfrac{1}{9}),\ ;$
$\ldots, (0, 1-p+(\lfloor 9p \rfloor - 1)\tfrac{1}{9})])$;
$\mathcal{P}_{EF} \leftarrow \mathcal{P}_{EF'} \| \mathcal{P}_{EF''}$;

$L_{EF} \leftarrow L_{EF'} \cup L_{EF''}$.

4.10 Se $L_E \subseteq L_{EF}$ /* (Subcaso 1) L_E é totalmente empacotado */

então

$\mathcal{P}_{\mathrm{UD}} \leftarrow \mathrm{UD}^x(L)$;
$\mathcal{P}_{OC} \leftarrow \mathrm{OC}((L \setminus L_{EF}) \cap \wp_4)$;
$\mathcal{P}_{2e} \leftarrow \mathrm{NFDH}^x((L \setminus L_{EF}) \cap \mathcal{C}\left[0, \frac{1}{3}\ ;\ 0, 1\right])$;
$\mathcal{P}_{2d} \leftarrow \mathrm{NFDH}^x((L \setminus L_{EF}) \cap \mathcal{C}\left[p, \frac{1}{2}\ ;\ 0, 1\right])$;
$\mathcal{P}' \leftarrow \mathcal{P}_{OC} \| \mathcal{P}_{2e} \| \mathcal{P}_{2d} \| \mathcal{P}_{EF}$;
$\mathcal{P}'' \leftarrow \{\mathcal{P} \in \{\mathcal{P}_{\mathrm{UD}}, \mathcal{P}'\} :\ H(\mathcal{P})$ é mínimo $\}$;
$\mathcal{P}_{aux} \leftarrow \mathcal{P}_{AB} \| \mathcal{P}_{CD} \| \mathcal{P}_1 \| \ldots \| \mathcal{P}_{25}$.
Seja L'' e L_{aux} listas de caixas empacotadas em $\mathcal{P}''$ e $\mathcal{P}_{aux}$, resp.
$\mathcal{P} \leftarrow \mathcal{P}_{aux} \| \mathcal{P}''$.

4.11 Se $L_F \subseteq L_{EF}$ /* (Subcaso 2) L_F é totalmente empacotada */

então

$\mathcal{P}_{OC} \leftarrow \mathrm{OC}((L \setminus L_{EF}) \cap \wp_4)$;
$\mathcal{P}_{2e} \leftarrow \mathrm{NFDH}^x((L \setminus L_{EF}) \cap \mathcal{C}\left[0, \frac{1}{18}\ ;\ \frac{1}{2}, 1\right])$;
$\mathcal{P}_{2d} \leftarrow \mathrm{NFDH}^x((L \setminus L_{EF}) \cap \mathcal{C}\left[p, 1\ ;\ \frac{1}{2}, 1\right])$;
$\mathcal{P}' \leftarrow \mathcal{P}_{OC} \| \mathcal{P}_{EF}$;
$\mathcal{P}_{aux} \leftarrow \mathcal{P}_{AB} \| \mathcal{P}_{CD} \| \mathcal{P}_{2e} \| \mathcal{P}_{2d} \| \mathcal{P}_1 \| \ldots \| \mathcal{P}_{25}$;
Seja L' e L_{aux} as listas de caixas empacotadas em $\mathcal{P}'$ e $\mathcal{P}_{aux}$, resp.
$\mathcal{P} \leftarrow \mathcal{P}_{aux} \| \mathcal{P}'$.

5 Senão, gere um empacotamento $\mathcal{P}$ de L como no passo 3 (de forma simétrica).

6 Retorne $\mathcal{P}$.

Fim algoritmo.

O próximo teorema nos dá um limite de desempenho assintótico para o Algoritmo R_k quando $k \to \infty$.

Teorema 5.4.5. *Para qualquer instância L do* PET^r*, temos que*

$$\mathrm{R}_k(L) \leq \alpha_k \cdot \mathrm{OPT}(L) + \left(2k + \frac{597}{8}\right) Z,$$

onde $\alpha_k \to 2{,}6607\ldots$ à medida que $k \to \infty$.

Prova. Apresentamos a prova para o caso onde todas as caixas do tipo (xy-type) $\mathcal{A}_k^{xy}$ foram empacotadas (passo 4). A prova para o outro caso (passo 5) é análoga. Esta prova é subdividida em 4 casos, de acordo com os passos 4.7 ($L_C \subseteq L_{CD}$), 4.10 ($L_E \subseteq L_{EF}$) e 4.11 ($L_F \subseteq L_{EF}$).

Como muitos passos do Algoritmo R_k são similares aos do Algoritmo TRI_k do PET, muitas das inequações obtidas nas análises de TRI_k são válidas para este algoritmo. Nós apenas as mencionaremos nas afirmações que fazemos a seguir.

Caso 1.1. $(L_C \subseteq L_{CD})$ e $(L_E \subseteq L_{EF})$.

Afirmação 1.1.

$$H(\mathcal{P}'') \leq \frac{1}{(1-p)(1-t)} \frac{V(L'')}{l \cdot w} + 4Z \quad \text{e} \quad H(\mathcal{P}_{aux}) \leq \frac{1}{r_1} \frac{V(L_{aux})}{l \cdot w} + (2k+68)Z.$$

Sejam $\mathcal{H}_1 := H(\mathcal{P}'') - \frac{53}{8}Z$ e $\mathcal{H}_2 := H(\mathcal{P}_{aux}) - (2k+68)Z$.

Usando a definição de $\mathcal{H}_1$ e $\mathcal{H}_2$ nas duas inequações acima, obtemos que

$$\mathrm{OPT}(L) \geq \frac{V(L)}{l \cdot w} = \frac{V(L'')}{l \cdot w} + \frac{V(L_{aux})}{l \cdot w} \geq (1-p)(1-t)\mathcal{H}_1 + r_1\mathcal{H}_2,$$

i.e.,

$$\mathrm{OPT}(L) \geq (1-p)(1-t)\mathcal{H}_1 + r_1\mathcal{H}_2. \tag{5.41}$$

Note que dos passos 1, 2, 4, 4.1 e 4.2 concluímos que a lista L'' satisfaz a condição do Lema 5.4.4. Portanto, temos

$$H(\mathcal{P}'') \leq \mathrm{UD}^x(L'') \leq \frac{5}{4}\mathrm{OPT}(L'') + \frac{53}{8}Z \leq \frac{5}{4}\mathrm{OPT}(L) + \frac{53}{8}Z,$$

i.e.,

$$\mathrm{OPT}(L) \geq \frac{4}{5}\mathcal{H}_1. \tag{5.42}$$

Conseqüentemente, das inequações (5.41) e (5.42) temos que

$$\mathrm{OPT}(L) \geq \max\{\frac{4}{5}\mathcal{H}_1, (1-p)(1-t)\mathcal{H}_1 + r_1\mathcal{H}_2\}.$$

Das definições de $\mathcal{H}_1$ e $\mathcal{H}_2$, temos que

$$H(\mathcal{P}) = H(\mathcal{P}'') + H(\mathcal{P}_{aux}) = \mathcal{H}_1 + \mathcal{H}_2 + \left(2k + \frac{597}{8}\right) Z.$$

Usando a última inequação, temos que

$$H(\mathcal{P}) \leq \left(\frac{\mathcal{H}_1 + \mathcal{H}_2}{\max\{\frac{4}{5}\mathcal{H}_1, (1-p)(1-t)\mathcal{H}_1 + r_1\mathcal{H}_2\}}\right) \mathrm{OPT}(L) + \left(2k + \frac{597}{8}\right) Z.$$

Mais ainda, para o Algoritmo TRI_k provamos que

$$\alpha'_k(r_1) = \frac{4 - 5(1-p)(1-t) + r_1}{4r_1} \geq \frac{\mathcal{H}_1 + \mathcal{H}_2}{\max\{\frac{4}{5}\mathcal{H}_1, (1-p)(1-t)\mathcal{H}_1 + r_1\mathcal{H}_2\}}$$

e portanto,

$$H(\mathcal{P}) \leq \alpha'_k(r_1) \cdot \mathrm{OPT}(L) + \left(2k + \frac{597}{8}\right) Z.$$

Como $r_1 \to \frac{4}{9}$ à medida que $k \to \infty$, podemos concluir que $\alpha'_k(r_1) \to 2{,}66076\ldots$ à medida que $k \to \infty$.

Caso 1.2. $(L_C \subseteq L_{CD})$ e $(L_F \subseteq L_{EF})$.

Afirmação 1.2.

$$H(\mathcal{P}') \leq \frac{1}{(1-t)/2}\frac{V(L')}{l \cdot w} + 2Z, \quad \text{e} \quad H(\mathcal{P}_{aux}) \leq \frac{1}{p}\frac{V(L_{aux})}{l \cdot w} + (2k+70)Z.$$

Sejam $\mathcal{H}_1 := H(\mathcal{P}') - 2Z$ e $\mathcal{H}_2 := H(\mathcal{P}_{aux}) - (2k+70)Z$.

Então, temos

$$\mathrm{OPT}(L) \geq \frac{V(L')}{l \cdot w} + \frac{V(L_{aux})}{l \cdot w} \geq \frac{1-t}{2}\mathcal{H}_1 + p\mathcal{H}_2. \tag{5.43}$$

Note que cada caixa em $L' \cap \wp_4$ considerada no passo 4.11, não pode ser girada, ou se puder, esta se encontra em $\wp_4$ novamente. Portanto, podemos concluir que

$$\mathrm{OPT}(L) \geq \mathrm{OPT}(L') \geq \mathcal{H}_1. \tag{5.44}$$

Procedendo como no Caso 1.1., usando as inequações (5.43) e (5.44) temos que

$$H(\mathcal{P}) \leq \alpha''_k \cdot \mathrm{OPT}(L) + (2k+72)Z,$$

onde $\alpha''_k = \frac{1+t+2p}{2p}$.

Assim, das análises dos subcasos 1.1 e 1.2 podemos concluir que

$$\mathrm{R}_k(L) \leq \alpha_k \cdot \mathrm{OPT}(L) + \left(2k + \frac{597}{8}\right) Z,$$

onde $\alpha_k \to \alpha'_k(\frac{4}{9}) = \alpha''_k = 2{,}66076\ldots$ à medida que $k \to \infty$.

Caso 2.1. $(L_D \subseteq L_{CD})$ e $(L_E \subseteq L_{EF})$.

Afirmação 2.1.

$$H(\mathcal{P}'') \leq \frac{1}{(1-p)\frac{1}{2}}\frac{V(L'')}{l \cdot w} + \frac{53}{8}Z \quad \text{e} \quad H(\mathcal{P}_{aux}) \leq \frac{1}{t}\frac{V(L_{aux})}{l \cdot w} + (2k+68)Z.$$

Sejam $\mathcal{H}_1 := H(\mathcal{P}'') - \frac{53}{8}Z$ e $\mathcal{H}_2 := H(\mathcal{P}_{aux}) - (2k+68)Z$.

Então, temos $\mathrm{OPT}(L) \geq \frac{V(L'')}{l \cdot w} + \frac{V(L_{aux})}{l \cdot w} \geq \frac{1-p}{2}\mathcal{H}_1 + t \cdot \mathcal{H}_2$.

Usando a mesma idéia do caso 1.1., temos que $\mathrm{OPT}(L) \geq \mathrm{OPT}(L'') \geq \frac{4}{5}\mathcal{H}_1$, e portanto, temos $H(\mathcal{P}) \leq \beta'_k(r_1)\mathrm{OPT}(L) + \left(2k + \frac{597}{8}\right) Z$, onde $\beta'_k(r_1) = \frac{1}{t} - \frac{5(1-p)}{8t} + \frac{5}{4}$.

Caso 2.2. $(L_D \subseteq L_{CD})$ e $(L_F \subseteq L_{EF})$.

Afirmação 2.2.

$$H(\mathcal{P}') \leq 4\frac{V(L')}{l \cdot w} + 2Z, \quad \text{e} \quad H(\mathcal{P}_{aux}) \leq \frac{1}{p}\frac{V(L_{aux})}{l \cdot w} + (2k+70)Z.$$

Seja $\mathcal{H}_1 := H(\mathcal{P}') - 2Z$ e $\mathcal{H}_2 := H(\mathcal{P}_{aux}) - (2k+70)Z$.

Neste caso podemos obter $\mathrm{OPT}(L) \geq \frac{V(L')}{l \cdot w} + \frac{V(L_{aux})}{l \cdot w} \geq \frac{1}{4}\mathcal{H}_1 + p\mathcal{H}_2$ e $\mathrm{OPT}(L) \geq \mathcal{H}_1$. Portanto, $H(\mathcal{P}) \leq \beta''_k \cdot \mathrm{OPT}(L) + (2k+72)Z$, onde $\beta''_k = \frac{3+4p}{4p}$.

Assim, para o dado valor de p, como no caso anterior, podemos concluir que

$$H(\mathcal{P}) \leq \beta_k \cdot \mathrm{OPT}(L) + \left(2k + \frac{597}{8}\right) Z,$$

onde $\beta_k \to \beta'_k(\frac{4}{9}) = \beta''_k = \frac{3+4p}{4p} = 2{,}66076\ldots$ à medida que $k \to \infty$.

O teorema segue das conclusões obtidas em todos os casos analisados. □

Os seguintes resultados provados para o Algoritmo TRI_k também são válidos para este algoritmo e podem ser provados analogamente.

Corolário 5.4.6. *Para qualquer instância L do PET^r e $k \geq 6$ temos*

$$\mathrm{R}_k(L) \leq \gamma_k \cdot \mathrm{OPT}(L) + \left(2k + \frac{597}{8}\right) Z,$$

onde $\gamma_k < 2{,}67$.

Proposição 5.4.7. *O limite de desempenho assintótico do Algoritmo R_k, $k \geq 6$, está entre 2,5 e 2,67.*

Prova. Segue diretamente do Corolário 5.4.6 e do Lema 5.3.17 (usando $m = 4$). □

5.4.2 Empacotamento em Caixa de Fundo Quadrado

Nesta seção, consideraremos o caso do PET^r quando B tem fundo quadrado. Sem perda de generalidade, consideraremos o $\text{PET}^r(1,1)$.

Primeiramente, apresentaremos um algoritmo chamado NFDH_p^{xy}, $0 < p < 1$, que será usado como subrotina. Este algoritmo se assemelha ao Algoritmo NF_p^{xy} desenvolvido para o PEP^r.

Este algoritmo empacota as caixas de L em uma caixa $B = (1, 1, \infty)$, da seguinte maneira. Primeiro, o Algoritmo NFDH_p^{xy} ordena L de forma não-crescente de altura, então ele começa a empacotar cada caixa na ordem dada por essa ordenação, gerando um empacotamento dividido em níveis. O Algoritmo NFDH_p^{xy} divide cada nível em duas partes; primeiro ele empacota na região $[0,1) \times [0,p)$ e então empacota na região $[0,1) \times [1-p,1)$. As caixas são empacotadas na região $[0,1) \times [0,p)$ usando o Algoritmo NFDH^x, até que uma caixa b_i não possa ser empacotada no mesmo nível, então NFDH_p^{xy} usa o Algoritmo NFDH^y para empacotar caixas $\rho(b_i), \rho(b_{i+1}), \ldots$ na região $[0,1) \times [1-p,1)$ até que uma caixa b_k não possa ser empacotada no mesmo nível. Neste ponto, o Algoritmo NFDH_p^{xy} considera as duas partes como sendo um nível e continua a empacotar a caixa b_k em um novo nível. O processo continua até que todas as caixas em L tenham sido empacotadas.

Outra variante do algoritmo acima é chamado de NFDH_p^{yx}. Este Algoritmo é similar ao Algoritmo NFDH_p^{xy}, exceto que NFDH_p^{yx} primeiro empacota caixas b com $x(b) \leq p$, na direção do eixo y, e então empacota as próximas caixas na direção do eixo x.

Antes de apresentar o Algoritmo BS, definiremos outra notação para especificar a partição da lista L. Denote por $\mathcal{X}'$ o conjunto de caixas dado por $\mathcal{X}' = \{b_i = (x_i, y_i, z_i) : \ y_i \leq 1 - x_i\}$.

A estratégia deste algoritmo também é a de dividir a lista de entrada em sublistas, combinar e aplicar algoritmos apropriados para cada uma delas. Quando a caixa B tem fundo quadrado, apenas aplicando algoritmos apropriados para as sublistas, como apresentadas na Figura 5.8, é possível obter um empacotamento dividido em duas partes. Um empacotamento com garantia de área $\frac{4}{9}$ e outro empacotamento ótimo com garantia de área $\frac{1}{4}$. Assim, usamos a estratégia de combinar conjuntos críticos para melhorar uma dessas frações.

Algoritmo BS

Entrada: Lista de caixas L.

Saída: Empacotamento $\mathcal{P}$ de L dentro de $B = (1, 1, \infty)$.

1 Seja $p \leftarrow 0{,}43322958$ e $q \leftarrow 1 - p$.

2 Gire as caixas de L de tal forma que para cada caixa b, $x(b) \leq y(b)$.

3 Divida L nas sublistas $L'_1, L'_2, L'_3, L_A, L_B, L_C, L_4, \ldots, L_{14}$, como segue (veja a Figura 5.8).

$L'_1 \leftarrow L \bigcap \mathcal{C}\,[q, 1\ ;\ q, 1]$, $L_A \leftarrow L \bigcap \mathcal{C}\,[\frac{1}{2}, q\ ;\ \frac{1}{2}, 1]$, $L'_2 \leftarrow L \bigcap \mathcal{C}\,[p, \frac{1}{2}\ ;\ \frac{1}{2}, 1] \setminus \mathcal{X}'$,

$L_B \leftarrow L \bigcap \mathcal{C}\,[\frac{1}{3}, p\ ;\ \frac{1}{2}, 1] \setminus \mathcal{X}'$, $L'_3 \leftarrow L \bigcap \mathcal{C}\,[p, \frac{1}{2}\ ;\ \frac{1}{3}, \frac{1}{2}]$, $L_C \leftarrow L \bigcap \mathcal{C}\,[\frac{1}{3}, p\ ;\ \frac{1}{3}, \frac{1}{2}]$,

$L_4 \leftarrow L \bigcap \mathcal{C}\,[\frac{1}{4}, \frac{1}{3}\ ;\ \frac{2}{3}, 1]$, $L_5 \leftarrow L \bigcap \mathcal{C}\,[\frac{1}{5}, \frac{1}{4}\ ;\ \frac{2}{3}, 1]$, $L_6 \leftarrow L \bigcap \mathcal{C}\,[0, \frac{1}{5}\ ;\ \frac{8}{13}, 1]$,

$L_7 \leftarrow L \bigcap \mathcal{C}\,[\frac{1}{3}, \frac{1}{2}\ ;\ \frac{1}{2}, \frac{2}{3}] \bigcap \mathcal{X}'$, $L_8 \leftarrow L \bigcap \mathcal{C}\,[\frac{1}{4}, \frac{1}{3}\ ;\ \frac{1}{2}, \frac{2}{3}]$, $L_9 \leftarrow L \bigcap \mathcal{C}\,[\frac{1}{5}, \frac{1}{4}\ ;\ \frac{1}{2}, \frac{2}{3}]$,

$L_{10} \leftarrow L \bigcap \mathcal{C}\,[0, \frac{1}{5}\ ;\ \frac{1}{2}, \frac{8}{13}]$, $L_{11} \leftarrow L \bigcap \mathcal{C}\,[\frac{1}{4}, \frac{1}{3}\ ;\ \frac{1}{3}, \frac{1}{2}]$, $L_{12} \leftarrow L \bigcap \mathcal{C}\,[0, \frac{1}{4}\ ;\ \frac{1}{3}, \frac{1}{2}]$,

$L_{13} \leftarrow L \bigcap \mathcal{C}\,[0, \frac{1}{3}\ ;\ \frac{1}{4}, \frac{1}{3}]$, $L_{14} \leftarrow L \bigcap \mathcal{C}\,[0, \frac{1}{4}\ ;\ 0, \frac{1}{4}]$.

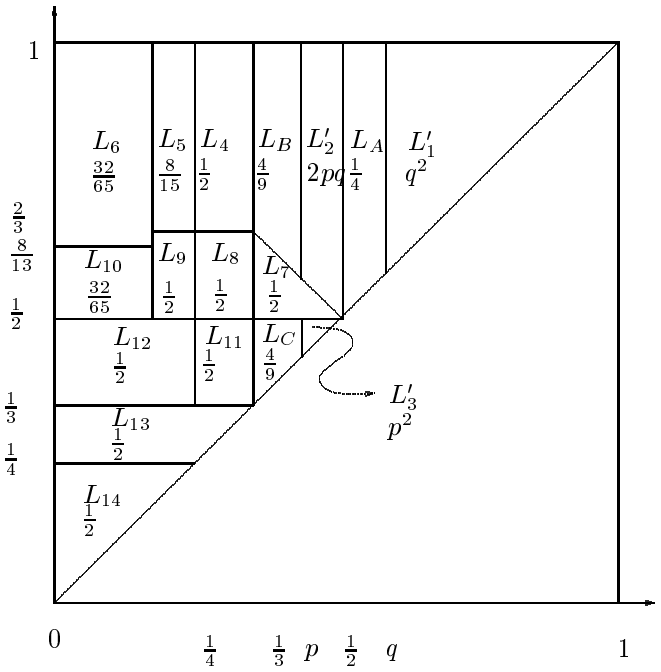


Figura 5.8: Partição da lista L gerada pelo Algoritmo BS.

4 $(\mathcal{P}_{AB}, L_{AB}) \leftarrow \text{COLUMN}(L_A, [(0,0)], L_B, [(q,0)])$;

5 $(\mathcal{P}_{AC}, L_{AC}) \leftarrow \text{COLUMN}(L_A \setminus L_{AB}, [(0,0)], L_C, [(q,0), (q, \frac{1}{2})])$;

6 $L_1 \leftarrow (L'_1 \cup L_A) \setminus (L_{AB} \cup L_{AC})$;

$L_2 \leftarrow (L'_2 \cup L_B) \setminus L_{AB}$;

$L_3 \leftarrow (L'_3 \cup L_C) \setminus L_{AC}$.

7 Obtenha um empacotamento $\mathcal{P}_7$ de L_7 da seguinte maneira.

7.1 Ordene L_7 em ordem não-crescente de altura.

7.2 Construa uma partição de L_7 dado por $L_7^1, L_7^2, \ldots, L_7^{n_7}$ tal que

$$\begin{cases} L_7 & = L_7^1 \| L_7^2 \| \ldots \| L_7^{n_7}, \\ |L_7^i| & = 3, \quad i = 1, \ldots, n_7 - 1, \\ |L_7^{n_7}| & \leq 3. \end{cases}$$

7.3 Gere um empacotamento $\mathcal{P}_7^i$ de L_7^i, $i = 1, \ldots, n_7$ como segue

7.3.1 Escolha $b \in L_7^i$, tal que $x(b)$ é mínimo.

7.3.2 Empacote $\rho(b)$ na posição $(0, 1 - x(b))$ e as caixas em $L_7^i \setminus \{b\}$ nas posições $(0,0)$ e $(\frac{1}{2}, 0)$.

7.4 $\mathcal{P} \leftarrow \mathcal{P}_7^1 \| \ldots \| \mathcal{P}_7^{n_7}$;

8 $\mathcal{P}' \leftarrow \mathrm{OC}(L_1) \| \mathcal{P}_{AB} \| \mathcal{P}_{AC}$;

9 $\mathcal{P}_{aux} \leftarrow \mathrm{NFDH}^x(L_2) \| \ldots \| \mathrm{NFDH}^x(L_6) \| \mathcal{P}_7 \| \mathrm{NFDH}^{xy}_{\frac{2}{3}}(L_8) \| \mathrm{NFDH}^{xy}_{\frac{2}{3}}(L_9) \|$
$\mathrm{NFDH}^{xy}_{\frac{8}{13}}(L_{10}) \| \mathrm{NFDH}^x(L_{11}) \| \ldots \| \mathrm{NFDH}^x(L_{13}) \| \mathrm{LL}(L_{14})$;

10 $\mathcal{P} \leftarrow \mathcal{P}' \| \mathcal{P}_{aux}$.

11 Retorne $\mathcal{P}$.

Fim algoritmo.

Teorema 5.4.8. *Para qualquer lista L para o* PET^r*, onde B tem fundo quadrado, a seguinte inequação é válida*

$$\mathrm{BS}(L) \leq 2{,}5273 \cdot \mathrm{OPT}(L) + 15Z.$$

Prova. Na Figura 5.8 ilustramos a partição de L em sublistas, e os valores em cada sublista representam a área garantida para cada uma delas. A demonstração destas garantias de área segue da mesma forma como feito anteriormente. Talvez o único empacotamento diferente seja o feito para a lista L_7. Observe que da forma como a lista L_7 foi definida, é sempre possível empacotar três caixas de L_7 no mesmo nível. Para os passos 3 e 4, podemos concluir que ou L_A é totalmente empacotado ou $L_B \cup L_C$ é totalmente empacotado em $\mathcal{P}_{AB} \| \mathcal{P}_{AC}$. Assim, dividimos a prova em dois casos.

Caso 1. L_A é totalmente empacotado em $\mathcal{P}_{AB} \| \mathcal{P}_{AC}$.

Temos aqui que

$$\begin{aligned} H(\mathcal{P}') &\leq \frac{1}{q^2} V(L') + 2Z, \\ H(\mathcal{P}_{aux}) &\leq \frac{9}{4} V(L_{aux}) + 13Z, \\ H(\mathcal{P}') &\leq \mathrm{OPT}(L) + 2Z. \end{aligned}$$

Como feito anteriormente, temos $H(\mathcal{P}) \leq \alpha_1 \cdot \mathrm{OPT}(L) + 15Z$, onde $\alpha_1 \leq \frac{\mathcal{H}_1+\mathcal{H}_2}{\max\{\mathcal{H}_1, q^2\mathcal{H}_1+\frac{4}{9}\mathcal{H}_2\}} \leq 2{,}5272\ldots$.

Caso 2. $(L_B \cup L_C)$ é totalmente empacotado em $\mathcal{P}_{AB} \| \mathcal{P}_{AC}$.

Temos aqui que

$$\begin{aligned} H(\mathcal{P}') &\leq 4V(L') + 2Z, \\ H(\mathcal{P}_{aux}) &\leq \frac{1}{2pq}V(L_{aux}) + 13Z, \\ H(\mathcal{P}') &\leq \mathrm{OPT}(L) + 2Z. \end{aligned}$$

Analogamente, temos $H(\mathcal{P}) \leq \alpha_2 \cdot \mathrm{OPT}(L) + 15Z$, onde $\alpha_2 \leq \frac{\mathcal{H}_1+\mathcal{H}_2}{\max\{\mathcal{H}_1, \frac{1}{4}\mathcal{H}_1+2pq\mathcal{H}_2\}} \leq 2{,}5272\ldots$.

O teorema segue dos dois casos acima. □

Proposição 5.4.9. *O limite de desempenho assintótico do Algoritmo BS está entre* 2,5 *e* 2,5273.

Prova. Segue diretamente do Teorema 5.4.8 e do Lema 5.3.17 (usando $m = 4$). □

5.4.3 Empacotamento *On-Line* em Caixa de Fundo Quadrado

Uma das principais aplicações do problema de empacotamento tridimensional ortogonal z-orientado é o problema de escalonamento de processos em computadores paralelos com *topologia* em forma de malha.

Quando os processos a serem escalonados podem ser executados em *batch* isto fica viável para algoritmos *off-line*, mas muitas vezes há uma necessidade de escalonamento dos processos à medida que são requisitados. E neste caso precisamos aplicar algoritmos *on-line*.

Na grande maioria das vezes, a malha de processadores é de dimensão quadrada [42]. Com esta restrição podemos aplicar a mesma idéia usada no algoritmo RR_k para o problema do PET^r e construir um algoritmo com limite de desempenho assintótico que pode se tornar tão próximo de 2,6875 quanto se queira.

O algoritmo que apresentamos, o qual denominamos de $\mathrm{RR}^{(3)}_{k,p}$ usa a mesma técnica de arredondamento sobre a altura das caixas para uma potência de p, $0 < p < 1$, usada para o Algoritmo $\mathrm{SRR}^{(3)}_{m,p}$. Uma vez que as alturas são *arredondadas*, o empacotamento de cada caixa se resume em um problema de empacotamento bidimensional *on-line*. E para isso usaremos o algoritmo RR_k.

Vamos supor que $B = (1, 1, \infty)$ e seja T um número tal que $T \geq \max\{z(c) : c \in L\}$. Normalize a altura T para 1 e todas as caixas de L na mesma proporção. Considere L como sendo a lista já normalizada.

Todos os algoritmos desta seção têm como entrada um parâmetro p, $0 < p < 1$, e um empacotamento produzido por estes algoritmos consiste de níveis, sendo que cada nível tem altura p^i, $i \geq 0$, i inteiro.

Algoritmo $\mathrm{RR}_{k,p}^{(3)}$

Entrada: Lista de caixas $L = (b_1, \ldots, b_n)$.

Saída: Empacotamento de L em caixa $B = (1, 1, \infty)$ para o PET^r.

1 $\mathcal{P} \leftarrow \emptyset$.

2 Para $i = 1$ até n faça

2.1 Seja $j \geq 0$ tal que b_i é uma j-caixa.

2.2 Seja $\mathcal{N}_j$ o conjunto de j-níveis gerados até o momento.

2.3 Use o algoritmo RR_k para empacotar b_i nos níveis $\mathcal{N}_j$, visualizando cada nível de $\mathcal{N}_j$ como uma placa $(1, 1)$ e b_i como um retângulo $(x(b_i), y(b_i))$. Caso necessário, crie um novo nível em $\mathcal{P}$ de forma que se $x(b_i) > \frac{1}{2}$ e $y(b_i) > \frac{1}{2}$ então este nível tem altura $z(b)$, caso contrário, o novo nível criado tem altura p^j.

3 Retorne $\mathcal{P}$.

Fim algoritmo.

Teorema 5.4.10. *Considere uma instância do* PET^r *que consiste de uma lista* L *de caixas a serem empacotadas em uma caixa* $B = (1, 1, \infty)$. *Então,*

$$\mathrm{RR}_{k,p}^{(3)}(L) \leq \alpha_{k,p} \cdot \mathrm{OPT}(L) + \left(\frac{k+14}{1-p}\right) Z,$$

onde $\lim_{k\to\infty, p\to 1} \alpha_{k,p} \leq 2{,}6875$.

Prova. Sejam

$$\begin{aligned} L' &= \{b \in L : x(b) > \frac{1}{2} \text{ e } y(b) > \frac{1}{2}\}, \\ L'' &= L \setminus L', \\ L''_i &= L'' \cap \mathcal{Z}[p^{i+1}, p^i] \end{aligned}$$

e N''_i o número de i-níveis gerados para as caixas de L''_i. Seja h_1 a altura do empacotamento formado pelas caixas de L' e h_2 a soma das alturas dos níveis de $\bigcup_{i\geq 0} N''_i$ menos $\frac{k+14}{1-p}$. Com isso temos que,

$$V(L') > \frac{1}{4} h_1 \tag{5.45}$$

$$\mathrm{OPT}(L) \geq h_1. \tag{5.46}$$

Pela inequação (4.40) temos que

$$S(L_i'') \geq r_1 \cdot (N_i'' - k - 14),$$

e portanto, temos que

$$\begin{aligned} V(L'') &= \sum_{i\geq 0} V(L_i'') \\ &> \sum_{i\geq 0} p^{i+1} S(L_i'') \\ &\geq p \cdot r_1 \sum_{i\geq 0} p^i (N_i'' - k - 14) \\ &= p \cdot r_1 \left(\sum_{i\geq 0} p^i N_i'' - (k+14) \sum_{i\geq 0} p^i \right). \end{aligned}$$

Assim,

$$h_2 = \sum_{i\geq 0} p^i N_i'' - \frac{(k+14)}{1-p} \leq \frac{V(L'')}{p \cdot r_1}.$$

Usando as mesmas técnicas usadas anteriormente, podemos obter que

$$\mathrm{RR}_{k,p}^{(3)}(L) \leq \alpha_{k,p} \cdot \mathrm{OPT}(L) + \left(\frac{k+14}{1-p} \right) Z,$$

onde $\alpha_{k,p} \leq \frac{h_1+h_2}{\max\{h_1, \frac{1}{4}h_1 + p \cdot r_1 \cdot h_2}\}$. Assim, $\lim_{k\to\infty, p\to 1} \alpha_{k,p} \leq 2{,}6875$. □

5.4.4 Algoritmo OTRI

Nesta seção, usamos o Algoritmo OBI_p para desenvolver um algoritmo *on-line* para o problema de empacotamento tridimensional usando rotações ortogonais. Este algoritmo pode ter seu limite de desempenho assintótico tão próximo de 3,25 quanto se queira.

Algoritmo OTRI_p

Entrada: Lista de caixas $L = (c_1, \ldots, c_n)$.

Saída: Empacotamento *on-line* de L em $B = (a, b, \infty)$, permitindo rotações ortogonais.

1 Empacote as caixas $b \in L$ na ordem dada por L.

- **1.1** Seja i tal que $p^{i+1} < \frac{z(b)}{Z} \leq p^i$.
- **1.2** Empacote b nos i-níveis usando o Algoritmo OBI, considerando cada nível como uma placa (a, b) e a caixa b como um retângulo $(x(b), y(b))$.
- **1.3** Caso o Algoritmo OBI não tenha conseguido empacotar b (no passo 1.2) nos níveis já existentes, então empacote b em um novo nível da seguinte maneira.

1.3.1 Se $\{b, \rho(b)\} \cap (\wp_1 \cup \wp_2 \cup \wp_3) \neq \emptyset$ então empacote b em um novo nível de altura p^i usando o Algoritmo OBI.

1.3.2 Caso contrário, empacote b (sozinho) em um novo nível de altura $z(b)$.

5 Retorne $\mathcal{P}$.

Fim algoritmo.

O seguinte resultado é valido para o Algoritmo OTRI$_p$.

Teorema 5.4.11. *Para qualquer instância L do* PET^r*, temos que*

$$\mathrm{OTRI}_p(L) \leq \alpha_p \cdot \mathrm{OPT}(L) + \frac{10}{1-p} Z,$$

onde $\alpha_p \to 3{,}25$ à medida que $p \to 1$.

Prova. Seja L_1 a lista de caixas que foram empacotadas no passo 1.3.2 e $L_2 := L \setminus L_1$, $L_2^i := L_2 \cap \mathcal{Z}[p^{i+1}, p^i]$.

Seja N_i'' o número de i-níveis gerados para as caixas de L_2^i. Seja h_1 a altura do empacotamento formado pelas caixas de L_1 e h_2 a soma das alturas dos níveis gerados para as caixas de L_2 menos $\frac{10}{1-p} Z$. Com isso temos que,

$$\frac{V(L_1)}{ab} > \frac{1}{4} h_1 \tag{5.47}$$

$$\mathrm{OPT}(L) \geq h_1. \tag{5.48}$$

Pela inequação (4.45) temos que

$$\frac{S(L_2^i)}{ab} \geq \frac{1}{3} n_2^i,$$

onde n_2^i é o número de níveis gerados para o empacotamento de L_2^i, menos 10.

Portanto, temos que

$$\begin{aligned} V(L_2) &= \sum_{i \geq 0} V(L_2^i) \\ &\geq \sum_{i \geq 0} p^{i+1} S(L_2^i) \\ &\geq p \cdot \frac{1}{3} \sum_{i \geq 0} p^i (N_i'' - 10) \\ &= p \cdot \frac{1}{3} \left(\sum_{i \geq 0} p^i N_i'' - 10 \sum_{i \geq 0} p^i \right). \\ &= p \cdot \frac{1}{3} \cdot h_2. \end{aligned}$$

Usando as mesmas técnicas usadas anteriormente, podemos obter que

$$\mathrm{OTRI}_p(L) \leq \alpha_p \cdot \mathrm{OPT}(L) + \left(\frac{10}{1-p}\right) Z,$$

onde $\alpha_p \leq \frac{h_1+h_2}{\max\{h_1, \frac{1}{4}h_1+p\cdot\frac{1}{3}\cdot h_2\}}$. Assim, $\lim_{p\to 1} \alpha_p \leq 3{,}25$. □

5.5 Resumo dos Algoritmos

A seguir apresentamos um resumo dos algoritmos, para o problema de empacotamento tridimensional, nas duas tabelas seguintes.

Os algoritmos desenvolvidos nesta tese estão marcados com $\star$ na coluna da referência.

Algoritmos para o PET

Algoritmo	Tipo	α	β	Ref.	Condição
$\mathrm{STP}_m^{(t)}$	*off-line*	$\alpha(\mathrm{STP}_m^{(t)})$	$\mathcal{O}(m \cdot Z)$	$\star$	Caixas de L têm fundo pequeno
$\mathrm{ISRR}_{m,p}$	*on-line*	$\asymp \left(\frac{m+2}{m+1}\right)^2 + \frac{2}{m(m+1)}$	$\mathcal{O}\left(\frac{m^2}{1-p}\right) Z$	$\star$	Caixas de L têm fundo pequeno
LS	*off-line*	2,5425	$\frac{101}{8} Z$	$\star$	Caixas de L têm fundo quadrado
CQQ	*off-line*	2,6875	$2Z$	[40]	Todas as caixas têm fundo quadrado
$\mathrm{SS}_1^{(t)}$	*off-line*	2,3605...	$4Z$	$\star$	Todas as caixas têm fundo quadrado
$\mathrm{SS}_m^{(t)}$	*off-line*	$\alpha(\mathrm{SS}_m^{(t)})$	$4Z$	$\star$	Todas as caixas têm fundo quadrado e pequeno
TRI_k	*off-line*	$\asymp$ 2,6607...	$\left(2k + \frac{597}{8}\right) Z$	$\star$	Caso Geral
$\mathrm{FFLS}_{r,s}$	*on-line*	$\asymp$ 2,89	$\mathcal{O}\left(\frac{1}{(1-r)}\frac{1}{(1-s)}\right) Z$	[44]	Caso Geral

Algoritmos para o PET^r

Algoritmo	Tipo	α	β	Ref.	Condição
C_m^*	*off-line*	$\left(\frac{m+1}{m}\right)^2$	$6Z$	[46]	Caixas de L têm fundo pequeno
$\mathrm{SRR}_{m,p}^{(3)}$	*on-line*	$\asymp \left(\frac{m+1}{m}\right)^2$	$\mathcal{O}\left(\frac{m^2}{1-p}\right) Z$	$\star$	Caixas de L têm fundo pequeno
BS	*off-line*	2,5273...	$15Z$	$\star$	Caixa B tem fundo quadrado
$\mathrm{RR}_{k,p}^{(3)}$	*on-line*	$\asymp$ 2,6875	$\mathcal{O}\left(\frac{k}{1-r}\right) Z$	$\star$	Caixa B tem fundo quadrado
R_k	*off-line*	$\asymp$ 2,6607...	$\left(2k + \frac{597}{8}\right) Z$	$\star$	Caso Geral
OTRI_p	*on-line*	$\asymp$ 3,25	$\mathcal{O}\left(\frac{10}{1-p}\right) Z$	$\star$	Caso Geral

Capítulo 6

Problema de Empacotamento em Contêineres

6.1 Introdução

O tema central deste capítulo é o Problema de Empacotamento em Contêineres. Tratamos aqui duas versões deste problema: a totalmente orientada, denotada por PEC, e a versão em que rotações ortogonais são permitidas, denotada por PEC^r.

Os primeiros a apresentar um algoritmo de aproximação para o PEC foram Coppersmith e Raghavan [12], que desenvolveram um algoritmo com limite de desempenho assintótico 6,25. Para esta mesma versão totalmente orientada, Li e Cheng [39] apresentaram um algoritmo que é uma generalização do Algoritmo H_M do Problema de Empacotamento Unidimensional, obtendo um limite de desempenho assintótico que pode se tornar tão próximo de 4,84 quanto se queira. Este mesmo limite é também atingido por um outro algoritmo desenvolvido por Csirik e van Vliet [14]. Esses dois últimos algoritmos são *on-line*. Este é o melhor limite de desempenho assintótico conhecido para este problema.

Blitz, van Vliet e Woeginger [4] mostraram que algoritmos *on-line* para o PEC (PEC^r) não podem ter limite de desempenho assintótico menor que 2,111.

Não encontramos na literatura resultados sobre algoritmos de aproximação para o PEC^r. Apresentamos um algoritmo para o caso geral do PEC^r com limite de desempenho assintótico não maior que 4,883. Para o caso especial de empacotamentos de cubos em cubos apresentamos um algoritmo com limite de desempenho não maior que 3,467. Para o caso *on-line*, apresentamos um algoritmo com um limite de desempenho assintótico 6,25. Também apresentamos algoritmos para o caso especial onde as caixas têm dimensões pequenas.

6.2 Definições

A seguir, definimos os dois problemas abordados neste capítulo.

Problema. *Problema do Empacotamento em Contêineres (*PEC*): Dadas uma lista de caixas L e caixas $B = (a, b, c)$, encontrar uma partição $L_1, \ldots, L_k$ de L e empacotamentos tridimensionais orientados de L_i em caixas B tal que k seja o menor possível.*

Problema. *Problema do Empacotamento em Contêineres com Rotação (*PEC^r*) Dadas uma lista de caixas L e caixas $B = (a, b, c)$, encontrar uma partição $L_1, \ldots, L_k$ de L e empacotamentos tridimensionais orientados de $L'_i \in \Gamma(L_i)$ em caixas B tal que k seja o menor possível.*

A Figura 6.1 ilustra um empacotamento de caixas em contêiners.

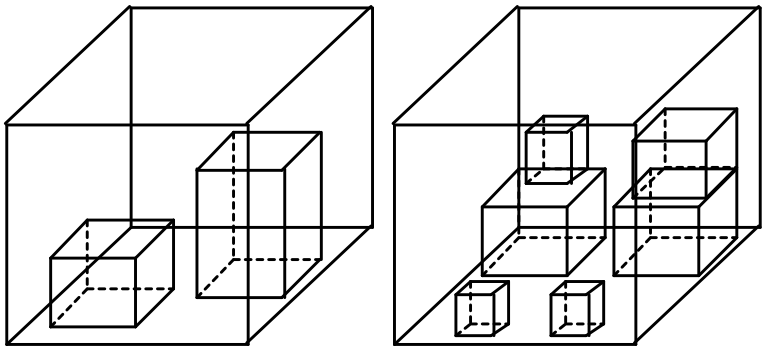


Figura 6.1: Exemplo de um empacotamento de caixas em contêineres.

O seguinte lema apresenta uma relação entre o volume garantido em cada contêiner com seu limite de desempenho assintótico.

Lema 6.2.1. *Seja L uma instância do* PEC *(*PEC^r*) e $\mathcal{P}$ um empacotamento de L onde cada contêiner de $\mathcal{P}$ contém caixas com volume total de pelo menos $v \cdot a \cdot b \cdot c$, exceto talvez em um número constante C de contêineres. Então,*

$$\#(\mathcal{P}) \leq \frac{1}{v}\frac{V(L)}{abc} + C.$$

A constante v mencionada no lema anterior é chamada de *garantia de volume* do empacotamento $\mathcal{P}$.

6.3 Caso Orientado

Nesta seção apresentamos algoritmos, *on-line* e *off-line*, para o caso especial onde todas as caixas de L têm dimensões pequenas; e algoritmos *off-line* para o caso onde as caixas e os contêineres são cubos.

6.3.1 Algoritmo H3D

Descrevemos a seguir um algoritmo para o PEC, denominado H3D *(Hibrid 3-D bin packing algorithm)*. Este algoritmo usa a mesma estratégia apresentada no Algoritmo HFF, do problema de empacotamento em placas. O Algoritmo HFF usa o Algoritmo FFDH para gerar níveis (do PEF) e em seguida aplica o Algoritmo FF do PEU para empacotar os níveis em placas. O Algoritmo H3D contrói seu empacotamento de forma análoga e usa um algoritmo do PET para gerar níveis e em seguida usa um algoritmo do PEU para empacotar estes níveis em contêineres. Os algoritmos do PET e do PEU não são fixos, mas são parte da entrada como subrotinas do Algoritmo H3D. Este algoritmo será usado como subrotina de outros algoritmos descritos nas seções seguintes.

Algoritmo H3D^z

Entrada: Lista de caixas L.

Subrotinas: Algoritmo $\mathcal{A}_{\text{PET}}$ para o PET em níveis e algoritmo $\mathcal{A}_{\text{UNI}}$ para o PEU.

Saída: Empacotamento das caixas de L em contêineres $B = (a, b, c)$.

1 $\mathcal{P} \leftarrow \mathcal{A}_{\text{PET}}(L)$.

2 Seja $\mathcal{N}$ o conjunto de níveis de $\mathcal{P}$.

3 Aplique o algoritmo $\mathcal{A}_{\text{UNI}}$ para empacotar os níveis de $\mathcal{N}$ em contêineres B. Cada nível $N \in \mathcal{N}$, de altura z_N, é visto como uma barra de altura z_N, e cada contêiner B é vista como uma barra de altura c. Seja $\mathcal{P}_{\text{H3D}}$ este empacotamento.

4 Retorne $\mathcal{P}_{\text{H3D}}$.

Fim algoritmo.

Denotamos por H3D^x e H3D^y as variantes deste algoritmo onde a geração dos níveis e o empacotamento unidimensional destes é feito na direção do eixo x e do eixo y, respectivamente. Note que se $\mathcal{A}_{\text{PET}}$ e $\mathcal{A}_{\text{UNI}}$ forem algoritmos *on-line*, tal que os níveis gerados por $\mathcal{A}_{\text{PET}}$ têm altura definida logo na sua geração, então o Algoritmo H3D^z, usando esses algoritmos, poderá ser implementado de forma a se tornar *on-line*.

Denotamos por NFDH$^{(c)}$ o Algoritmo H3D^z que utiliza os algoritmos NFDH$^{(t)}$, do PET, e NF, do PEU, como subrotinas.

6.3.2 Empacotamento de Caixas Pequenas

Vejamos agora o caso em que as caixas da lista L têm dimensões pequenas. Para isso, considere o Algoritmo $\mathrm{BI}_m^{(t)}$ descrito para o PET. Podemos usá-lo como subrotina do H3D para gerar empacotamento de caixas com dimensões pequenas para o PEC.

Algoritmo $\mathrm{H3D}_m(L, \mathcal{A}_{uni})$

Entrada: Lista L de caixas com dimensões não maiores que $\frac{1}{m}$.

Saída: Empacotamento de L em contêineres $B = (1, 1, 1)$.

1 Divida L em duas sublistas L_1 e L_2 tal que

$$L_1 \leftarrow L \cap \mathcal{Z}[\frac{1}{m+1}, \frac{1}{m}]; \qquad L_2 \leftarrow L \setminus L_1;$$

2 $\mathcal{P}_i \leftarrow \mathrm{H3D}(\mathrm{BI}_m^{(t)}, L_i, \mathcal{A}_{uni}), \quad i = 1, 2.$

3 Retorne $\mathcal{P}_1 \cup \mathcal{P}_2$.

Fim algoritmo.

Lema 6.3.1. *Seja L uma lista de caixas com dimensões não maiores que $\frac{1}{m}$, $m \geq 1$. Então*

$$\mathrm{H3D}_m(L, NF) \leq \left(\frac{m+1}{m}\right)^3 V(L) + 14.$$

Prova. O Algoritmo $\mathrm{BI}_m^{(t)}$ garante pelo menos área de $\left(\frac{m}{m+1}\right)^2$ em todos os níveis, exceto talvez em 6 níveis. Considere os níveis gerados pelo empacotamento da lista L_1. Como é possível colocar pelo menos m níveis em cada contêiner B, cada nível contendo caixas com altura pelo menos $\frac{1}{m+1}$, temos que

$$V(L_1) \geq m \cdot \frac{1}{m+1} \cdot \left(\frac{m}{m+1}\right)^2 ((\#(\mathcal{P}_1) - 1) - 6).$$

Assim,

$$\#(\mathcal{P}_1) \leq \left(\frac{m+1}{m}\right)^3 V(L_1) + 7.$$

Da mesma forma, podemos provar o mesmo resultado para o empacotamento gerado para as caixas de L_2.

Seja $\mathcal{P}_2'$ um empacotamento gerado pelo Algoritmo $\mathrm{BI}_m^{(t)}$ para a lista L_2. Note que $\mathcal{P}_2$ é gerado a partir dos níveis de $\mathcal{P}_2'$. Como os níveis em cada contêiner de $\mathcal{P}_2$ cobrem até pelo menos uma altura de $\left(1 - \frac{1}{m+1}\right)$, exceto talvez no último contêiner, temos que

$$H(\mathcal{P}_2') \geq \left(1 - \frac{1}{m+1}\right) \cdot (\#(\mathcal{P}_2) - 1).$$

Como $H(\mathcal{P}_2') \leq \left(\frac{m+1}{m}\right)^2 V(L_2) + 6Z$, pelo Lema 5.3.6, temos que

$$\left(\frac{m+1}{m}\right)^2 V(L_2) + \frac{6}{m+1} \geq H(\mathcal{P}_2') \geq \left(1 - \frac{1}{m+1}\right) \cdot (\#(\mathcal{P}_2) - 1),$$

e portanto,

$$\begin{aligned}(\#(\mathcal{P}_2) - 1) &\leq \left(\frac{m+1}{m}\right)\left[\left(\frac{m+1}{m}\right)^2 V(L_2) + \frac{6}{m+1}\right] \\ &= \left(\frac{m+1}{m}\right)^3 V(L_2) + \frac{6}{m} \\ &\leq \left(\frac{m+1}{m}\right)^3 V(L_2) + 6.\end{aligned}$$

Assim,

$$\#(\mathcal{P}_2) \leq \left(\frac{m+1}{m}\right)^3 V(L_2) + 7.$$

A conclusão segue das inequações obtidas acima para $\#(\mathcal{P}_1)$ e $\#(\mathcal{P}_2)$. □

Teorema 6.3.2. *Seja L uma lista de caixas com dimensões não maiores que $\frac{1}{m}$. Então,*

$$\mathrm{H3D}_m(L, \mathrm{NF}) \leq \left(\frac{m+1}{m}\right)^3 \mathrm{OPT}(L) + 14.$$

A seguir, apresentamos um outro algoritmo para empacotamento de itens pequenos, que faz uma combinação de itens críticos, usando a mesma técnica do Algoritmo STP_m. De fato, este algoritmo pode ser visto como uma generalização do Algoritmo STP_m para o caso de empacotamento em contêineres. Chamamos este algoritmo de SCP_m.

Algoritmo SCP_m

Entrada: Lista de caixas $L \subset \mathcal{C}_m^{xyz}$

Saída: Empacotamento $\mathcal{P}$ das caixas de L em contêineres $B = (1, 1, 1)$.

1 Sejam $p \leftarrow \frac{\sqrt{16m^6+76m^5+141m^4+142m^3+85m^2+28m+4}-2m^3-7m^2-7m-2}{2m^2(2+m^2+3m)}$; $\quad \%\frac{1}{m+2} < p < \frac{1}{m+1}$

$q \leftarrow \frac{1-p}{m}$;

$k \leftarrow \frac{2m^2(m+2)}{4m^3+11m^2+7m+2-\sqrt{85m^2+141m^4+142m^3+28m+16m^6+76m^5+4}}$;

2 $L_A \leftarrow L \cap \mathcal{X}[\frac{1}{m+1}, q] \cap \mathcal{Y}[\frac{1}{m+1}, q] \cap \mathcal{Z}[\frac{1}{m+1}, q]$;

3 $\mathcal{P}_A \leftarrow \mathrm{NFDH}^{(c)}(L_A)$;

$S_A \leftarrow$ Conjunto de contêineres em $\mathcal{P}_A$.

% Note que dentro de cada contêiner de S_A há regiões $(p, 1, 1)$, $(1, p, 1)$ e $(1, 1, p)$ vazias.

4 Seja $L_{B,i}^x \leftarrow L \cap \mathcal{C}^{xyz}[\frac{1}{k}, p; \frac{1}{k}, \frac{1}{m}; \frac{1}{i+1}, \frac{1}{i}]$, $i = m, \ldots, k-1$.

5 Gere um empacotamento combinando caixas de L_A com caixas de $L^x_{B,1} \cup \ldots \cup L^x_{B,k}$ da seguinte maneira.

Aplique o Algoritmo NF^x sobre a lista $L^x_{B,i}$ para empacotar dentro da região $(p,1,1)$ de cada contêiner em S_A, colocando $m \cdot i$ caixas de $L^x_{B,i}$ em cada região $(p,1,1)$, exceto talvez na última região usada.

Seja $\mathcal{P}^x_{AB,i}$ o empacotamento construído, que combina caixas de L_A e $L^x_{B,i}$.

$S_A \leftarrow S_A \setminus \mathcal{P}^x_{AB,i}$;

$\mathcal{P}^x_{AB} \leftarrow \mathcal{P}^x_{AB,1} \| \ldots \| \mathcal{P}^x_{AB,k-m}$.

6 Construa empacotamentos $\mathcal{P}^y_{AB}$ e $\mathcal{P}^z_{AB}$ usando o procedimento descrito no passo 5, trocando as definições de conjunto de forma análoga.

$\mathcal{P}_{AB} \leftarrow \mathcal{P}^x_{AB} \| \mathcal{P}^y_{AB} \| \mathcal{P}^z_{AB}$.

Seja L_{AB} a lista das caixas empacotadas em $\mathcal{P}_{AB}$.

Atualize(L).

7 Para $i = 1, \ldots, k-m$, sejam

$$\begin{array}{rclrcl} \mathcal{X}'_i & := & \{c : \frac{1}{m+i} < x(c) \leq \frac{1}{m+i-1}\}, & \mathcal{X}'_{k-m+1} & := & \{c : 0 < x(c) \leq \frac{1}{k}\}, \\ \mathcal{Y}'_i & := & \{c : \frac{1}{m+i} < y(c) \leq \frac{1}{m+i-1}\}, & \mathcal{Y}'_{k-m+1} & := & \{c : 0 < y(c) \leq \frac{1}{k}\}, \\ \mathcal{Z}'_i & := & \{c : \frac{1}{m+i} < z(c) \leq \frac{1}{m+i-1}\}, & \mathcal{Z}'_{k-m+1} & := & \{c : 0 < z(c) \leq \frac{1}{k}\}. \end{array}$$

$L_{rst} \leftarrow L \cap \mathcal{X}'_r \cap \mathcal{Y}'_s \cap \mathcal{Z}'_t, \quad 1 \leq r,s,t \leq k-m.$

8 Aplique uma variante do Algoritmo NFDH (a que garantir menor limite de desempenho assintótico) sobre as listas L_{rst}, gerando um empacotamento $\mathcal{P}_{rst}$, $(r,s,t) \neq (k,k,k)$.

9 $\mathcal{P}_{kkk} \leftarrow \mathrm{H3D}_{m+k}(L_{kkk})$;

10 Retorne a concatenação de todos os empacotamentos gerados.

Fim algoritmo.

A prova do resultado a seguir é análoga à feita para o Algoritmo STP_m. Assim, apenas apresentamos uma prova bem resumida.

Teorema 6.3.3. *Seja L uma lista de caixas com dimensões não maiores que $\frac{1}{m}$. Então,*

$$\mathrm{SCP}_m(L) \leq \alpha_m \cdot \mathrm{OPT}(L) + \beta_m,$$

onde $\alpha_m = \frac{2m^4+6m^3+9m^2+7m+2+\sqrt{16m^6+76m^5+141m^4+142m^3+85m^2+28m+4}}{2m^2(m+1)^2}$ *e* $\beta_m = \mathcal{O}(m^3)$.

Prova. Consideramos dois casos. Primeiramente abordamos o caso em que todos os itens de L_A são empacotados em L_{AB} (Caso 1); em seguida (Caso 2) analisamos a situação em que isto não acontece.

Caso 1. $L_A \subseteq L_{AB}$.

Neste caso, tanto o empacotamento $\mathcal{P}_{AB}$ quanto o empacotamento $\mathcal{P}_1$ têm garantia de volume de pelo menos $\frac{m^3}{(m+1)^2}q$. Esta garantia é devido ao empacotamento de $\mathcal{P}_1$ que atribui m^3 caixas em cada contêiner (cada caixa com volume de pelo menos $q\frac{1}{m+1}\frac{1}{m+1}$), exceto talvez no último. O empacotamento $\mathcal{P}^x_{AB,i}$, $i = m, \ldots, k-1$ tem pelo menos garantia de volume $m^3\left(\frac{1}{m+1}\right)^3 + m \cdot i\frac{1}{m+1}\frac{1}{i+1}\frac{1}{k} \geq \left(\frac{m}{m+1}\right)^3 + \frac{m^2}{k(m+1)^2}$. Substituindo o valor de k, temos que esta garantia de volume é pelo menos $\frac{m^3}{(m+1)^2}q$. Os demais empacotamentos têm garantia de volume de pelo menos $\frac{m^2(m+1)}{(m+1)^2(m+2)}$. Este mínimo é atingido para caixas na região $\mathcal{C}^{xyz}[\frac{1}{m+1}, \frac{1}{m}; \frac{1}{m+1}, \frac{1}{m}; \frac{1}{m+2}, \frac{1}{m+1}]$ (ou em regiões simétricas a esta).

Assim, temos o empacotamento dividido em duas partes. Uma parte, um empacotamento assintoticamente ótimo com garantia de volume $\frac{qm^3}{(m+1)^2}$, e outra parte com garantia de volume $\frac{m^2(m+1)}{(m+1)^2(m+2)}$.

Usando esses resultados, podemos obter a inequação desejada.

Caso 2. $L_B \subseteq L_{AB}$.

Neste caso, podemos conseguir uma garantia de volume de pelo menos $\frac{m^3}{(m+1)^3}$ para o empacotamento $\mathcal{P}_1 \| \mathcal{P}_{AB}$ (que é um empacotamento assintoticamente ótimo), e de $\frac{m^2(m+1)}{(m+1)^2}p$ para os demais empacotamentos. Este mínimo sendo atingido por caixas na região $\mathcal{C}^{xyz}[\frac{1}{m+1}, \frac{1}{m}; \frac{1}{m+1}, \frac{1}{m}; p, \frac{1}{m+1}]$. Estes valores nos permitem provar a inequação desejada, com α_m como especificado.

A partir da prova dos dois casos acima, podemos concluir que este algoritmo tem um limite de desempenho assintótico α_m. O valor de p foi tomado de modo a termos o mesmo limite para ambos os casos.

O valor de k foi tomado de forma a ser o menor inteiro positivo tal que $\frac{m^2(m+1)p}{(m+1)^2} \leq \left(1 - \frac{1}{k}\right) m^2 \left(\frac{1}{m+1}\right)^2$ e $\left(\frac{m}{m+1}\right)^3 + \frac{m^2}{k(m+1)^2} \geq \frac{m^3}{(m+1)^2}q$. A constante aditiva acompanhando o limite de desempenho assintótico é $\mathcal{O}(k^3)$. Como $k = \mathcal{O}(m)$ temos que $\beta_m = \mathcal{O}(m^3)$. □

A Tabela 6.1 apresenta valores de α_m (limites de desempenho assintótico do Algoritmo SCP_m), para m entre 1 e 10. Indicamos também o respectivo valor de $p = p(m)$.

6.3.3 Empacotamento *On-Line* de Caixas Pequenas

Este algoritmo usa idéia análoga à do Algoritmo $\mathrm{OFF}^{(2)}_m$. A lista de entrada L é particionada em 3 sublistas. O algoritmo mantém três tipos de contêineres, uma para cada sublista.

As caixas da primeira lista têm altura no intervalo $(1/(m+1), 1/m]$. Assim, em cada contêiner são alocados m níveis. As caixas da primeira lista são empacotadas nos níveis deste tipo usando

m	$\alpha(\mathrm{SCP}_m)$	$p = p(m)$
1	6,022634...	0,348422...
2	3,015766...	0,261824...
3	2,232760...	0,208430...
4	1,882391...	0,172825...
5	1,685433...	0,147514...
6	1,559705...	0,128631...
7	1,472655...	0,114015...
8	1,408887...	0,102374...
9	1,360197...	0,092884...
10	1,321820...	0,085001...

Tabela 6.1: Valores de $\alpha(\mathrm{SCP}_m)$ e $p = p(m)$ para valores de m entre 1 e 10.

o algoritmo $\mathrm{OFF}_m^{(2)}$. O mesmo é feito para a segunda lista, sendo que as caixas desta lista têm altura no intervalo $(1/(m+2), 1/(m+1)]$. Para a terceira lista, é usado um esquema de arredondamento para uma potência de p, $0 < p < 1$, como feito para o Algoritmo $\mathrm{OFF}_m^{(2)}$. Novamente, para não deixar o fator de perda constante, devido ao arredondamento, p é uma função de m.

Algoritmo $\mathrm{OFF}_m^{(3)}$

Entrada: Lista L de caixas L com dimensões no máximo $\frac{1}{m}$.
Saída: Empacotamento de L em contêineres $B = (1, 1, 1)$.

1 Divida a lista L em sublistas $L_1, \ldots, L_3$ da seguinte forma

$L_1 \leftarrow L \cap \mathcal{Z}[\frac{1}{m+1}, \frac{1}{m}]$;
$L_2 \leftarrow L \cap \mathcal{Z}[\frac{1}{m+2}, \frac{1}{m+1}]$;
$L_3 \leftarrow L \setminus (L_1 \cup L_2)$.

2 Gere o empacotamento $\mathcal{P}_1$ de L_1 da seguinte maneira: Considere os níveis de altura $\frac{1}{m}$ gerados até o momento. Empacote a próxima caixa não empacotada de L_1 nestes níveis usando o Algoritmo $\mathrm{OFF}_m^{(2)}$. Caso necessite, gere um novo nível com altura $\frac{1}{m}$ no último contêiner, ou em um contêiner novo, caso precise.

3 Gere o empacotamento $\mathcal{P}_2$ de L_2 de forma análoga ao construído no passo 2, mas com níveis de altura $\frac{1}{m+1}$.

4 $\mathcal{P}_3 \leftarrow \mathrm{H3D}^z(\mathrm{SRR}_{m,p}^{(3)}, L_3, \mathrm{FF})$, com $p = \frac{m(m+2)}{(m+1)^2}$;

5 $\mathcal{P} \leftarrow \mathcal{P}_1 \cup \ldots \cup \mathcal{P}_3$.

6 Retorne $\mathcal{P}$.

Fim algoritmo.

Novamente, este algoritmo não é *on-line* da forma como foi descrito. Mas sua transformação para *on-line* é simples. Deixamos esta transformação para o leitor interessado.

Lema 6.3.4. *Para toda lista L de caixas com dimensões no máximo $\frac{1}{m}$, $m \geq 2$, temos que*

$$\mathrm{OFF}_m^{(3)}(L) \leq \left(\frac{m+1}{m}\right)^3 V(L) + \mathcal{O}(m^4).$$

Prova. Seja N_i o número de níveis gerados para empacotar L_i, $i = 1, 2$. Primeiramente, vamos analisar o empacotamento $\mathcal{P}_1$ de L_1. Como é usado o Algoritmo $\mathrm{OFF}_m^{(2)}$ para empacotar caixas de L_1 em níveis de altura $\frac{1}{m}$, temos

$$\begin{aligned} V(L_1) &\geq \left(\frac{1}{m+1}\right) \cdot S(L_1) \\ &\geq \left(\frac{1}{m+1}\right)\left(\frac{m}{m+1}\right)^2 \cdot (N_1 - \mathcal{O}(m^2)) \quad \text{(pelo Lema 4.4.8)} \\ &\geq \left(\frac{1}{m+1}\right)\left(\frac{m}{m+1}\right)^2 \cdot (m \cdot (\#(\mathcal{P}_1) - 1) - \mathcal{O}(m^2)) \\ &= \left(\frac{m}{m+1}\right)^3 \#(\mathcal{P}_1) - \mathcal{O}(m). \end{aligned}$$

O seguinte resultado pode ser provado de maneira análoga para o empacotamento $\mathcal{P}_2$ de L_2.

$$V(L_2) \geq \left(\frac{m}{m+1}\right)^3 \#(\mathcal{P}_2) - \mathcal{O}(m).$$

Considere agora o empacotamento $\mathcal{P}_3$ de L_3. Seja L_3^i, $i \geq 0$, as i-caixas de L_3 e N_3^i o número de i-níveis construídos para L_3^i. Note que o Algoritmo $\mathrm{SRR}_{m,p}^{(3)}$ usa o Algoritmo $\mathrm{OFF}_m^{(2)}$ para empacotar as caixas em níveis. Assim, pelo Lema 4.4.7 temos

$$\begin{aligned} V(L_3^i) &\geq p^{i+1} S(L_3^i) \\ &\geq p^{i+1}\left[\left(\frac{m}{m+1}\right)^2 N_3^i - \mathcal{O}(m^2)\right]. \end{aligned}$$

Logo,

$$\begin{aligned} V(L_3) &= \sum_{i\geq 0} V(L_3^i) \\ &\geq \sum_{i\geq 0} p^{i+1}\left[\left(\frac{m}{m+1}\right)^2 N_3^i - \mathcal{O}(m^2)\right] \end{aligned}$$

$$
\begin{aligned}
&= p\left[\left(\frac{m}{m+1}\right)^2 \sum_{i\geq 0} p^i N_3^i - \mathcal{O}\left(\frac{m^2}{1-p}\right)\right] \\
&\geq p\left[\left(\frac{m}{m+1}\right)^2 \left(1-\frac{1}{m+2}\right)(\#(\mathcal{P}_3)-1) - \mathcal{O}\left(\frac{m^2}{1-p}\right)\right].
\end{aligned}
$$

Substituindo p, temos

$$\#(\mathcal{P}_3) \leq \left(\frac{m+1}{m}\right)^3 V(L_3) + \mathcal{O}(m^4).$$

O teorema segue das inequações provadas para os empacotamentos $\mathcal{P}_1$,$\mathcal{P}_2$ e $\mathcal{P}_3$. □

Com isto o seguinte resultado é imediato.

Teorema 6.3.5. *Para toda lista de caixas L, onde nenhuma caixa de L tem dimensão maior que $\frac{1}{m}$, $m \geq 2$, vale a seguinte inequação para o empacotamento em contêineres $(1,1,1)$.*

$$\mathrm{OFF}_m^{(3)}(L) \leq \left(\frac{m+1}{m}\right)^3 \mathrm{OPT}(L) + \mathcal{O}(m^4).$$

Usando a mesma estratégia usada para o Algoritmo IOFF_m, podemos refinar este algoritmo de modo a termos um limite de desempenho melhor que o apresentado para o Algoritmo $\mathrm{OFF}_m^{(3)}$. Vamos chamar este algoritmo melhorado de $\mathrm{IOFF}_m^{(3)}$. Este algoritmo procede da seguinte maneira. Primeiro, o algoritmo divide a lista L em duas sublistas, L_1 e L_2, onde

$$
\begin{aligned}
L_1 &\leftarrow L \cap \mathcal{X}[\frac{1}{m+1}, \frac{1}{m}] \cap \mathcal{Y}[\frac{1}{m+1}, \frac{1}{m}] \cap \mathcal{Z}[\frac{1}{m+1}, \frac{1}{m}], \\
L_2 &\leftarrow L \setminus L_1.
\end{aligned}
$$

Para a lista L_1, o Algoritmo $\mathrm{IOFF}_m^{(3)}$ empacota m^3 caixas em cada contêiner, exceto talvez no último. Como cada caixa tem volume maior que $\left(\frac{1}{m+1}\right)^3$, o empacotamento de L_1 tem garantia de volume de pelo menos $\left(\frac{m}{m+1}\right)^3$.

Subdividindo a lista L_2 e usando os algoritmos IOFF_m e $\mathrm{OFF}_m^{(3)}$ podemos obter um empacotamento de L_2 com garantia de volume $m \cdot \left(\frac{1}{m+1}\right) \cdot m \cdot \left(\frac{1}{m+1}\right) \cdot (m+1) \cdot \left(\frac{1}{m+2}\right) = \frac{m^2}{(m+1)(m+2)}$.

Por fim, o Algoritmo $\mathrm{IOFF}_m^{(3)}$ retorna a concatenação dos empacotamentos das listas L_1 e L_2.

O seguinte resultado pode ser provado para este algoritmo.

Teorema 6.3.6. *Para toda lista de caixas L, onde nenhuma caixa de L tem dimensão maior que $\frac{1}{m}$, $m \geq 2$, a seguinte inequação é válida.*

$$\mathrm{IOFF}_m^{(3)}(L) \leq \alpha_m \cdot \mathrm{OPT}(L) + \mathcal{O}(m^4),$$

onde $\alpha_m = \frac{h_1+h_2}{\max\{h_1, \left(\frac{m}{m+1}\right)^3 h_1 + \left(\frac{m^2}{(m+1)(m+2)}\right) h_2\}} \leq \frac{m^4+5m^3+10m^2+7m+2}{m^2(m+1)^2}$.

6.3.4 Empacotamento de cubos em cubos

Meir e Moser [45] provaram que se um conjunto $L = (b_1, \ldots, b_n)$ de cubos, tem volume $V(L) \leq l^3 + (1-l)^3$, onde l é a dimensão do maior cubo de L, então, L pode ser empacotado em um cubo unitário pelo Algoritmo NFDH$^{(c)}$. De fato, eles provaram mais que isto,

Teorema 6.3.7. *Qualquer lista de cubos k-dimensionais L, com dimensões $x_1 \geq x_2 \geq \cdots \geq x_n \geq \cdots$ pode ser empacotado pelo Algoritmo* NFDH$^{(c)}$ *dentro de um retângulo k-dimensional de tamanho $a_1 \times a_2 \times \ldots \times a_k$ se $a_j > x_1$, $j = 1, \ldots, k$ e $x_1^k + (a_1 - x_1)(a_2 - x_1)\cdots(a_k - x_1) \geq V(L)$.*

Usando este resultado, é fácil derivar um algoritmo para empacotar cubos de dimensões pequenas dentro de cubos (contêineres). Usamos a mesma estratégia usada anteriormente no caso tridimensional. Dividimos a lista de entrada em duas partes. Aplicamos um algoritmo trivial na primeira e usamos o resultado do Teorema 6.3.7 para empacotar a segunda lista (que contém cubos menores que a primeira lista).

Algoritmo CUB$_m(L)$

Entrada: Lista de cubos $L = (b_1, \ldots, b_n)$ onde $x(b_i) \leq \frac{1}{m}$

Saída: Empacotamento $\mathcal{P}$ de L em cubos $B = (1, 1, 1)$

1 Particione L em duas sublistas L_1 e L_2 da seguinte maneira.

$$\begin{aligned} L_1 &\leftarrow \{b \in L : \frac{1}{m+1} < x(b) \leq \frac{1}{m}\}, \\ L_2 &\leftarrow \{b \in L : 0 < x(b) \leq \frac{1}{m+1}\}. \end{aligned}$$

2 $\mathcal{P} \leftarrow$ NFDH$^{(c)}(L_1)$.

3 Gere um empacotamento $\mathcal{P}_2$ da lista L_2 da seguinte maneira:

3.1 Particione a lista L_2 em sublistas $L_2^1, L_2^2, \ldots, L_2^k$, $L_2 = L_2^1 \| \ldots \| L_2^k$, tal que

$$\begin{aligned} L_2^i &= (b_1^i, b_2^i, \ldots, b_{n_i}^i), \\ S(L_2^i) &\leq \left(\frac{1}{m'}\right)^3 + \left(1 - \frac{1}{m'}\right)^3, \quad i = 1, \ldots, k \quad \text{e} \\ S(L_2^i) + S(b_1^{i+1}) &> \left(\frac{1}{m'}\right)^3 + \left(1 - \frac{1}{m'}\right)^3, \quad i = 1, \ldots, k-1, \end{aligned}$$

onde $m' = m + 1$.

3.2 $\mathcal{P}_2^i \leftarrow$ NFDH$^{(c)}(L_2^i)$, para $i = 1, \ldots, k$;

3.3 $\mathcal{P}_2 \leftarrow \mathcal{P}_2^1 \| \ldots \| \mathcal{P}_2^k$.

4 Retorne $\mathcal{P}_1 \| \mathcal{P}_2$.

Fim algoritmo.

Lema 6.3.8. *Para toda lista de cubos $L = (b_1, \ldots, b_n)$ tal que $x(b_i) \leq \frac{1}{m}$, $m \geq 2$, a seguinte desigualdade é válida para o empacotamento em cubos de dimensões (1,1,1).*

$$\mathrm{CUB}_m(L) \leq \left(\frac{m+1}{m}\right)^3 V(L) + 2.$$

Prova.

É fácil ver que para o empacotamento gerado para a lista L_1 temos em cada contêiner, exceto talvez no último, pelo menos m^3 cubos, cada um com volume $\left(\frac{1}{m+1}\right)^3$. Assim, temos

$$\#(\mathcal{P}_1) \leq \left(\frac{m+1}{m}\right)^3 V(L_1) + 1. \tag{6.1}$$

Para o empacotamento da lista L_2, temos que esta é subdividida em sublistas $L_2^1, \ldots, L_2^k$. Note que pelo Teorema 6.3.7 cada sublista L_2^i é empacotada pelo Algoritmo NFDH$^{(c)}$ em um cubo unitário. Assim, temos que $k = \#(\mathcal{P}_2)$ e $V(L_2) \geq \left(1 - \frac{1}{m+1}\right)^3$, $i = 1, 2, \ldots, k-1$. Portanto, temos

$$\#(\mathcal{P}_2) \leq \left(\frac{m+1}{m}\right)^3 V(L_2) + 1. \tag{6.2}$$

O lema segue das inequações (6.1) e (6.2). □

Teorema 6.3.9. *Para toda lista de cubos $L = (b_1, \ldots, b_n)$ tal que $x(b_i) \leq \frac{1}{m}$, $m \geq 2$, a seguinte desigualdade é válida para o empacotamento em cubos (contêineres) de dimensões (1,1,1).*

$$\mathrm{CUB}_m(L) \leq \left(\frac{m+1}{m}\right)^3 \mathrm{OPT}(L) + 2.$$

Proposição 6.3.10. *O limite de desempenho assintótico $\left(\frac{m+1}{m}\right)^3$ do Algoritmo CUB_m é justo.*

Prova. Para provar este resultado, considere uma lista de cubos $L_2 = L_2^1 \| L_2^2 \| \ldots \| L_2^k$ onde $L_2^i = (X, Y_1, Y_2, \ldots, Y_t)$ onde $X = \left(\frac{1}{m+1}, \frac{1}{m+1}, \frac{1}{m+1}\right)$ e $Y_j = (\epsilon, \epsilon, \epsilon)$, onde ϵ é um valor pequeno, múltiplo de $\left(\frac{1}{m+1}\right)$ e $V(X) + \sum_{j=1}^{t} V(Y_j) = \left(1 - \frac{1}{m+1}\right)^3 + \epsilon^3$.

Note que em um empacotamento gerado pelo Algoritmo CUB_m, o volume ocupado por cada contêiner é de $\left(1 - \frac{1}{m}\right)^3 + \epsilon^3$. Por outro lado, um empacotamento ótimo pode preencher totalmente um contêiner, exceto talvez o último contêiner. Assim, colocando k grande e ϵ bem pequeno, é possível mostrar que o empacotamento gerado por esta instância apresenta limite de desempenho assintótico que pode se tornar tão próximo de $\left(\frac{m+1}{m}\right)^3$ quanto se queira. □

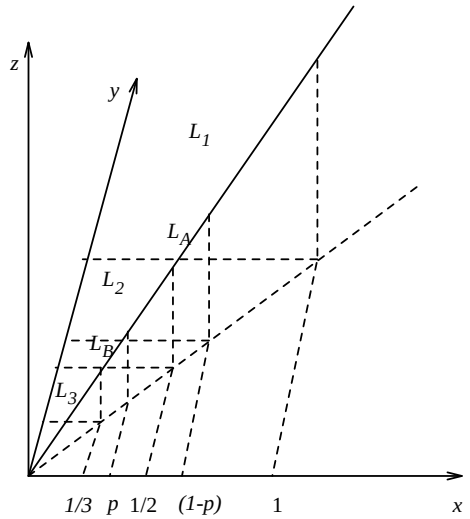


Figura 6.2: Conjunto de listas usadas no Algoritmo CUBO.

A seguir, usamos a mesma técnica de combinar conjuntos críticos para o caso especial de empacotamento de cubos em cubos.

Algoritmo $\mathrm{CUBO}_1(L)$

Entrada: Lista de cubos L.

Saída: Empacotamento $\mathcal{P}$ de L em cubos $B = (1, 1, 1)$.

1 Sejam $L_1, \ldots, L_3, L_A, L_B$ sublistas de L da seguinte maneira.

$$L_i \leftarrow L \cap \mathcal{X}[\tfrac{1}{i+1}, \tfrac{1}{i}] \quad i = 1, 2, \quad L_3 \leftarrow L \cap \mathcal{X}[0, , \tfrac{1}{3}],$$
$$L_A \leftarrow L_1 \cap \mathcal{X}[\tfrac{1}{2}, (1-p)], \qquad L_B \leftarrow L_2 \cap \mathcal{X}[\tfrac{1}{3}, p],$$
$$\text{onde } p = 0,354014.$$

2 Gere um empacotamento parcial $\mathcal{P}_{AB}$ de $L_A \cup L_B$, tal que $\mathcal{P}_{AB}$ é a concatenação dos empacotamentos $\mathcal{P}^1_{AB}, \ldots, \mathcal{P}^k_{AB}$, onde $\mathcal{P}^i_{AB}$ é um empacotamento em um contêiner e contém um cubo de L_A e sete cubos de L_B, exceto talvez no último contêiner, que pode ter menos cubos de L_B. O empacotamento $\mathcal{P}_{AB}$ deve conter, ou todos os cubos de L_A ou todos os cubos de L_B.

Atualize(L_1, L_2).

3 $\mathcal{P}_i \leftarrow \mathrm{NFDH}^{(c)}(L_i)$, $i = 1, 2$;

4 $\mathcal{P}_3 \leftarrow \mathrm{CUB}_3(L_3)$;

5 $\mathcal{P}_{aux} \leftarrow \mathcal{P}_2 \| \mathcal{P}_3 \| \mathcal{P}_{AB}$.

6 Retorne $\mathcal{P}_1 \| \mathcal{P}_{aux}$.

Fim algoritmo.

Teorema 6.3.11. *Para toda lista de cubos L, vale a seguinte desigualdade para o empacotamento de L em cubos $B = (1, 1, 1)$.*

$$\mathrm{CUBO}_1(L) \leq 3{,}466 \cdot \mathrm{OPT}(L) + 4.$$

Prova. Primeiramente, considere o empacotamento $\mathcal{P}_{AB}$. Como cada contêiner de $\mathcal{P}_{AB}$, exceto talvez o último, contém um cubo de L_A e sete de L_B, temos que

$$\#(\mathcal{P}_{AB}) \leq \frac{1}{83/216} V(L_{AB}) + 1, \tag{6.3}$$

onde L_{AB} é o conjunto de cubos de L empacotados em $\mathcal{P}_{AB}$.

Considere o empacotamento $\mathcal{P}_3$ da lista L_3. Pelo Lema 6.3.8, temos

$$\#(\mathcal{P}_3) \leq \frac{1}{27/64} V(L_3) + 2. \tag{6.4}$$

A seguir, dividimos a demonstração em dois casos, conforme o passo 2 do algoritmo. Primeiro se todos os cubos de L_B foram empacotados em $\mathcal{P}_{AB}$ (Caso 1), em seguida se todos os cubos de L_A foram empacotados em $\mathcal{P}_{AB}$ (Caso 2).

Caso 1. L_B é totalmente empacotado em $\mathcal{P}_{AB}$.

Como todos os cubos de L_1 têm pelo menos $\frac{1}{8}$ de volume,

$$\#(\mathcal{P}_1) \quad \leq \quad \frac{1}{1/8} V(L_1). \tag{6.5}$$

Neste caso, todos os cubos de L_2 têm dimensão de pelo menos p. Como é empacotado oito cubos de L_2 por contêiner, temos

$$\#(\mathcal{P}_2) \quad \leq \quad \frac{1}{8p^3} V(L_2) + 1. \tag{6.6}$$

De (6.3), (6.4) e (6.6), e como $8p^3 = \min\{8p^3, \frac{83}{216}, \frac{27}{64}\}$, obtemos

$$\#(\mathcal{P}_{aux}) \quad \leq \quad \frac{1}{8p^3} V(L_{aux}) + 4. \tag{6.7}$$

Note que só podemos empacotar um cubo de L_1 por contêiner, portanto

$$\#(\mathcal{P}_1) \quad \leq \quad \mathrm{OPT}(L). \tag{6.8}$$

Procedendo da mesma forma como feito anteriormente, definindo $h_1 = \#(\mathcal{P}_1)$ e $h_2 = \#(\mathcal{P}_{aux}) - 4$, temos a partir das inequações (6.5), (6.7) e (6.8) que

$$\mathrm{CUBO}_1(L) \leq \alpha' \cdot \mathrm{OPT}(L) + 4,$$

onde $\alpha' = \frac{h_1+h_2}{\max\{h_1, \frac{1}{8}h_1+8p^3h_2\}} \leq 3{,}4652\ldots$.

Caso 2. L_A é totalmente empacotado em $\mathcal{P}_{AB}$.

Neste caso, a garantia de volume para os cubos de L_1 é melhor que a conseguida no Caso 1. Cada cubo de L_1 tem pelo menos dimensão de $(1-p)$. Assim,

$$\#(\mathcal{P}_1) \quad \leq \quad \frac{1}{(1-p)^3} V(L_1).$$

Para o empacotamento $\mathcal{P}_2$, obtemos uma garantia de pelo menos $\frac{8}{27}$ de volume. O mesmo é válido para os empacotamentos $\mathcal{P}_3$ e $\mathcal{P}_{AB}$. Portanto,

$$\#(\mathcal{P}_{aux}) \quad \leq \quad \frac{1}{8/27} V(L_{aux}) + 4.$$

Do mesmo modo como no caso 1, não podemos empacotar dois cubos de L_1 em um mesmo contêiner. Com isso, obtemos

$$\#(\mathcal{P}_1) \quad \leq \quad \mathrm{OPT}(L).$$

A partir destas inequações e procedendo como feito para o caso 1, é possível obter que

$$\mathrm{CUBO}_1(L) \leq \alpha'' \cdot \mathrm{OPT}(L) + 4,$$

onde $\alpha' = \frac{h_1+h_2}{\max\{h_1, (1-p)^3h_1+\frac{8}{27}h_2\}} \leq 3{,}4652\ldots$. □

Proposição 6.3.12. *O limite de desempenho assintótico do Algoritmo $CUBO_1$ está entre* 3,074 *e* 3,466.

Prova. Considere a seguinte instância. $L = L_1 \| L_2$, onde $L_1 = (X_1, X_2, \ldots, X_{n_1})$; $X_i = \left(\frac{1}{2}+\epsilon, \frac{1}{2}+\epsilon, \frac{1}{2}+\epsilon\right)$ e L_2 é a mesma instância construída para a Proposição 6.3.10 com os valores de $k = \frac{56}{27}n_1$ e $m = 3$.

A idéia por trás desta instância é obter um empacotamento ótimo de L em n_1 contêineres, sendo que cada contêiner deste empacotamento ótimo contém uma caixa de L_1 e todo o espaço restante é preenchido com caixas de L_2. Assim, colocando ϵ bem pequeno e n_1 grande, temos que $\frac{\mathrm{CUBO}_1(L)}{\mathrm{OPT}(L)}$ pode se tornar tão próximo de $\frac{n_1+\frac{56}{27}n_1}{n_1}$ quanto se queira. I.e., $\frac{83}{27} = 3{,}074\ldots$. □

Novamente, o Algoritmo CUBO_1 pode ser generalizado para o empacotamento de cubos em $\mathcal{Q}_m^{xyz}$. A seguinte tabela apresenta os limites que podem ser conseguidos para m entre 1 e 10.

6.4 Caso Com Rotações

Nesta seção, abordamos o problema de empacotamento em contêineres, permitindo-se rotação ortogonal em torno de qualquer um dos eixos. Apresentamos dois algoritmos para o caso geral

m	$\alpha(\text{CUBO}_m)$	p
1	3,46520933...	0,35401480
2	2,42362851...	0,26355815
3	1,98710755...	0,20916664
4	1,75134789...	0,17320333
5	1,60493006...	0,14773256
6	1,50557632...	0,12876848
7	1,43390862...	0,11410801
8	1,37984938...	0,10243878
9	1,33765946...	0,09293160
10	1,30383840...	0,08503735

Tabela 6.2: Valores de $\alpha(\text{CUBO}_m)$ e $p = p(m)$ para valores de m entre 1 e 10.

deste problema. Um algoritmo *off-line* que pode ter um limite de desempenho assintótico tão próximo de 4,883 quanto se queira e outro, *on-line*, com um limite de desempenho assintótico 6,25.

6.4.1 Algoritmo $\text{BOX}_{k,\epsilon}$

Apresentamos um algoritmo, chamado $\text{BOX}_{k,\epsilon}$ com um limite de desempenho assintótico que pode se tornar tão próximo de 4,882... quanto possível. Este algoritmo é *off-line* e é descrito para o problema de empacotar uma lista de caixas L em contêineres $B = (a, b, c)$ onde são permitidas rotações ortogonais em torno de qualquer eixo.

Antes de apresentarmos este algoritmo precisamos de alguns procedimentos que serão usados como subrotinas deste algoritmo. Vamos usar o mesmo esquema do Algoritmo COLUMN^r usado para o PET^r. Assim, vamos apresentar uma modificação deste algoritmo para o problema de empacotamento em contêineres. Chamamos este algoritmo de FFC^{xy} (*First Fit COLUMN* for x and y axis).

O Algoritmo FFC^{xy} usa a estratégia do Algoritmo COLUMN com a estratégia do Algoritmo FF (*First Fit* do problema de empacotamento unidimensional) para empacotar as caixas sobre as colunas formadas pelo Algoritmo Combine.

Os parâmetros de entrada são uma lista de caixas L, dois conjuntos de caixas $\mathcal{T}_1$ e $\mathcal{T}_2$ e duas listas de coordenadas p_1 e p_2 associadas a estes conjuntos. Cada coluna começa no fundo de uma caixa B em uma coordenada $p \in p_1 \cup p_2$. As colunas situadas nas coordenadas p_i terão as caixas do conjunto $L \cap \mathcal{T}_i$, $i = 1, 2$ e começam no plano xy e crescem em direção ao eixo z.

Quando estivermos abordando o PEC^r, dizemos que uma caixa $b_i = (x_i, y_i, z_i)$ é do tipo $\mathcal{T}$ se alguma permutação (x'_i, y'_i, z'_i) de (x_i, y_i, z_i) é tal que $(x'_i, y'_i, z'_i) \in \mathcal{T}$ e $x'_i \leq a$, $y'_i \leq b$ e $z'_i \leq c$, onde a, b e c são as dimensões máximas permitidas para os eixos x, y e z, respectivamente.

Algoritmo FFC^{xy}

Entrada: $(L, \mathcal{T}_1, \mathcal{T}_2, [p_1], [p_2])$ % p_i são listas de pontos.

Saída: Empacotamento onde, ou todas as caixas do tipo $\mathcal{T}_1$ ou todas as caixas do tipo $\mathcal{T}_2$ são totalmente empacotadas.

1 Enquanto todas as caixas de ambos os tipos não forem totalmente empacotadas, faça

- **1.1** Sejam $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_i$ os empacotamentos em contêineres $B_1, \ldots, B_i$ feitas até o momento, respectivamente.
- **1.2** Tente empacotar a próxima caixa b' do tipo $\mathcal{T}_1$ em alguma coluna de caixas de L_1 em $\mathcal{P}_1, \ldots, \mathcal{P}_i$ na ordem em que foram criadas, e sem violar os limites de cada contêiner B_j, que contém o empacotamento $\mathcal{P}_j$. Caso necessite, gire a caixa b' de modo a termos $b' \in \mathcal{T}_1$.
- **1.3** Se falhar no passo 1.2, tente empacotar a próxima caixa b'', do tipo $\mathcal{T}_2$, usando a mesma estratégia, usada no passo 1.2 mas para as colunas de caixas do tipo $\mathcal{T}_2$.
- **1.4** Se falhar nos passos 1.2 e 1.3, comece um novo empacotamento $\mathcal{P}_{i+1}$ com colunas vazias nas posições de $p_1 \cup p_2$, em um contêiner B_{i+1}. Coloque caixas não empacotadas b' do tipo $\mathcal{T}_1$ e b'' do tipo $\mathcal{T}_2$ em duas colunas de tipos $\mathcal{T}_1$ e $\mathcal{T}_2$, respectivamente, em $\mathcal{P}_{i+1}$. Caso necessite, gire previamente as caixas para que estas se encontrem nos respectivos conjuntos dos tipos. Incremente i.

2 Retorne $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_i$.

Fim algoritmo.

Damos agora uma idéia de como funciona o Algoritmo BOX$_{k,\epsilon}$. Considere as listas L_{ijk}, $i, j, k \in \{0, 1\}$, definidas no passo 1 do algoritmo (veja a Figura 6.3).

Usando combinação de conjuntos críticos e algoritmos apropriados para cada sublista, o Algoritmo BOX$_{k,\epsilon}$ fornece uma garantia de volume de pelo menos $\frac{2}{9}$ para os empacotamentos $\mathcal{P}_{ijk}$, exceto talvez no empacotamento da lista L_{111} e em uma das listas L_{ijk} tal que $i+j+k = 2$. Apenas para mostrar a idéia, considere ser L_{110} este conjunto. Prevendo isto, é feito uma combinação das caixas críticas responsáveis pela garantia de volume próxima de $\frac{2}{9}$ com as caixas críticas de L_{111} relacionadas com a garantia de volume próximo de $\frac{1}{8}$. Isto possibilita melhorar ou a garantia de $\frac{2}{9}$ ou a garantia de $\frac{1}{8}$.

Para garantir o volume de $\frac{2}{9}$ nas caixas de L_{101} e L_{011}, usamos a estratégia de combinar sublistas críticas de conjuntos definidos por $\mathcal{A}_k^{xy}$ e $\mathcal{B}_k^{xy}$, visualizando as caixas de L_{101} e L_{011} pelo

plano xy. Isto nos possibilita ter esta garantia em pelo menos um destes conjuntos, digamos L_{101}. Com isto, aplique novamente esta estratégia com os conjuntos L_{011} e L_{110}, desta vez, visualizando as caixas pelo plano zx. E neste caso, novamente conseguimos garantir um volume de $\frac{2}{9}$ para o empacotamento de uma destas listas. A outra lista que ainda contém caixas críticas é o conjunto L_{110} referenciado acima com $i+j+k=2$. As caixas que não são criticas no aspecto volumétrico, são empacotadas normalmente usando algoritmos do tipo NF.

Para empacotar as caixas de L_{111} e L_{110}, usamos algoritmos unidimensionais.

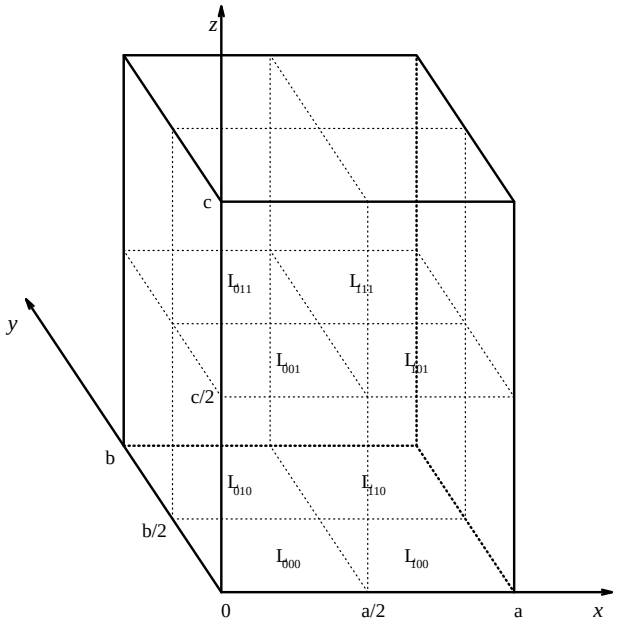


Figura 6.3: Listas L_{ijk}.

Agora podemos apresentar o Algoritmo $\mathrm{BOX}_{k,\epsilon}$.

Algoritmo $\mathrm{BOX}_{k,\epsilon}(L)$

Entrada: Lista de caixas L.

Saída: Empacotamento $\mathcal{P}$ de L em caixas $B=(a,b,c)$, permitindo rotações em torno de qualquer eixo.

1 Sejam

$$\begin{aligned} &\mathcal{X}_0 \leftarrow \mathcal{X}[0,\tfrac{1}{2}] && \mathcal{X}_1 \leftarrow \mathcal{X}[\tfrac{1}{2},1] \\ &\mathcal{Y}_0 \leftarrow \mathcal{Y}[0,\tfrac{1}{2}] && \mathcal{Y}_1 \leftarrow \mathcal{Y}[\tfrac{1}{2},1] \\ &\mathcal{Z}_0 \leftarrow \mathcal{Z}[0,\tfrac{1}{2}] && \mathcal{Z}_1 \leftarrow \mathcal{Z}[\tfrac{1}{2},1] \\ &\mathcal{T}_{ijk} \leftarrow \mathcal{X}_i \cap \mathcal{Y}_j \cap \mathcal{Z}_k, \quad ijk \in \{0,1\}\ ; \end{aligned}$$

2 $p \leftarrow 0{,}4507833184$.

3 Gire todas as caixas possíveis $b \in L \cap \mathcal{T}_{111}$ de forma que b se encontre em um dos conjuntos $\mathcal{T}_{ijk}$, $ijk \neq 111$.

4 $\mathcal{P}_{AB}^{xy} \leftarrow \text{COMBINE-AB}^{xy}(L, \text{rtype}, \mathcal{T}_{011}, \mathcal{T}_{101}, \text{FFC}^{xy})$.

Atualize(L);

Se $\text{rtype}(L, \mathcal{T}_{011} \cap \mathcal{A}_k^{xy}) = \emptyset$ então

$\mathcal{P}_{AB}^{yz} \leftarrow \text{COMBINE-AB}^{yz}(L, \text{rtype}, \mathcal{T}_{101}, \mathcal{T}_{110}, \text{FFC}^{yz})$.
Atualize(L);
$\mathcal{P}_{AB} \leftarrow \mathcal{P}_{AB}^{xy} \cup \mathcal{P}_{AB}^{yz}$.

Caso contrário % todas as caixas do tipo (rtype) $(\mathcal{T}_{101} \cap \mathcal{B}_k^{xy})$ foram empacotados.

$\mathcal{P}_{AB}^{xz} \leftarrow \text{COMBINE-AB}^{xz}(L, \text{rtype}, \mathcal{T}_{110}, \mathcal{T}_{011}, \text{FFC}^{xz})$.
Atualize(L).
$\mathcal{P}_{AB} \leftarrow \mathcal{P}_{AB}^{xy} \cup \mathcal{P}_{AB}^{xz}$.

5 Consideramos a seguir que as caixas dos tipos $(\mathcal{T}_{011} \cap \mathcal{A}_k^{xy})$ e $(\mathcal{T}_{101} \cap \mathcal{A}_k^{yz})$ foram totalmente empacotadas.

5.1 Gire todas as caixas possíveis $b \in L$ do tipo $\mathcal{T}_{110}$ de forma que b se encontre em um dos conjuntos $\mathcal{T}_{ijk}$, $ijk \notin \{111, 110\}$.

5.2 Gire todas as caixas possíveis $b \in L$ do tipo $\mathcal{T}_{110} \cup \mathcal{T}_{111}$ de forma que $z(b)$ seja o menor possível.

5.3 Divida a lista L em 8 sublistas L_{ijk}, $i, j, k \in \{0, 1\}$, tal que $L_{ijk} \leftarrow L \cap \mathcal{X}_i \cap \mathcal{Y}_j \cap \mathcal{Z}_k$. (Veja a Figura 6.3.)

5.4 $\mathcal{P}_{000} \leftarrow \text{H3D}_2(L_{000}, \text{NF})$.

5.5 Gere um empacotamento $\mathcal{P}_{CD}$ da seguinte maneira.

$\mathcal{P}_{CD}^{011} \leftarrow \text{FFC}^{xy}(L_{011} \cap \mathcal{X}[\frac{1}{3}, p], L_{111} \cap \mathcal{X}[\frac{1}{2}, 1-p], [(0,0)], [(0,p)])$;
Atualize(L_{011}, L_{111});
$\mathcal{P}_{CD}^{101} \leftarrow \text{FFC}^{yz}(L_{101} \cap \mathcal{Y}[\frac{1}{3}, p], L_{111} \cap \mathcal{Y}[\frac{1}{2}, 1-p], [(0,0)], [(0,p)])$;
Atualize(L_{101}, L_{111});
$\mathcal{P}_{CD}^{001} \leftarrow \text{FFC}^{xy}(L_{001} \cap \mathcal{X}[\frac{1}{3}, \frac{1}{2}] \cap \mathcal{Y}[\frac{1}{3}, p], L_{111} \cap \mathcal{Y}[\frac{1}{2}, 1-p], [(0,0), (0, \frac{1}{2})], [(0,p)])$;
Atualize(L_{001}, L_{111});
$\mathcal{P}_{CD}^{010} \leftarrow \text{FFC}^{zx}(L_{010} \cap \mathcal{Z}[\frac{1}{3}, \frac{1}{2}] \cap \mathcal{X}[\frac{1}{3}, p], L_{111} \cap \mathcal{X}[\frac{1}{2}, 1-p], [(0,0), (0, \frac{1}{2})], [(0,p)])$;
Atualize(L_{010}, L_{111});
$\mathcal{P}_{CD}^{100} \leftarrow \text{FFC}^{yz}(L_{100} \cap \mathcal{Y}[\frac{1}{3}, \frac{1}{2}] \cap \mathcal{Z}[\frac{1}{3}, p], L_{111} \cap \mathcal{Z}[\frac{1}{2}, 1-p], [(0,0), (0, \frac{1}{2})], [(0,p)])$;
Atualize(L_{100}, L_{111});
$\mathcal{P}_{CD} \leftarrow \mathcal{P}_{CD}^{011} \cup \mathcal{P}_{CD}^{101} \cup \mathcal{P}_{CD}^{001} \cup \mathcal{P}_{CD}^{010} \cup \mathcal{P}_{CD}^{100}$.

5.6 Gere empacotamentos das caixas restantes das listas L_{ijk} tal que $i+j+k=1$.

5.6.1 Gere empacotamento $\mathcal{P}_{001}$ das caixas restantes de L_{001} da seguinte maneira. Seja $L^{18}_{001},\ldots,L^{25}_{001}$ uma partição de L_{001} tal que (veja a Figura 4.12).

$$L^{18}_{001} \leftarrow L_{001}\cap\mathcal{C}^{xy}\left[\tfrac{1}{3},\tfrac{1}{2}\ ;\ \tfrac{1}{3},\tfrac{1}{2}\right],\quad L^{19}_{001} \leftarrow L_{001}\cap\mathcal{C}^{xy}\left[\tfrac{1}{3},\tfrac{1}{2}\ ;\ \tfrac{1}{4},\tfrac{1}{3}\right],$$
$$L^{20}_{001} \leftarrow L_{001}\cap\mathcal{C}^{xy}\left[\tfrac{1}{3},\tfrac{1}{2}\ ;\ 0,\tfrac{1}{4}\right],\quad L^{21}_{001} \leftarrow L_{001}\cap\mathcal{C}^{xy}\left[\tfrac{1}{4},\tfrac{1}{3}\ ;\ \tfrac{1}{3},\tfrac{1}{2}\right],$$
$$L^{22}_{001} \leftarrow L_{001}\cap\mathcal{C}^{xy}\left[\tfrac{1}{4},\tfrac{1}{3}\ ;\ 0,\tfrac{1}{3}\right],\quad L^{23}_{001} \leftarrow L_{001}\cap\mathcal{C}^{xy}\left[0,\tfrac{1}{4}\ ;\ \tfrac{1}{3},\tfrac{1}{2}\right],$$
$$L^{24}_{001} \leftarrow L_{001}\cap\mathcal{C}^{xy}\left[0,\tfrac{1}{4}\ ;\ \tfrac{1}{4},\tfrac{1}{3}\right],\quad L^{25}_{001} \leftarrow L_{001}\cap\mathcal{C}^{xy}\left[0,\tfrac{1}{4}\ ;\ 0,\tfrac{1}{4}\right].$$

$\mathcal{P}^i_{001} \leftarrow \mathrm{H3D}^z(\mathrm{NFDH}^y, L^i_{001}, \mathrm{NF}),\quad i=18,\ldots,22;$
$\mathcal{P}^i_{001} \leftarrow \mathrm{H3D}^z(\mathrm{NFDH}^x, L^i_{001}, \mathrm{NF}),\quad i=23,24;$
$\mathcal{P}^{25}_{001} \leftarrow \mathrm{H3D}^z(\mathrm{LL}_4, L^{25}_{001}, \mathrm{NF});$
$\mathcal{P}_{001} \leftarrow \mathcal{P}^{18}_{001}\cup\ldots\cup\mathcal{P}^{25}_{001}.$

5.6.2 Gere empacotamento $\mathcal{P}_{010}$ das caixas restantes de L_{010} de forma análoga à feita no passo 5.6.1, gerando os níveis na direção do eixo y.

5.6.3 Gere empacotamento $\mathcal{P}_{100}$ das caixas restantes de L_{100} de forma análoga à feita no passo 5.6.1, gerando os níveis na direção do eixo x.

5.7 Gere empacotamento das caixas restantes de L_{011} e L_{101}.

5.7.1 Gere empacotamento $\mathcal{P}_{011}$ das caixas restantes de L_{011}.
Seja $L^1_{011},\ldots,L^{17}_{011}$ uma partição de L_{011} da seguinte maneira (veja a Figura 4.12).
$L^i_{011} \leftarrow L_{011}\cap\mathcal{Y}[\frac{1}{i+2},\frac{1}{i+1}],\quad i=1,\ldots,16;$
$L^{17}_{011} \leftarrow L_{011}\cap\mathcal{Y}[0,\frac{1}{18}];$
$\mathcal{P}^i_{011} \leftarrow \mathrm{H3D}^{xy}(\mathrm{NFDH}^y, L^i_{011}, \mathrm{NF}),\quad i=1,\ldots,17;$
$\mathcal{P}_{011} \leftarrow \mathcal{P}^1_{011}\cup\ldots\cup\mathcal{P}^{17}_{011}.$

5.7.2 Gere empacotamento $\mathcal{P}_{101}$ das caixas restantes de L_{101} de forma análoga à feita no passo 5.7.1, considerando o plano yz ao invés do plano xy.

5.8 Gere empacotamento das caixas restantes de L_{110} e L_{111}.

5.8.1 $L_{UNI} \leftarrow L_{110}\cup L_{111};$

5.8.2 Considere cada caixa d de L_{UNI} como uma barra de comprimento $z(d)$ e cada caixa B como uma barra de comprimento c.

5.8.3 $\mathcal{P}'_{UNI} \leftarrow \mathrm{FFD}^z(L_{UNI});$

5.8.4 $\mathcal{P}''_{UNI} \leftarrow \mathrm{VL}^z_\epsilon(L_{UNI});$

5.8.5 $\mathcal{P}_{UNI} \leftarrow (\mathcal{P}\in\{\mathcal{P}'_{UNI},\mathcal{P}''_{UNI}\}|\#(\mathcal{P})$ é mínimo).

5.9 $\mathcal{P}_{aux} \leftarrow \mathcal{P}_{AB}\cup\mathcal{P}_{CD}\cup\mathcal{P}_{000}\cup\mathcal{P}_{001}\cup\mathcal{P}_{010}\cup\mathcal{P}_{100}\cup\mathcal{P}_{011}\cup\mathcal{P}_{101};$

5.10 $\mathcal{P} \leftarrow \mathcal{P}_{UNI}\cup\mathcal{P}_{aux}.$

6 Para os demais casos, o algoritmo é análogo ao passo 5, diferindo apenas nos planos e direções sobre os quais o empacotamento é construído.

7 Retorne $\mathcal{P}$.

Fim algoritmo.

Teorema 6.4.1. *Para toda lista de caixas L para o* PET^r*, a seguinte desigualdade é válida* $\text{BOX}(L) \leq \alpha_{k,\epsilon} \cdot \text{OPT}(L) + \beta_{k,\epsilon}$*, onde* $\lim_{k\to\infty,\epsilon\to 0} \alpha_{k,\epsilon} \leq 4{,}8821\ldots$*, e* $\beta_{k,\epsilon}$ *é constante para valores constantes de* k *e* ϵ*.*

Prova. Vamos dividir a demonstração em dois casos conforme o conteúdo da lista M, definido como

$$M := L_{111} \cap \mathcal{X}[\frac{1}{2}, 1-p] \cap \mathcal{Y}[\frac{1}{2}, 1-p] \cap \mathcal{Z}[\frac{1}{2}, 1-p],$$

após o passo 5.5.

Caso 1. ($M \neq \emptyset$) após o passo 5.5.

Pelo Lema 6.3.1 temos que

$$\#(\mathcal{P}_{000}) \leq \frac{1}{8/27}\frac{V(L_{000})}{abc} + 14. \tag{6.9}$$

Para o empacotamento $\mathcal{P}_{AB}$, note que em cada contêiner B, deste empacotamento, a área de fundo ocupada em cada contêiner de L_{AB} é pelo menos $\frac{17}{36}$ do fundo do contêiner, isto em todas as caixas, exceto talvez em $2(k+14)$ contêineres. Como cada caixa de L_{AB} tem altura maior que $\frac{1}{2}$ da dimensão z, temos que

$$\#(\mathcal{P}_{AB}) \leq \frac{1}{17/72}V(L_{AB}) + 2k + 28. \tag{6.10}$$

Para o empacotamento $\mathcal{P}_{CD}$, note que cada contêiner B de $\mathcal{P}^i_{CD}$, $i \in \{011, 101, 001, 010, 100\}$ tem área de fundo pelo menos $\frac{1}{4} + \frac{r_1}{2}$ do fundo do contêiner, exceto talvez no último contêiner de cada empacotamento $\mathcal{P}^i_{CD}$. Considerando também que cada caixa tem altura maior que $\frac{1}{2}$ da dimensão correspondente, temos

$$\#(\mathcal{P}_{CD}) \leq \frac{1}{\frac{1}{8} + \frac{r_1}{4}}V(L_{AB}) + 6. \tag{6.11}$$

Para o empacotamento $\mathcal{P}_{001}$, temos que cada um dos empacotamentos $\mathcal{P}^i_{001}$, tem uma ocupação de área de fundo de pelo menos $\frac{17}{36}$ do fundo do contêiner e cada caixa tem altura pelo menos $\frac{1}{2}$ da dimensão correspondente. Note que as caixas que tinham pouca garantia de área em L^{18}_{001} foram totalmente empacotados em $\mathcal{P}_{CD}$, caso contrário não teríamos $M \neq \emptyset$. Assim, procedendo da mesma forma que anteriormente, temos

$$\#(\mathcal{P}_{001}) \leq \frac{1}{17/72}\frac{V(L'_{001})}{abc} + 8. \tag{6.12}$$

A mesma análise feita para o empacotamento $\mathcal{P}_{001}$ pode ser feita para os empacotamentos $\mathcal{P}_{010}$ e $\mathcal{P}_{100}$. Assim, valem as seguintes desigualdades

$$\#(\mathcal{P}_{010}) \leq \frac{1}{17/72}\frac{V(L'_{010})}{abc} + 8, \tag{6.13}$$

$$\#(\mathcal{P}_{100}) \leq \frac{1}{17/72}\frac{V(L'_{100})}{abc} + 8. \tag{6.14}$$

Agora considere o empacotamento $\mathcal{P}_{011}$. Note que cada um dos empacotamentos $\mathcal{P}^i_{011}$ tem área de fundo de pelo menos p do fundo do contêiner (este mínimo atingido para a lista L^1_{011}) exceto talvez no último contêiner de cada empacotamento $\mathcal{P}^i_{011}$. Assim,

$$\#(\mathcal{P}_{011}) \leq \frac{1}{p/2}\frac{V(L'_{011})}{abc} + 17. \tag{6.15}$$

Da mesma forma, temos para o empacotamento $\mathcal{P}_{101}$ a seguinte desigualdade

$$\#(\mathcal{P}_{101}) \leq \frac{1}{p/2}\frac{V(L'_{101})}{abc} + 17. \tag{6.16}$$

A partir de (6.9)—(6.16) e considerando que $\frac{p}{2} = \min\{\frac{p}{2}, \frac{17}{72}, \frac{1}{8} + \frac{r_1}{4}\}$, temos que

$$\#(\mathcal{P}_{aux}) \leq \frac{1}{p/2}\frac{V(L_{aux})}{abc} + C^k_{aux}. \tag{6.17}$$

Finalmente, considere o empacotamento $\mathcal{P}_{UNI}$ gerado para caixas de L_{110} e L_{111} no passo 5.4. O volume mínimo em cada contêiner B de $\mathcal{P}'_{UNI}$, exceto talvez no último contêiner, é pelo menos $\frac{abc}{8}$. Assim,

$$\#(\mathcal{P}'_{UNI}) \leq \frac{1}{1/8}\frac{V(L_{UNI})}{abc} + 1.$$

Note que após a rotação feita no passo 5.1, não há mais nenhuma caixa de L_{UNI} que possa ser girada de forma que se encontre em alguns dos tipos $\mathcal{T}_{ijk}$, $ijk \notin \{110, 111\}$. Assim, depois do passo 5.2, todas as caixas de L_{UNI} terão a menor altura possível, sem sair de $\mathcal{T}_{110} \cup \mathcal{T}_{111}$. Portanto, ao aplicarmos o Algoritmo VL^z_ϵ no passo 5.8.4, temos

$$\#(\mathcal{P}''_{UNI}) \leq (1+\epsilon)\cdot \mathrm{OPT}(L_{UNI}) + C^\epsilon_{UNI}.$$

Como $\#(\mathcal{P}_{UNI}) \leq \max\{\#(\mathcal{P}'_{UNI}), \#(\mathcal{P}''_{UNI})\}$, temos que

$$\#(\mathcal{P}_{UNI}) \leq \frac{1}{1/8}\frac{V(L_{UNI})}{abc} + 1. \tag{6.18}$$

$$\#(\mathcal{P}_{UNI}) \leq (1+\epsilon)\cdot \mathrm{OPT}(L_{UNI}) + C^\epsilon_{UNI} \tag{6.19}$$

A partir de (6.17)—(6.19) podemos concluir que

$$\#(\mathcal{P}) \leq \alpha'_{k,\epsilon}\cdot \mathrm{OPT}(L) + \beta_{k,\epsilon},$$

onde $\alpha'_{k,\epsilon} \leq \frac{h_1+h_2}{\max\{\frac{1}{1+\epsilon}h_1, \frac{1}{8}h_1+\frac{p}{2}h_2\}}$ e $\beta_{k,\epsilon} = C^k_{aux} + C^\epsilon_{UNI}$.

Caso 2. ($M = \emptyset$) após o passo 5.1.

A análise quando $M = \emptyset$ é análoga.

Pelas análises feitas no caso 1, temos

$$\begin{aligned}
\#(\mathcal{P}_{000}) &\leq \frac{1}{8/27}\frac{V(L_{000})}{abc} + 14, \\
\#(\mathcal{P}_{AB}) &\leq \frac{1}{17/72}\frac{V(L_{AB})}{abc} + 2k + 28, \\
\#(\mathcal{P}_{CD}) &\leq \frac{1}{\frac{1}{8}+\frac{r_1}{4}}\frac{V(L_{CD})}{abc} + 6.
\end{aligned}$$

Neste caso, cada empacotamento $\mathcal{P}^i_{001}$ tem garantia de área de pelo menos $\frac{4}{9}$. Como cada caixa de L'_{001} tem altura maior que $\frac{1}{2}$ da dimensão correspondente a altura, temos uma garantia de volume de pelo menos $\frac{2}{9}$. A mesma análise é feita para os empacotamentos $\mathcal{P}_{010}$ e $\mathcal{P}_{100}$. Assim, temos

$$\begin{aligned}
\#(\mathcal{P}_{001}) &\leq \frac{1}{2/9}\frac{V(L'_{001})}{abc} + 8, \\
\#(\mathcal{P}_{010}) &\leq \frac{1}{2/9}\frac{V(L'_{010})}{abc} + 8, \\
\#(\mathcal{P}_{100}) &\leq \frac{1}{2/9}\frac{V(L'_{100})}{abc} + 8.
\end{aligned}$$

Para os empacotamentos $\mathcal{P}^i_{011}$ temos uma garantia de área de pelo menos r_1. Como a altura das caixas é pelo menos $\frac{1}{2}$ da dimensão correspondente, temos uma garantia de volume de pelo menos $\frac{r_1}{2}$. Análise análoga é feita para o empacotamento $\mathcal{P}_{101}$. Assim,

$$\begin{aligned}
\#(\mathcal{P}_{011}) &\leq \frac{1}{r_1/2}\frac{V(L'_{011})}{abc} + 17, \\
\#(\mathcal{P}_{101}) &\leq \frac{1}{r_1/2}\frac{V(L'_{101})}{abc} + 17.
\end{aligned}$$

Pelas inequações acima, temos

$$\#(\mathcal{P}_{aux}) \leq \frac{1}{r_1/2}V(L_{aux}) + C^k_{aux}.$$

Como todas as caixas de M, $M \subseteq L_{111}$, foram empacotadas, temos que o volume mínimo das caixas de L_{111} é pelo menos $\frac{1-p}{4}$. Assim, considerando os empacotamentos de $\mathcal{P}_{110}$ e $\mathcal{P}_{111}$, temos

$$\begin{aligned}
\#(\mathcal{P}_{UNI}) &\leq \frac{1}{(1-p)/4}V(L_{UNI}) + 1. \\
\#(\mathcal{P}_{UNI}) &\leq (1+\epsilon)\cdot \mathrm{OPT}(L_{UNI}) + C^\epsilon_{UNI}
\end{aligned}$$

Assim, obtemos que

$$\#(\mathcal{P}) \leq \alpha''_{k,\epsilon} \cdot \mathrm{OPT}(L) + \beta_{k,\epsilon},$$

onde $\alpha'_{k,\epsilon} \leq \frac{h_1+h_2}{\max\{\frac{1}{1+\epsilon}h_1, \frac{1-p}{4}h_1+\frac{r_1}{2}h_2\}}$ e $\beta_{k,\epsilon} = C^k_{aux} + C^\epsilon_{UNI}$.

Seja $\alpha_{k,\epsilon} = \max\{\alpha'_{k,\epsilon}, \alpha''_{k,\epsilon}\}$. Como para $k \to \infty$ temos $r_1 \to \frac{4}{9}$, podemos concluir pelos dois casos acima que $\lim_{k\to\infty,\epsilon\to 0} \alpha_{k,\epsilon} \leq 4{,}8821\ldots$. □

6.4.2 Algoritmo OBOX$_p$

Apresentamos nesta seção um algoritmo para o empacotamento *on-line* de caixas em contêineres usando de rotações ortogonais. Este algoritmo tem análise de limite de desempenho assintótico baseada na análise feita para o algoritmo desenvolvido por Coppersmith e Raghavan [12]. De fato, o algoritmo desenvolvido em [12] pode ser facilmente adaptado para o PEC^r, usando da mesma adaptação feita para o Algoritmo OBI. A descrição do algoritmo será um pouco diferente, pois aproveitamos alguns outros algoritmos descritos nesta tese.

Algoritmo OBOX

Entrada: Lista de caixas $L = (b_1, \ldots, b_n)$.

Saída: Empacotamento *on-line* de L em contêineres $R = (a, b, c)$, permitindo rotações ortogonais.

1 Sejam

$$\begin{aligned}
&\mathcal{X}_0 \leftarrow \mathcal{X}[0, \tfrac{1}{2}], && \mathcal{X}_1 \leftarrow \mathcal{X}[\tfrac{1}{2}, 1],\\
&\mathcal{Y}_0 \leftarrow \mathcal{Y}[0, \tfrac{1}{2}], && \mathcal{Y}_1 \leftarrow \mathcal{Y}[\tfrac{1}{2}, 1],\\
&\mathcal{Z}_0 \leftarrow \mathcal{Z}[0, \tfrac{1}{2}], && \mathcal{Z}_1 \leftarrow \mathcal{Z}[\tfrac{1}{2}, 1],\\
&\mathcal{T}_{ijk} \leftarrow \mathcal{X}_i \cap \mathcal{Y}_j \cap \mathcal{Z}_k, \quad ijk \in \{0,1\}.
\end{aligned}$$

2 Gire as caixas possíveis $b \in L \cap \mathcal{T}_{111}$, de forma que b se encontre em um dos conjuntos $\mathcal{T}_{ijk}$, $ijk \neq 111$.

3 $L_{ijk} \leftarrow L \cap \mathcal{T}_{ijk}$, $i, j, k \in \{0, 1\}$.

4 Construa empacotamentos $\mathcal{P}_{ijk}$ $i + j + k = 2$ da seguinte maneira:

4.1 Construa empacotamento $\mathcal{P}_{011}$ da lista L_{011} da seguinte maneira:

4.1.1 Particione a lista L_{011} nas listas L^1_{011} e L^2_{011} da seguinte maneira:

$$\begin{aligned}
L^1_{011} &\leftarrow L_{011} \cap \mathcal{X}[\tfrac{1}{3}, \tfrac{1}{2}],\\
L^2_{011} &\leftarrow L_{011} \cap \mathcal{X}[0, \tfrac{1}{3}].
\end{aligned}$$

4.1.2 Construa empacotamento $\mathcal{P}^i_{011}$ de L^i_{011} usando o Algoritmo NF, considerando cada caixa $b \in L^i_{011}$ como um item unidimensional de comprimento $x(b)$ e cada contêiner B como uma barra de comprimento a.

4.1.3 $\mathcal{P}_{011} \leftarrow \mathcal{P}^1_{011} \| \mathcal{P}^2_{011}$.

4.2 Construa empacotamento $\mathcal{P}_{101}$ e $\mathcal{P}_{110}$ para as listas L_{101} e L_{110} de forma análoga ao construído para o empacotamento $\mathcal{P}_{011}$, mas usando as dimensões y e z, em vez de x, respectivamente.

5 Construa empacotamento $\mathcal{P}_{ijk}$ $i + j + k = 1$ da seguinte maneira:

5.1 Construa empacotamento $\mathcal{P}_{001}$ da lista L_{001} usando o Algoritmo $\mathrm{OFF}_2^{(2)}$, considerando cada caixa $b \in L_{001}$ como um retângulo de dimensões $(x(b), y(b))$ e cada contêiner como uma placa de dimensões (a, b).

5.2 Construa empacotamento $\mathcal{P}_{010}$ e $\mathcal{P}_{100}$ para as listas L_{010} e L_{100} de forma análoga ao construído para o empacotamento $\mathcal{P}_{001}$, mas usando o plano zx e yz em vez de xy, respectivamente.

6 $\mathcal{P}_{000} \leftarrow \mathrm{OFF}_2^{(3)}(L_{000})$;

7 $\mathcal{P} \leftarrow \mathcal{P}_{000} \cup \mathcal{P}_{001}\mathcal{P}_{010} \cup \mathcal{P}_{100} \cup \mathcal{P}_{011} \cup \mathcal{P}_{101} \cup \mathcal{P}_{110} \cup \mathcal{P}_{111}$.

8 Retorne $\mathcal{P}$.

Fim algoritmo.

Novamente, da forma como este algoritmo foi descrito, ele é *off-line*. Mas sua transformação em algoritmo *on-line* é fácil, já que todos os algoritmos usados como subrotinas são também *on-line*. Assim, consideramos que este algoritmo é *on-line*.

O seguinte teorema apresenta um limite de desempenho para o Algoritmo OBOX. A demonstração seguirá o mesmo modelo do feito no Teorema 4.5.6.

Teorema 6.4.2. *Para qualquer instância L do* PEC^r*, temos que*

$$\mathrm{OBOX}(L) \leq 6{,}25 \cdot \mathrm{OPT}(L) + C,$$

onde C é uma constante.

Prova. Primeiramente, note que (pelo passo 1) a lista L_{111} contém caixas tais que duas delas não podem ser empacotados em um mesmo contêiner. Assim,

$$\begin{aligned} \mathrm{OPT}(L) &\geq \mathrm{OPT}(L_{111}) \\ &= \#(\mathcal{P}_{111}). \end{aligned} \tag{6.20}$$

Como cada caixa de L_{111} tem volume de pelo menos $\frac{abc}{8}$, temos

$$\#(\mathcal{P}_{111}) \leq \frac{V(L_{111})}{abc/8}. \tag{6.21}$$

Considere a lista L^i_{011}, $i = 1, 2$. O Algoritmo NF garante uma altura preenchida em cada contêiner, de pelo menos $\frac{2}{3}a$, exceto talvez na última. Como cada caixa de L_{011} tem área de fundo $\frac{bc}{4}$, temos

$$\#(\mathcal{P}_{011}) \leq \frac{V(L_{011})}{abc/6} + 2.$$

Este mesmo raciocínio pode ser feito para os empacotamentos das listas L_{101} e L_{110}. Portanto,

$$\#(\mathcal{P}_{ijk}) \leq \frac{V(L_{ijk})}{abc/6} + 2, \quad \text{para} \quad i + j + k = 2. \tag{6.22}$$

Considere o empacotamento $\mathcal{P}_{001}$. Note que todas as caixas de L_{001} têm altura maior que $\frac{c}{2}$. Como as caixas de L_{001} são empacotadas pelo Algoritmo $\mathrm{OFF}^{(2)}_2$, pelo Lema 4.4.7, temos

$$\#(\mathcal{P}_{001}) \leq \frac{V(L_{001})}{(abc)2/9} + 2.$$

Usando esta mesma análise para o empacotamento das listas L_{010} e L_{100}, temos

$$\#(\mathcal{P}_{ijk}) \leq \frac{V(L_{ijk})}{abc/6} + C_1, \quad \text{para} \quad i + j + k = 1, \tag{6.23}$$

onde C_1 é uma constante.

As caixas de L_{000} têm todas as dimensões menores que $\frac{1}{2}$ da dimensão de cada caixa. Assim, pelo Lema 6.3.4, temos

$$\#(\mathcal{P}_{000}) \leq \frac{S(L_{000})}{abc/6} + C_2, \tag{6.24}$$

onde C_2 é uma constante.

Definindo

$$\begin{aligned} n_1 &:= \#(\mathcal{P}_{111}) \quad \text{e} \\ n_2 &:= \sum_{i,j,k \in \{0,1\} \text{ e } ijk \neq 111} \#(\mathcal{P}_{ijk}) - C, \end{aligned}$$

onde $C = C_1 + C_2 + 6$ temos

$$\frac{V(L \setminus L_{111})}{abc} \geq \frac{1}{6} n_2. \tag{6.25}$$

Da mesma forma que feito antes,

$$\begin{aligned} \mathrm{OPT}(L) &\geq \frac{V(L)}{abc} \\ &\geq \frac{V(L_{111})}{abc} + \frac{V(L \setminus L_{111})}{abc} \\ &\geq \frac{1}{8} n_1 + \frac{1}{6} n_2. \end{aligned}$$

Portanto, podemos obter que

$$\mathrm{OBOX}(L) \leq \alpha \cdot \mathrm{OPT}(L) + C,$$

onde $\alpha = \frac{n_1+n_2}{\max\{n_1, \frac{1}{8}n_1+\frac{1}{6}n_2\}} \leq 6{,}25$. □

6.5 Resumo dos Algoritmos

Algoritmos para o PEC

Algoritmo	**Tipo**	α	β	**Ref.**	**Condição**
SCP_m	*off-line*	$\alpha(\mathrm{SCP}_m)$	$\mathcal{O}(m^3)$	$\star$	Caixas de L com dimensões pequenas
$\mathrm{IOFF}_m^{(3)}$	*on-line*	$\frac{m^4+5m^3+10m^2+7m+2}{m^2(m+1)^2}$	$\mathcal{O}(m^4)$	$\star$	Caixas de L com dimensões pequenas
CUBO_1	*off-line*	3,4653...	4	$\star$	Todas as caixas são cubos.
CUBO_m	*off-line*	$\left(\frac{m+1}{m}\right)^3$	$\mathcal{O}(m)$	$\star$	Caixas de L são cubos pequenos, $m \geq 2$.
LC/CV	*on-line*	$\asymp 4{,}84$	$\mathcal{O}\left(\frac{1}{1-r}\frac{1}{1-s}\right)$	[39, 14]	Caso Geral

Algoritmos para o PEC^r

Algoritmo	**Tipo**	α	β	**Ref.**	**Condição**
$\mathrm{H3D}_m(L, NF)$	*off-line*	$\left(\frac{m+1}{m}\right)^3$	11	$\star$	Caixas de L com dimensões pequenas
$\mathrm{OFF}_m^{(3)}$	*on-line*	$\left(\frac{m+1}{m}\right)^3$	$\mathcal{O}(m^4)$	$\star$	Caixas de L com dimensões pequenas
$\mathrm{BOX}_{k,\epsilon}$	*off-line*	$\asymp 4{,}882\ldots$	$\mathcal{O}\left(\frac{k}{\epsilon}\right)$	$\star$	Caso Geral
OBOX	*on-line*	6,25	$\mathcal{O}(1)$	$\star$	Caso Geral

Capítulo 7

Considerações Finais

Neste capítulo fazemos alguns comentários finais sobre a tese. Mencionamos nossas contribuições, algumas considerações sobre implementação e discutimos questões em aberto e direções de pesquisa futura.

7.1 Contribuições

Nesta tese, apresentamos vários algoritmos de aproximação para problemas de empacotamento, muitos dos quais com limites de desempenho melhores que os conhecidos anteriormente. Desenvolvemos também algoritmos para problemas não tratados na literatura, sob essa abordagem. Os limites de desempenho obtidos estão resumidos nas tabelas do Apêndice A.

Em termos teóricos, contribuímos na introdução de técnicas para o desenvolvimento e análise de algoritmos para problemas de empacotamento onde rotações ortogonais são permitidas. A idéia de combinar conjuntos críticos foi anteriormente usada na dissertação de mestrado de Miyazawa [46], mas um tratamento mais elaborado foi apresentado nesta tese. Esse tipo de técnica, até onde sabemos, não foi utilizado por outros autores. A técnica que utilizamos no caso em que rotações ortogonais são permitidas nos parece muito promissora e esperamos que ela constitua um primeiro passo na busca de melhores algoritmos de aproximação para este tipo de problema. Quanto à idéia de combinar conjuntos críticos, esperamos que seu uso seja refinado e usado em mais casos de problemas de empacotamento. Notamos que as próprias definições de conjuntos críticos podem ser revistas de forma a tentar melhorar os limites de desempenho.

O primeiro algoritmo que desenvolvemos usando a idéia de conjuntos críticos, de uma forma mais elaborada, é o Algoritmo TRI_k (para o problema de empacotamento tridimensional). Este algoritmo está descrito em [48], num volume especial sobre algoritmos de aproximação. O Algoritmo R_k (para o problema de empacotamento tridimensional z-orientado) surgiu de uma adaptação do Algoritmo TRI_k. Este algoritmo e variações deste para casos especiais do mesmo

problema, LS, BS e SS estão descritos em [47].

Alguns resumos ou mini-artigos sobre resultados desta tese estão publicados nos anais dos seguintes congressos e encontros: *XV International Symposium on Mathematical Programming* (1994), *I Encontro em Ciência da Computação da UFMS* (1996), *I Oficina Nacional de Corte e Empacotamento* (1996), *XVI International Symposium on Mathematical Programming* (1997) e *XX Congresso Nacional de Matemática Aplicada e Computacional* (1997) [49].

7.2 Considerações sobre implementação

Na descrição dos algoritmos desta tese, não nos preocupamos em apresentar aspectos relacionados à implementação, eficiência ou outros detalhes. Assim, tecemos a seguir algumas considerações a respeito de alguns itens referentes à implementação, de forma que os algoritmos apresentados aqui também possam ter um bom desempenho na prática.

1. Uma das estratégias de empacotamento mais usada nas subrotinas dos algoritmos aqui apresentados é a estratégia NF (*Next Fit, Next Fit Decreasing,...*). Vimos que o uso de tal estratégia, bastante simples, conduz a bons limites de desempenho assintótico. Mas na prática, a substituição desta estratégia pela estratégia FF (*First Fit*) pode levar a comportamentos muito melhores, sem perda de eficiência e qualidade de aproximação. Uma implementação deste algoritmo com complexidade de tempo $\mathcal{O}(n \log n)$ é discutida em [30].

2. Alguns algoritmos que apresentamos usam a seguinte estratégia: fazem uma ordenação da lista de entrada L, particionam L em sublistas $L_1, \ldots, L_k$, $L = L_1 \| \ldots \| L_k$, de forma que itens de cada sublista L_i tenham espaço (comprimento, área ou volume) total não superior a um certo fator do espaço total do recipiente. Feito isto, é usado um algoritmo, digamos $\mathcal{A}$, que sob estas condições de espaço, garante o empacotamento de cada L_i em apenas um recipiente. Por fim o algoritmo retorna a concatenação dos empacotamentos gerados para cada sublista (veja algoritmos BI_m, $\mathrm{BI}_m^{(t)}$ e CUB_m).

   Uma modificação nesses algoritmos que pode levar a comportamentos melhores pode ser feita da seguinte maneira. Seja $L = (e_1, e_2, \ldots, e_n)$ a lista de entrada na ordem desejada. Seja $L_1 = (e_1, e_2, \ldots, e_i)$ a maior lista que o algoritmo $\mathcal{A}$ pode empacotar em um recipiente. Teste se a lista $L_1 \cup \{e_{i+2}\}$ pode ser empacotada em um recipiente. Caso possa, insira o elemento e_{i+2} na lista L_1. Em seguida faça o mesmo procedimento para o elemento e_{i+3}. Faça isto até o elemento e_n. Seja $\mathcal{P}_1$ o empacotamento de L_1 gerado pelo algoritmo $\mathcal{A}$. Remova de L os elementos da lista L_1, mantendo a mesma ordenação para os itens remanescentes. Em seguida, faça para uma lista L_2 o mesmo procedimento feito para a

lista L_1, gerando um empacotamento $\mathcal{P}_2$. Continue desta maneira, gerando empacotamentos $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_v$, até que todos os itens de L sejam empacotados. Por fim, retorne a concatenação dos empacotamentos gerados.

3. Algoritmos que combinam itens, de digamos listas L_A e L_B, podem gerar espaços vagos, dentro do empacotamento combinado, que nunca vão ser usados. Um exemplo disto ocorre na Figura 4.13. Assim, recomendamos que estes espaços sejam aproveitados com itens pequenos, ou que se faça uma redefinição dos itens críticos, deste empacotamento combinado, de forma a aproveitar melhor o espaço em cada recipiente.

4. Caso se ache a implementação do Algoritmo VL_ϵ muito complexa, ou inviável para as condições de uso, sugerimos trocá-lo pelo Algoritmo MFFD (*Modified First Fit Decreasing*), apresentado por Johnson e Garey [34], que tem limite de desempenho assintótico $\frac{71}{60} = 1{,}18333\ldots$ (ou por algum outro algoritmo com bom limite de desempenho). Observamos que esta mudança aumenta o limite de desempenho assintótico dos algoritmos alterados, embora este aumento não seja grande, mas na prática esta mudança pode ser muito mais proveitosa, já que estes algoritmos são bem simples de serem implementados e têm bom comportamento esperado.

5. Como já mencionamos, muitas aplicações práticas são vistas como problemas de corte, em vez de problemas de empacotamento. Assim, em vez de um item ser empacotado em um recipiente, este é cortado de um material (recipiente). Mais ainda, estes cortes muitas vezes devem ser do tipo guilhotina. Salientamos que todos os algoritmos desta tese, exceto os algoritmos que usam o Algoritmo UD (do PEF) como subrotina, geram empacotamentos guilhotináveis. Assim, dos algoritmos propostos, apenas os algoritmos TRI_k e R_k não geram empacotamentos guilhotináveis. Note que uma das principais aplicações destes problemas é o escalonamento de processos em computadores paralelos. Nesta aplicação, não importa se o empacotamento é ou não guilhotinável.

7.3 Questões em Aberto e Pesquisas Futuras

> "It is typical of this business that one answer leads to n more questions."
>
> Ronald L. Graham [27]

Nesta seção, apresentamos alguns problemas em aberto e suspeitas que obtivemos, e que podem conduzir a futuras pesquisas.

1. Eeste trabalho é centrado no desenvolvimento de algoritmos de aproximação para problemas de empacotamento. Aperfeiçoando-se algumas das técnicas aqui introduzidas, talvez

seja possível obter algoritmos com limites de desempenho melhores dos que os apresentados nesta tese. Utilizar essas técnicas em outros problemas também nos parece um tópico promissor para futuros trabalhos.

2. Nas *Preliminares* mencionamos que um dos problemas mais simples envolvendo rotações ortogonais é quando temos apenas um tipo de item sendo empacotado (*Pallet Loading Problem*). Observamos que algumas pesquisas sobre este problema têm sido feitas, muitas delas a respeito de heurísticas. Ainda que este problema venha sendo tratado como sendo $\mathcal{NP}$-difícil, não encontramos nenhum artigo da literatura mostrando a complexidade computacional do problema. Assim, deixamos esta questão como uma pergunta em aberto.

3. Li e Cheng [44] apresentaram um algoritmo *on-line* para o PET, com um limite de desempenho assintótico 2,89. Este algoritmo é baseado na estratégia FF (de limite de desempenho assintótico 1,7) de forma que o algoritmo de empacotamento tridimensional se dividisse no uso de duas aplicações do Algoritmo FF juntamente com a técnica de arredondamento em potências de p, $0 < p < 1$. O valor 2,89 reflete bem estas duas aplicações do Algoritmo FF, sendo igual a $(1{,}7)^2$. Neste mesmo artigo, esses autores questionam a existência de algum algoritmo polinomial para este problema com limite de desempenho melhor. Nesta tese, respondemos a esta questão apresentando um algoritmo com um limite de desempenho assintótico de 2,67. Nosso algoritmo, entretanto, é *off-line*. Uma pergunta natural que surge é se é possível usar o Algoritmo H_M (de limite de desempenho assintótico 1,6910...) da mesma forma como feito para o Algoritmo FF e obter um limite de desempenho melhor para o caso *on-line*.

4. Uma questão em aberto, já há algum tempo, é sobre a existência de um algoritmo para o PEU que não usa mais que um número constante de barras que o ótimo [7, 8]. Ou seja, existe algoritmo $\mathcal{A}$ para o PEU e constante C tal que $\mathcal{A}(L) \leq \mathrm{OPT}(L) + C$?

5. Uma linha que pode gerar boas pesquisas é a inserção de uma limitação no número de recipientes sendo empacotados na memória por algoritmos *on-line* (*bounded-space*). Ou seja, para empacotar um item, o algoritmo deve considerar um máximo de k recipientes, digamos chamados ativos, e uma vez que este deixa de ser ativo, nunca mais volta a ser considerado. Um exemplo disto é o Algoritmo NF para o PEU, com $k = 1$. Neste algoritmo, apenas a barra sendo empacotada é considerada (ativa). Quando um item não pode ser empacotado nesta barra, esta deixa de ser ativa e o item é empacotado em uma nova barra, que se torna a barra ativa. Este tipo de abordagem é muito usada para o problema de empacotamento unidimensional, mas pouco usado para problemas de dimensões maiores.

6. A permissão de rotações ortogonais abre um novo leque de possíveis algoritmos (de aproximação) que podem ser obtidos. Podemos considerar todos os problemas vistos neste trabalho sob este enfoque, gerando novas e interessantes variantes. Algumas abordagens

que podem ser consideradas juntamente com a possibilidade de rotações ortogonais são: algoritmos *on-line*, algoritmos *bounded-space*, empacotamentos guilhotináveis, empacotamentos em níveis, análises de caso médio, algoritmos exatos, etc.

7. O Algoritmo HFF para o PEP (com um limite de desempenho assintótico 2,125) é construído usando-se uma aplicação do Algoritmo FFDH$^{(f)}$ para o PEF (com limite de desempenho assintótico $\frac{17}{10}$) e uma aplicação do Algoritmo FFD para o PEU (com limite de desempenho assintótico $\frac{11}{9}$). Os autores [6] questionam se o limite de desempenho assintótico para o Algoritmo HFF pode ser melhorado para $\frac{17}{10}\frac{11}{9} = 2{,}0777\ldots$.

8. É possível formular o PEU como um problema de programação linear inteira da forma: minimizar $x \cdot \mathbf{1}$ sujeito a $A \cdot x = b$. Nesta formulação, A é uma matriz cujas linhas são indexadas pelos itens de L e as colunas representam as formas possíveis de se empacotar itens dentro de uma barra. Assim, cada coluna representa um possível *padrão de empacotamento* em uma barra. O vetor $b \geq \mathbf{0}$ representa a quantidade de itens de cada tipo necessários no empacotamento. E finalmente, o vetor $x \geq \mathbf{0}$ deve ser um vetor inteiro, de forma a representar uma solução para o empacotamento (*i.e.*, a quantidade de cada padrão usado nesta solução). Assim, claramente, o objetivo deste sistema é minimizar o número de barras usadas para empacotar L. O seguinte comportamento tem sido observado: se removermos a restrição de integralidade do vetor x, e resolvermos o problema relaxado, o valor da solução (fracionária) arredondada para cima é menor ou igual ao valor ótimo do empacotamento mais um. Isto é sempre verdade ? [53]

9. Dados k retângulos distintos $r_1, \ldots, r_k$ e multiplicidades n_i para cada retângulo r_i e $n = \sum_{i=1}^{k} n_i$, onde k é constante, pergunta-se: é possível empacotar estes n retângulos em uma placa de tamanho unitário ? Chamaremos este problema de k-*Rectangle Packing Problem*. Schiermeyer [54] propõe duas conjecturas:

   - Conjectura Fraca: O 2-*Rectangle Packing Problem* pode ser resolvido em tempo polinomial.
   - Conjectura Forte: O k-*Rectangle Packing Problem* pode ser resolvido em tempo polinomial para cada k fixo.

10. Os limites de inaproximabilidade para os limites de desempenho assintóticos dos algoritmos *on-line* para os problemas de empacotamento bi- e tridimensional (PEP, PET e PEC) ainda estão muito distantes dos valores obtidos até o momento. Assim, outra linha que pode gerar boas pesquisas é a obtenção de melhores limites para esses problemas, inclusive considerando rotações ortogonais.

11. Quando consideramos problemas onde rotações ortogonais são permitidas, podemos permitir rotações em torno de todos os eixos ou também podemos fixar algum eixo. No problema PET^r, permitimos rotações em torno do eixo z. Uma das aplicações interessantes para este problema é em escalonamento de processos em computadores paralelos. Já no problema PEC^r permitimos rotações em torno de qualquer eixo. Assim, uma extensão natural é considerar qualquer tipo de rotação para o PET^r ou restringirmos as rotações em torno de apenas um eixo para o PEC^r.

12. Em todos os problemas de empacotamento que consideramos, onde o objetivo é o de minimizar o número de recipientes usado no empacotamento, os recipientes têm as mesmas dimensões. Uma extensão deste problema é considerar empacotamentos em recipientes não necessariamente iguais.

Tabelas com Limites de Desempenho Assintótico

Nas tabelas apresentadas a seguir indicamos os algoritmos com os melhores limites de desempenho conhecidos para os problemas de empacotamento estudados nesta tese. Para cada algoritmo $\mathcal{A}$, os valores de α e β apresentados são relativos aos limites de desempenho assintótico desse algoritmo. Mais precisamente, são as constantes de $\mathcal{A}(L) \leq \alpha \cdot OPT(L) + \beta$. Denotamos por Z a altura do maior item, quando o termo couber.

Apresentamos a referência onde está apresentado cada algoritmo, sendo que os algoritmos que desenvolvemos aqui estão marcados com $\star$. Para esses últimos algoritmos que desenvolvemos, acrescentamos um campo indicando o melhor limite de desempenho assintótico anteriormente conhecido, caso exista. Para os algoritmos que foram desenvolvidos para o caso orientado, cujo limite de desempenho também é válido para o caso com rotações, colocamos o problema relacionado entre parênteses. Alguns dos algoritmos estão batizados com as iniciais dos seus respectivos autores.

Algoritmos para o Problema de Empacotamento Unidimensional

Algoritmo	**Tipo**	α	β	**Ref.**	**Condição**
FFD	*off-line*	$\frac{m+3}{m+2} - \frac{1}{m(m+1)(m+2)}$	4	[13]	Itens pequenos
H_{m+2}	*on-line*	$\alpha(\mathrm{H}_{m+2})$	3	[37]	Itens pequenos
FFD,BFD	*off-line*	1,5	—	[55]	Caso Geral
VL_ϵ	*off-line*	$1+\epsilon$	$\left(\frac{1}{\epsilon}\right)^2$	[16]	Caso Geral
KK	*off-line*	1	$\mathcal{O}\left(\frac{log^2 \mathrm{OPT}(L)}{\mathrm{OPT}(L)}\right)$	[35]	Caso Geral
Harmonic + 1	*on-line*	$\asymp 1{,}5888\ldots$	$\mathcal{O}(M)$	[52]	Caso Geral

Algoritmos para o Problema de Empacotamento em Faixa

Algoritmo	Problema	Tipo	α	β	Ref.	Condição
M,S	PEF	*off-line*	2	—–	[54, 57]	Caso Geral
UD	PEF	*off-line*	1,25	$\frac{53}{8}Z$	[2]	Caso Geral
SF	PEF	*off-line*	$\frac{m+2}{m+1}$	$2Z$	[9]	Retângulos de largura pequena
IOSSP$_{m,p}$	PEF	*on-line*	$\asymp \frac{m+2}{m+1} + \frac{1}{(m+1)^2}$	$\frac{4Z}{1-p}$	$\star$	Retângulos de largura pequena
Shelf(H_M, p)	PEF	*on-line*	$\asymp$ 1,691	$\frac{M}{1-p}$	[15]	Caso Geral
SSP$_m$	PEF^r	*off-line*	$\frac{m+1}{m}$	$2Z$	$\star$	Retângulos de largura pequena
OSSP$_{m,p}$	PEF^r	*on-line*	$\asymp \frac{m+1}{m}$	$\frac{2Z}{1-p}$	$\star$	Retângulos de largura pequena
SPR	PEF^r	*off-line*	1,6123 ...	$4Z$	$\star$	Caso Geral
OSPR	PEF^r	*on-line*	$\asymp$ 1,75 ...	$\frac{2Z}{1-p}$	$\star$	Caso Geral

Algoritmos para o Problema de Empacotamento em Placas

Algoritmo	Problema	Tipo	α	β	Ref.	Condição
HFF	PEP	*off-line*	2,125	5	[6]	Caso Geral
STP$_m$	PEP	*off-line*	α(STP$_m$)	$\mathcal{O}(m)$	$\star$	Retângulos de L com dimensões pequenas
IOFF$_m$	PEP	*on-line*	$\left(\frac{m+2}{m+1}\right)^2 + \frac{2}{m(m+1)}$	$\mathcal{O}(m^2)$	$\star$	Retângulos de L com dimensões pequenas
FF$^{(2)}$	PEP	*on-line*	$\asymp$ 2,86	$\mathcal{O}\left(\frac{1}{1-r}\frac{1}{1-s}\right)$	[39, 14]	Caso Geral
SS$_m$	PEP	*off-line*	α(SS$_m$)	$4Z$	$\star$	Empacotamento de quadrados pequenos em quadrados
BI$_m$	PEP^r	*off-line*	$\left(\frac{m+1}{m}\right)^2$	6	$\star$	Retângulos de L com dimensões pequenas
OFF$_m^{(2)}$	PEP^r	*on-line*	$\left(\frac{m+1}{m}\right)^2$	$\mathcal{O}(m^2)$	$\star$	Retângulos de L com dimensões pequenas
BI$_{k,\epsilon}$	PEP^r	*off-line*	$\asymp$ 2,639 ...	$\mathcal{O}\left(k + \frac{1}{\epsilon}\right)$	$\star$	Caso Geral
RR$_k$	PEP^r	*on-line*	$\asymp$ 2,6875	$\mathcal{O}(k)$	$\star$	Empacotamento em Placas quadradas
OBI	PEP^r	*on-line*	3,25	10	$\star$	Caso Geral

Algoritmos para o Problema de Empacotamento Tridimensional

Algoritmo	Problema	Tipo	α	β	Ref.	Condição	Anterior
TRI_k	PET	*off-line*	$\asymp 2{,}6607\ldots$	$\left(2k+\frac{597}{8}\right)Z$	$\star$	Caso Geral	$\asymp 2{,}89$ [44]
LS	PET	*off-line*	2,5425	$\frac{101}{8}Z$	$\star$	Caixas de L têm fundo quadrado	—
$\mathrm{STP}_m^{(t)}$	PET	*off-line*	$\alpha(\mathrm{STP}_m^{(t)})$	$\mathcal{O}(m \cdot Z)$	$\star$	Todas as caixas têm fundo pequeno	$\left(\frac{m+1}{m-1}\right)$ [41]
$\mathrm{FFLS}_{r,s}$	PET	*on-line*	$\asymp 2{,}89$	$\mathcal{O}\left(\frac{1}{(1-r)}\frac{1}{(1-s)}\right)Z$	[44]	Caso Geral	3,25 [12]
$\mathrm{ISRR}_{m,p}$	PET	*on-line*	$\asymp \left(\frac{m+2}{m+1}\right)^2+\frac{2}{m(m+1)}$	$\mathcal{O}\left(\frac{m^2}{1-p}\right)Z$	$\star$	Todas as caixas têm fundo pequeno	—
$\mathrm{SS}_1^{(t)}$	PET	*off-line*	$2{,}3605\ldots$	$4Z$	$\star$	Todas as caixas têm fundo quadrado	2,6875 [40]
$\mathrm{SS}_m^{(t)}$	PET	*off-line*	$\alpha(\mathrm{SS}_m^{(t)})$	$4Z$	$\star$	Todas as caixas têm fundo quadrado e pequeno	—
$\mathrm{BI}_m^{(t)}$	PET^r	*off-line*	$\left(\frac{m+1}{m}\right)^2$	$6Z$	[46]	Todas as caixas têm fundo pequeno	$\left(\frac{m+1}{m-1}\right)$ [41]
$\mathrm{SRR}_{m,p}^{(3)}$	PET^r	*on-line*	$\left(\frac{m+1}{m}\right)^2$	$\mathcal{O}\left(\frac{m^2}{1-p}\right)Z$	$\star$	Todas as caixas têm fundo pequeno	—
R_k	PET^r	*off-line*	$\asymp 2{,}6607\ldots$	$\left(2k+\frac{597}{8}\right)Z$	$\star$	Caso Geral	3,0491 [46]
BS	PET^r	*off-line*	$2{,}5273\ldots$	$15Z$	$\star$	Caixa B tem fundo quadrado	—
$\mathrm{RR}_{k,p}^{(3)}$	PET^r	*on-line*	$\asymp 2{,}6875$	$\mathcal{O}\left(\frac{k}{1-r}\right)Z$	$\star$	Caixa B tem fundo quadrado	—
OTRI_p	PET^r	*on-line*	$\asymp 3{,}25$	$\mathcal{O}\left(\frac{10}{1-p}\right)Z$	$\star$	Caso Geral	—

Algoritmos para o Problema de Empacotamento em Contêiners

Algoritmo	Problema	Tipo	α	β	Ref.	Condição
Li-Cheng/CsirikV	PEC	*on-line*	$\asymp 4{,}84$	$\mathcal{O}\left(\frac{1}{1-r}\frac{1}{1-s}\right)$	[39, 14]	Caso Geral
SCP_m	PEC	*off-line*	$\alpha(\mathrm{SCP}_m)$	$\mathcal{O}(m^3)$	$\star$	Caixas com dimensões pequenas
$\mathrm{IOFF}_m^{(3)}$	PEC	*on-line*	$\frac{m^4+5m^3+10m^2+7m+2}{m^2(m+1)^2}$	$\mathcal{O}(m^4)$	$\star$	Caixas de L com dimensões pequenas
CUBO_1	PEC	*off-line*	$3{,}4653\ldots$	4	$\star$	Todas as caixas são cubos.
CUBO_m	PEC	*off-line*	$\alpha(\mathrm{CUBO}_m)$	$\mathcal{O}(m)$	$\star$	Caixas de L são cubos pequenos, $m \geq 2$.
$\mathrm{H3D}_m(L, NF)$	PEC^r	*off-line*	$\left(\frac{m+1}{m}\right)^3$	11	$\star$	Caixas com dimensões pequenas
$\mathrm{OFF}_m^{(3)}$	PEC^r	*on-line*	$\left(\frac{m+1}{m}\right)^3$	$\mathcal{O}(m^4)$	$\star$	Caixas de L com dimensões pequenas
$\mathrm{BOX}_{k,\epsilon}$	PEC^r	*off-line*	$\asymp 4{,}882\ldots$	$\mathcal{O}\left(\frac{k}{\epsilon}\right)$	$\star$	Caso Geral
OBOX	PEC^r	*on-line*	6,25	$\mathcal{O}(1)$	$\star$	Caso Geral

Algoritmos para Empacotamento de itens Pequenos

Algoritmo	**Problema**	**Tipo**	**Ref.**	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$	$m=7$	$m=8$	$m=9$	$m=10$
FFD	PEU	*off-line*	[13]	1,334	1,250	1,200	1,167	1,143	1,125	1,112	1,100	1,091	1,084
H_M	PEU	*on-line*	[37, 21]	1,691	1,423	1,302	1,233	1,192					
SF	PEF	*off-line*	[9]	1,500	1,333	1,250	1,200	1,167	1,143	1,125	1,111	1,100	1,091
$\mathrm{IOSSP}_{m,p}$	PEF	*on-line*	$\star$	1,750	1,445	1,313	1,240	1,195	1,164	1,141	1,124	1,110	1,100
$\mathrm{SSP}_{m,p}$	PEF^r	*off-line*	$\star$	2,000	1,500	1,334	1,250	1,200	1,167	1,143	1,125	1,112	1,100
$\mathrm{OSSP}_{m,p}$	PEF^r	*on-line*	$\star$	2,000	1,500	1,334	1,250	1,200	1,167	1,143	1,125	1,112	1,100
STP_m	PEP	*off-line*	$\star$	3,049	2,027	1,683	1,511	1,408	1,339	1,291	1,254	1,226	1,203
IOFF_m	PEP	*on-line*	$\star$	3,250	2,112	1,730	1,540	1,428	1,354	1,302	1,263	1,233	1,209
SS_m	PEP	*off-line*	$\star$	2,361	1,836	1,598	1,464	1,378	1,319	1,276	1,243	1,217	1,196
BI_m	PEP^r	*off-line*	$\star$	4,000	2,250	1,778	1,563	1,441	1,362	1,307	1,266	1,235	1,211
$\mathrm{OFF}_m^{(2)}$	PEP^r	*on-line*	$\star$	4,000	2,250	1,778	1,563	1,441	1,362	1,307	1,266	1,235	1,211
$\mathrm{STP}_m^{(t)}$	PET	*off-line*	$\star$	3,049	2,027	1,683	1,511	1,408	1,339	1,291	1,254	1,226	1,203
$\mathrm{ISRR}_{m,p}$	PET	*on-line*	$\star$	3,250	2,112	1,730	1,540	1,428	1,354	1,302	1,263	1,233	1,209
$\mathrm{SS}_m^{(t)}$	PET	*off-line*	$\star$	2,361	1,836	1,598	1,464	1,378	1,319	1,276	1,243	1,217	1,196
$\mathrm{BI}_m^{(t)}$	PET^r	*off-line*	[46]	4,000	2,250	1,778	1,563	1,441	1,362	1,307	1,266	1,235	1,211
$\mathrm{SRR}_{m,p}^{(3)}$	PET^r	*on-line*	$\star$	4,000	2,250	1,778	1,563	1,441	1,362	1,307	1,266	1,235	1,211
$\mathrm{IOFF}_m^{(3)}$	PEC	*on-line*	$\star$	6,250	3,112	2,285	1,915	1,708	1,576	1,486	1,419	1,369	1,329
SCP_m	PEC	*off-line*	$\star$	6,023	3,016	2,233	1,882	1,685	1,560	1,473	1,409	1,361	1,322
CUBO_m	PEC	*off-line*	$\star$	3,466	2,424	1,988	1,752	1,605	1,506	1,434	1,380	1,338	1,304
$\mathrm{H3D}_m(L, \mathrm{NF})$	PEC^r	*off-line*	$\star$	8,000	3,376	2,371	1,954	1,729	1,589	1,493	1,424	1,372	1,332
$\mathrm{OFF}_m^{(3)}$	PEC^r	*on-line*	$\star$	8,000	3,376	2,371	1,954	1,729	1,589	1,493	1,424	1,372	1,332

Aplicações

Neste apêndice listamos algumas aplicações para os problemas de empacotamentos abordados nesta tese. As aplicações onde as versões com rotações ortogonais podem ser usadas estão marcadas com o símbolo $^{(r)}$.

Problema de Empacotamento Unidimensional

- *Alocação de comerciais em TV.*

  Tem-se n comerciais para televisão $p_1, p_2, \ldots, p_n$, tendo cada comercial p_i uma duração de s_i segundos. Esses comerciais devem ser apresentados em intervalos de um programa de televisão, onde cada intervalo possui C segundos. Deseja-se alocar esses comerciais de modo a diminuir o número de intervalos necessários para apresentar esses comerciais.

- *Alocação de programas em discos e fitas magnéticas.*

  Programas de computador de tamanhos $p_1, p_2, \ldots, p_n$, cada programa p_i com tamanho s_i bytes, devem ser colocados em trilhas (ou setores) de discos magnéticos, de forma a usar o menor número de trilhas para gravar os n programas.

- *Carregamento de veículos.*

  Carregar n cargas com peso s_i $(i = 1, \ldots, n)$, em veículos com limite de peso C, de maneira a não violar a restrição de lotação de cada veículo, com o objetivo de minimizar o número de veículos necessários.

- *Escalonamento de tarefas.*

  Minimizar o número de máquinas necessárias para completar n tarefas $p_1, p_2, \ldots, p_n$, em um dado limite de tempo C. Sabe-se que cada tarefa p_i requer um tempo s_i para ser executada.

- *Problema da programação de veículos.*

  Cargas (por ex., jornais) devem ser transportadas por veículos, a partir de um depósito até os pontos de entrega, em no máximo C horas. Cada veículo pode fazer várias viagens. Programar n viagens com tempo de duração s_i horas ($i = 1, \ldots, n$), de maneira a não atrasar a entrega das cargas e com o objetivo de minimizar o número de veículos necessários.

- *Corte de bobinas (papel, aço, ...).*

  Bobinas (por ex., papel), de comprimento C são produzidas por uma fábrica e devem ser cortadas em diversos rolos de comprimentos $s_1, \ldots, s_n$, de forma a minimizar o número de bobinas necessárias.

- *Corte de vigas (em madeireiras, construção civil, ...).*

  Barras (por ex., ferro), de comprimento C, devem ser cortados em diversas barras menores de comprimentos $s_1, \ldots, s_n$. O objetivo é cortar o menor número possível de barras de comprimento C.

- *Pré-paginação.*

  Frações de páginas de memória (de computador) de tamanhos $s_1, \ldots, s_n$ devem ser alocadas em páginas de tamanho C bytes (frações de página são requeridas por segmentos de programas e estes devem aparecer em um menor número possível de páginas, por ex., *loops* internos, *arrays*). Encontrar um alocação que minimize o número de páginas de tamanho C.

Problema de Empacotamento em Placas

- *Corte de placas (vidro$^{(r)}$, chapas$^{(r)}$, madeira, compensado$^{(r)}$, ...).*

  Placas R de tamanho (a, b) devem ser cortadas em placas menores de vários tamanhos $(r_1, \ldots, r_n)$. O objetivo é minimizar o número de placas R necessárias para cortar as placas menores.

Problema de Empacotamento em Faixa

- *Corte de retalhos em fábrica de tecidos, ou em confecção de roupas$^{(r)}$ (rotações permitidas para os retalhos sem orientação devida à estampa, nem às linhas do tecido).*

  Retalhos de tecidos retangulares $r_1, \ldots, r_n$ devem ser cortados em um rolo de tecido de largura a, objetivando-se minimizar o comprimento do rolo de tecido a ser cortado.

- *Corte de películas de filme fotográfico*$^{(r)}$.

  Uma película de filme de largura a deve ser cortada em n fotos de tamanhos retangulares, objetivando-se minimizar o comprimento da película de filme usada.

- *Escalonamento de tarefas.*

  Um conjunto de n programas $r_1, \ldots, r_n$ devem ser executados dispondo se de uma quantidade de recurso a (por exemplo memória de computador). Cada programa $r_i = (w_i, h_i)$ necessita de w_i unidades do recurso e leva h_i unidades de tempo para ser executado. O objetivo é minimizar o tempo necessário para executar os n programas.

Problema de Empacotamento Tridimensional

- *Empacotamento de caixas em galpões*$^{(r)}$.

  Caixas $c_1, \ldots, c_n$ devem ser armazenadas em um galpão com fundo $a \times b$, de forma a minimizar a altura da disposição final das caixas.

- *Escalonamento de tarefas em sistemas particionáveis de malha conexa (job scheduling in partitionable mesh connected systems)* $^{(r)}$.

  Um computador paralelo com uma configuração que forma uma malha de processadores de forma retangular (em geral quadrada) deve executar n processos $c_1, \ldots, c_n$, onde cada processo $c_i = (x_i, y_i, z_i)$ usa uma submalha de tamanho (x_i, y_i) gastando z_i unidades de tempo. O objetivo aqui é alocar os processos de modo a minimizar o tempo total para que o computador execute os n processos.

Problema de Empacotamento em Contêineres

- *Empacotamento em contêineres*$^{(r)}$.

  Caixas $c_1, \ldots, c_n$ devem ser empacotadas em contêiners de tamanho (a, b, c). O objetivo é fazer o empacotamento de modo a minimizar o número de contêiners usados.

- *Carregamento de cargas em furgões*$^{(r)}$.

  Carregar cargas $c_1, \ldots, c_n$ em veículos de transporte com dimensões de (a, b, c). O objetivo aqui é minimizar o número de veículos necessários para transportar as n cargas.

- *Corte de espumas para colchões*$^{(r)}$.

  Espumas de colchões devem ser cortadas de blocos de espumas maiores. O objetivo é minimizar o número destes blocos de espuma usados.

Referências Bibliográficas

[1] B. S. Baker, *A new proof for the first-fit decreasing bin-packing algorithm*, J. of Algorithms, 6 (1985), pp. 49–70.

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

[3] B. S. Baker, E. G. Coffman Jr., and R. L. Rivest, *Orthogonal packings in two-dimensions*, SIAM J. Comput., 9 (1980), pp. 846–855.

[4] D. Blitz, A. van Vliet, and G. J. Woeginger, *Lower bounds on the asymptotic worst-case ratio of online bin packing algorithms*, Unpublished manuscript, (1996).

[5] D. J. Brown, B. S. Baker, and H. P. Katseff, *Lower bounds for the on-line two dimensional packing algorithms*, Acta Informatica, 18 (1982), pp. 207–225.

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

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

[8] ——, *Approximation algorithms (ed. D. Hochbaum)*, PWS, 1997, ch. Approximation algorithms for bin packing - a survey.

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

[10] E. G. Coffman, Jr. and P. W. Shor, *Average-case analysis of cutting and packing in two dimensions*, European J. Operational Research, 44 (1990), pp. 134–144.

[11] S. Cook, *The complexity of theorem-proving procedures*, in Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, 1971, pp. 151–158.

[12] D. Coppersmith and P. Raghavan, *Multidimensional on-line bin packing: algorithms and worst-case analysis*, Oper. Res. Lett., 8 (1989), pp. 17–20.

[13] J. Csirik, *The parametric behavior of the first-fit decreasing bin packing algorithm*, J. of Algorithms, 15 (1993), pp. 1–28.

[14] J. Csirik and A. van Vliet, *An on-line algorithm for multidimensional bin packing*, Operations Research Letters, 13 (1993), pp. 149–158.

[15] J. Csirik and G. J. Woeginger, *Shelf algorithms for on-line strip packing*, Optimierung und Kontrolle. Bericht Nr. 79, Karl-Franzens-Universität Graz & Technische Universität Graz, Juli 1996.

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

[17] K. A. Dowsland, *An exact algorithm for the pallet loading problem*, European Journal of Operational Research, 31 (1987), pp. 78–84.

[18] K. A. Dowsland, *Efficient automated pallet loading*, European J. Operational Research, 44 (1990), pp. 232–238.

[19] K. A. Dowsland and W. B. Dowsland, *Packing problems*, European J. Operational Research, 56 (1992), pp. 2–14.

[20] J. B. Frenk and G. Galambos, *Hybrid next fit algorithm for the two-dimensional rectangle bin packing problem*, Computing, 39 (1987), pp. 201–217.

[21] G. Galambos, *Parametric lower bound for on-line bin-packing*, SIAM J. Algebraic Discrete Methods, 7 (1986), pp. 362–367.

[22] M. R. Garey, R. L. Graham, D. S. Johnson, and A. C. Yao, *Resource constrained scheduling as generalized bin packing*, J. Combinatorial Theory Ser. A, 21 (1976), pp. 257–298.

[23] M. R. Garey, R. L. Graham, and J. D. Ullman, *Worst-case analysis of memory allocation algorithms*, in Proc. 4th Annual ACM Symp. on the Theory of Computing, 1972, pp. 143–150.

[24] M. R. Garey and D. S. Johnson, *Complexity results for multiprocessor scheduling under resource constraints*, SIAM J. Comput., 4 (1975), pp. 397–411.

[25] ——, *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness*, Freeman, San Francisco, 1979.

[26] R. L. Graham, *Bounds for certain multiprocessor anomalies*, Bell System Technical Journal, 45 (1966), pp. 1563–1581.

[27] ——, *The Mathematical Gardner*, Wadsworth International, 1981, ch. Fault-free Tilings of Rectangles, pp. 120–126.

[28] D. Hochbaum, ed., *Approximation Algorithms for $\mathcal{NP}$-hard problems*, PWS, 1997.

[29] Hoffman, *The Mathematical Gardner*, Wadsworth International, 1981, ch. Packing problems and Inequalities, pp. 212–225.

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

[31] ——, *Fast algorithms for bin packing*, J. Comput. Syst. Sci., 8 (1974), pp. 272–314.

[32] ——, *The NP-completeness columm: an ongoing guide; The tale of the second prover*, J. Algorithms, 13 (1992), pp. 502 – 524.

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

[34] D. S. Johnson and M. R. Garey, *A $\frac{71}{60}$ theorem for bin packing*, J. Complexity, 1 (1985), pp. 65–106.

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

[36] R. M. Karp, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds., New York, 1972, Plenum Press, pp. 85–103.

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

[38] J. Y.-T. Leung, T. W. Tam, C. S. Wong, G. H. Young, and F. Y. L. Chin, *Packing squares into a square*, J. Parallel and Distributed Computing, 10 (1990), pp. 271–275.

[39] K. Li and K.-H. Cheng, *A generalized harmonic algorithm for on-line multidimensional bin packing*, TR UH-CS-90-2, University of Houston, January 1990.

[40] ——, *On three-dimensional packing*, SIAM J. Comput., 19 (1990), pp. 847–867.

[41] ——, *Static job scheduling in partitionable mesh connected systems*, J. Parallel and Distributed Computing, 10 (1990), pp. 152–159.

[42] ——, *A two-dimensional buddy system for dynamic resource allocation in a partitionable mesh connected system*, J. Parallel and Distributed Computing, 12 (1991), pp. 79–83.

[43] ——, *Generalized first-fit algorithms in two and three dimensions*, Int. J. Found. Comput Sci., 1 (1992), pp. 131–150.

[44] ——, *Heuristic algorithms for on-line packing in three dimensions*, J. of Algorithms, 13 (1992), pp. 589–605.

[45] A. Meir and L. Moser, *On packing of squares and cubes*, J. Combinatorial Theory Ser. A, 5 (1968), pp. 116–127.

[46] F. K. Miyazawa, *Algoritmos de Empacotamento Tridimensional: novas estratégias e análises de desempenho*, master's thesis, Universidade de São Paulo IME-USP, São Paulo-SP Brasil, dezembro 1993.

[47] F. K. Miyazawa and Y. Wakabayashi, *Polynomial approximation algorithms for the orthogonal z-oriented 3-D packing problem*, Tech. Report RT–MAC–9512, Instituto de Matemática e Estatística – Universidade de São Paulo – Brasil, 1995. Submetido para publicação.

[48] ——, *An algorithm for the three-dimensional packing problem with asymptotic performance analysis*, Algorithmica, 18 (1997), pp. 122–144.

[49] ——, *Approximation algorithms for packing small items*, in XX Congresso Nacional de Matemática Aplicada e Computacional (Mini Simpósio de Corte e Empacotamento), 1997.

[50] R. Morábito and S. Morales, *Uma nova heurística para o problema do carregamento de paletes do produtor*, in XX Congresso Nacional de Matemática Aplicada e Computacional (Mini Simpósio de Corte e Empacotamento), 1997.

[51] R. Motwani, *Lectures Notes on Approximation Algorithms*, Tech. Report STAN-CS-92-1435, Department of Computer Science, Stanford University, 1992.

[52] M. B. Richey, *Improved bounds for harmonic-based bin packing algorithms*, Discr. Appl. Math., 34 (1991), pp. 203–227.

[53] G. Scheithauer and J. Terno, *Theoretical investigations on the modified integer round-up property for the one-dimensional cutting stock problem*, Oper. Res. Lett., 20 (1997), pp. 93–100.

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

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

[56] Smith, A. and De Cani, P., *An algorithm to optimize the layout of boxes in pallets*, Journal of Operational Research, 31 (1980), pp. 573–578.

[57] A. Steinberg, *A strip-packing algorithm with absolute performance bound 2*, SIAM J. Comput., 26 (1997), pp. 401–409.

[58] P. E. Sweeney and E. R. Paternoster, *Cutting and packing problems: a categorized, application-oriented research bibliography*, J. Operational Research Society, 43 (1992), pp. 691 – 706.

[59] A. van Vliet, *An improved lower bound for online bin packing algorithms*, Inform. Process. Lett., 43 (1992), pp. 277–284.

[60] H. H. Yanasse, J. C. Furtado, L. A. N. Lorena, M. N. Arenales, N. Soma, N. Maculan, and R. Morábito, *O Problema de Corte e Empacotamento e Aplicações Industriais*, XX Congresso Nacional de Matemática Aplicada e Computacional e II Oficina Nacional de Problemas de Corte e Empacotamento. ICMSC-USP, 1997.

[61] M. Yue, *A simple proof of the inequality* $\mathrm{FFD}(L) \leq \frac{11}{9}\mathrm{OPT}(L) + 1$, $\forall L$ *for the FFD bin-packing algorithm*, Acta Math. Appl. Sin., Engl. Ser., 4 (1991), pp. 321–331.

Índice Remissivo