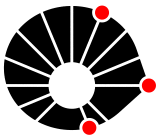

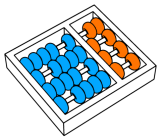

Algoritmos de Aproximação para o Problema do Empacotamento

R. V. Saraiva *R. C. S. Schouery*

Technical Report - IC-20-04 - Relatório Técnico
April - 2020 - Abril

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Algoritmos de Aproximação para o Problema do Empacotamento

Rachel Vanucchi Saraiva Rafael C. S. Schouery

14/04/2020

Resumo

O Problema do Empacotamento, em que busca-se a menor quantidade de recipientes necessários para armazenar um conjunto de itens, é muito relevante para o setor industrial por modelar problemas de logística, como armazenamento de produtos e corte de materiais. Como esse problema é NP-difícil, não pode ser resolvido em tempo eficiente a não ser que $P = NP$. Esse relatório reúne as provas das razões de aproximação de algoritmos de aproximação clássicos para o problema: Next Fit, First Fit, First Fit Decreasing, e os APTAS de Karmarkar e Karp e de La Vega e Lueker. Também contém provas gerais quanto a inaproximabilidade do problema por razão menor que 3/2, e resultados semelhantes para algoritmos online e de espaço limitado para o problema.

1 Introdução

A área de otimização combinatória estuda formas de tomar decisões que maximizem ou minimizem um certo valor desejado como, por exemplo, encontrar o número mínimo de recipientes necessários para guardar um conjunto de itens. Tal problema é conhecido como *Problema do Empacotamento*, e é o foco deste texto.

Formalmente, uma instância do Problema do Empacotamento é definida por n itens de tamanhos $a_1, \ldots, a_n$, e uma capacidade C, tal que $0 < a_i < C$ para $i = 1, \ldots, n$. Uma solução para o Problema do Empacotamento é uma partição dos itens, tal que a soma dos tamanhos dos itens em cada parte não ultrapasse C. Chamamos cada parte de um *recipiente*.

É possível, sem perda de generalidade, considerar $C = 1$, pois caso contrário basta dividir os tamanhos dos itens pela capacidade original. O valor da solução é a quantidade de recipientes usados, e é esse valor que buscamos minimizar. A Figura 1 mostra um exemplo de instância desse problema.

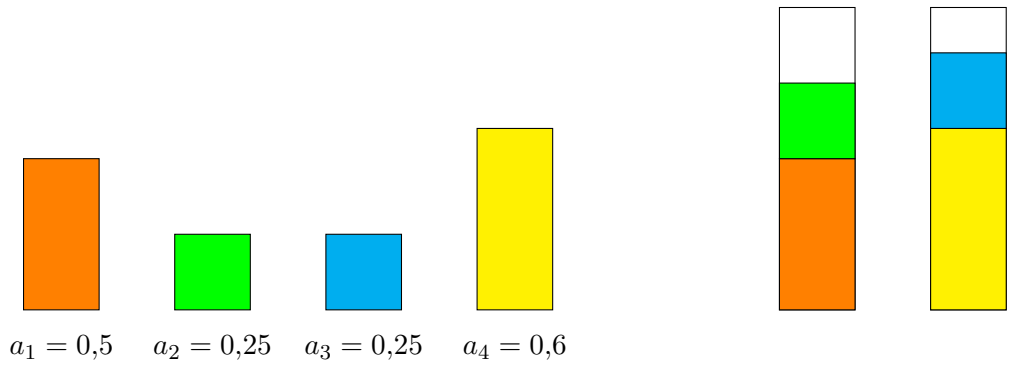

Figura 1: Exemplo de instância do Problema do Empacotamento com quatro itens, empacotada em dois recipientes.

O Problema do Empacotamento é de grande importância para o setor industrial por modelar problemas de logística, como o armazenamento de produtos e arquivos digitais, a distribuição de tarefas para diversos computadores, entre outros. Soluções boas para problemas de empacotamento minimizam os gastos com essas tarefas e assim podem reduzir o custo de produtos e serviços, tornando-os mais competitivos e acessíveis. Muitas vezes esses problemas também precisam ser resolvidos com rapidez e assim justifica-se a busca por algoritmos eficientes para esse problema.

O Problema do Empacotamento também pode ser aplicado em situações relacionadas ao corte de materiais, onde C representa o tamanho das placas, folhas ou rolos de material, e os itens são as peças a serem cortadas, de forma que o posicionamento das peças pode ser visto como um empacotamento das peças no material. Minimizando-se a quantidade de placas de material necessárias para obter as peças desejadas, reduz-se o custo da produção das mesmas. Variantes do problema conhecidas como Problema do Empacotamento Bidimensi-

onal e Tridimensional consideram peças e materiais com duas ou três dimensões e diferentes formas geométricas como, por exemplo, o empacotamento de caixas em containers. Nessas variantes, não basta decidir apenas quais itens serão colocados em cada recipiente, é necessário também considerar suas posições e rotações, e garantir que seus interiores não se sobreponham.

Outro problema a ser estudado, relacionado ao Problema do Empacotamento, é o *Problema da Mochila*. No Problema da Mochila são dados n itens que possuem um tamanho s_i e um valor v_i, para $i = 1, \ldots, n$, e uma mochila de capacidade B. O objetivo é achar um subconjunto desses itens cuja soma de seus valores seja a maior possível e a soma de seus tamanhos não ultrapasse B. É possível, sem perda de generalidade, considerar $B = 1$, caso contrário basta dividir os tamanhos dos itens pela capacidade original. Assim como o Problema do Empacotamento, o Problema da Mochila também é aplicado em questões de logística como carregamento de carga, buscando maximizar o aproveitamento de um espaço limitado. A Figura 2 mostra um exemplo de instância desse problema.

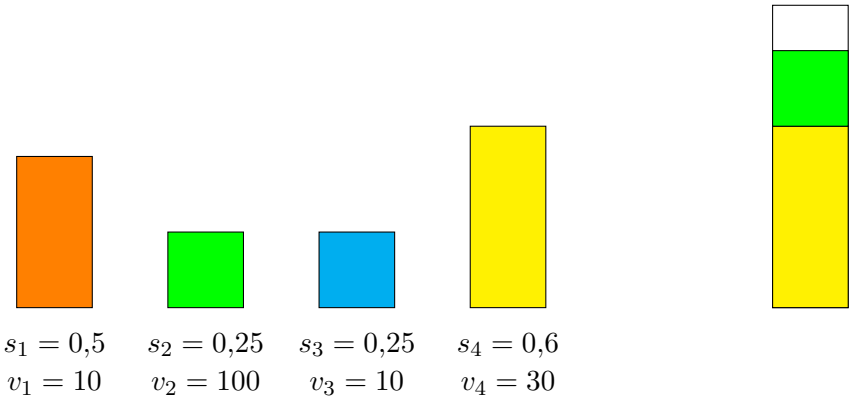

Figura 2: Exemplo de instância do Problema do Mochila com quatro itens e uma solução de valor 130.

O Problema do Empacotamento é NP-difícil e portanto não pode ser resolvido eficientemente, isto é, em tempo polinomial no tamanho de sua entrada, a não ser que $P = NP$. O mesmo vale para o Problema da Mochila. Assim, são necessárias abordagens especiais para

esses problemas. A abordagem considerada neste projeto é o desenvolvimento de *algoritmos de aproximação*, que encontram em tempo polinomial soluções garantidamente próximas de uma solução ótima.

2 Definições

Formalmente, dizemos que um algoritmo A é uma *α-aproximação* se A executa em tempo polinomial e, para qualquer instância I de um problema, encontra uma solução de valor $A(I)$ tal que, no caso de um problema de minimização, $A(I) \leq \alpha OPT(I)$, onde $OPT(I)$ é o valor de uma solução ótima para a instância e $\alpha \geq 1$. Por exemplo, uma 2-aproximação devolve uma solução com valor no máximo o dobro do valor de uma solução ótima. O fator α é chamado *razão de aproximação do algoritmo*. No caso de um problema de maximização, $A(I) \geq \alpha OPT(I)$ e $\alpha \leq 1$.

Chamamos de *Esquema de Aproximação de Tempo Polinomial* (PTAS, de *Polynomial-Time Approximation Scheme* em inglês) uma família de algoritmos $\{A_\varepsilon\}$, tal que para cada $\varepsilon > 0$, o algoritmo A_ε é uma $(1+\varepsilon)$-aproximação (para problemas de minimização) ou uma $(1-\varepsilon)$-aproximação (para problemas de maximização).

Algoritmos de aproximação também podem ser usados como medida do quão difícil é um problema, pois por vezes pode-se provar que, além do problema ser NP-difícil, também não pode existir α-aproximação para esse problema para um α suficientemente pequeno, a não ser que $P = NP$. Essa dificuldade em aproximar é chamada de *inaproximabilidade* do problema, e se aplica, por exemplo, ao Problema do Empacotamento, como pode ser visto no seguinte teorema.

Teorema 1. *(D. Simchi-Levi, 2006 [1]) Não existe uma α-aproximação para o Problema do Empacotamento com $\alpha < 3/2$, a não ser que $P = NP$.*

Demonstração. Nossa redução será a partir do *Problema da Partição*, um problema de decisão NP-difícil onde, dados n números inteiros $b_1, \ldots, b_n$ cuja soma $B = \sum_{i=1}^{n} b_i$ é par,

deve-se decidir se é possível particionar $\{1, \ldots, n\}$ em dois subconjuntos S e T tal que, $\sum_{i \in S} b_i = \sum_{i \in T} b_i$. A partir de uma instância I do Problema da Partição pode-se definir uma instância I' do Problema de Empacotamento associada a ele onde $C = B/2$ e $a_i = b_i$. Note que a resposta do Problema da Partição para I é "sim" se e somente se I' pode ser empacotada em exatamente 2 recipientes, ou seja, se $OPT(I') = 2$, onde $OPT(I')$ é o valor de uma solução ótima do Problema do Empacotamento para I'.

Se esse empacotamento fosse realizado por um algoritmo de aproximação A cuja razão de aproximação fosse menor que $3/2$, então, para cada instância I para a qual o problema da partição deve responder "sim", $A(I') < \frac{3}{2} OPT(I') = 3$. Como o valor de qualquer solução do empacotamento tem que ser inteiro (por representar uma quantidade de recipientes), $A(I') \leq 2 = OPT(I')$. Por outro lado, como $OPT(I') \leq A(I') < \frac{3}{2} OPT(I')$, se o algoritmo A encontra esse empacotamento em dois recipientes, então $OPT(I') = 2$. Portanto esse algoritmo decide o Problema da Partição em tempo polinomial, o que só pode ocorrer se $P = NP$, já que o Problema da Partição é NP-difícil. □

Apesar desse limitante, é possível encontrar aproximações melhores se a definição de razão de aproximação for relaxada, passando a permitir pequenos termos aditivos, o que é chamado de *razão de aproximação assintótica*. Um algoritmo A com razão de aproximação assintótica ρ produz solução de valor $A(I) \leq \rho OPT(I) + c$, onde c é uma constante. Para instâncias do problema onde é esperado que o valor da solução ótima seja muito maior que a constante c, ou seja, quando são necessários muitos recipientes, essa constante é desprezível e o algoritmo é praticamente uma ρ-aproximação. Assim, é possível achar algoritmos para o Problema do Empacotamento que possuam c pequeno e razão assintótica ρ menor que $3/2$. Isso motiva a definição de um *Esquema de Aproximação de Tempo Polinomial Assintótico* (APTAS, de *Asymptotic Polynomial-Time Approximation Scheme* em inglês), uma família de algoritmos $\{A_\varepsilon\}$ com uma constante c, onde para cada $\varepsilon > 0$, existe um algoritmo A_ε que encontra uma solução de valor no máximo $(1+\varepsilon)OPT(I)+c$ para problemas de minimização.

Um problema é chamado de *problema offline* se todos os dados de entrada da instância

são recebidos ao mesmo tempo. No caso do Problema do Empacotamento, ele é um problema offline quando já se sabe todos os itens a serem empacotados e seus tamanhos. Porém, em algumas situações os dados são recebidos com o tempo, não todos de uma vez, e é necessário tomar decisões irreversíveis assim que chegam sem ter conhecimento do resto da instância que chegará mais tarde. Esse tipo de problema é chamado *problema online*, e assim um algoritmo que devolve uma solução para esse tipo de problema é um *algoritmo online*. O Problema do Empacotamento é online se os itens são recebidos um a um e precisam ser colocados nos recipientes assim que chegam, sem ter conhecimento dos próximos itens e sem poder mais tarde retirar um item do recipiente escolhido, ou seja, a decisão feita no recebimento do item é final.

Por exigir que as decisões para compor a solução sejam tomadas sem que se tenha todos os dados do problema, os problemas online são geralmente mais difíceis de se achar solução ótima ou até mesmo de se aproximar. A eficiência de um algoritmo online é analisada de forma semelhante à análise de algoritmos de aproximação, ou seja, compara-se o valor da solução gerada com o valor da solução ótima do problema offline (solução que leva em consideração todos os dados da instância). Formalmente, um algoritmo online A é chamado *α-competitivo* se, para qualquer instância I de um problema online de minimização, retorna uma solução de valor $A(I)$ tal que $A(I) \leq \alpha OPT(I)$, onde $OPT(I)$ é o valor da solução ótima do problema offline. Assim como para os algoritmos de aproximação, também é possível relaxar essa definição para permitir termos aditivos.

Lema 1. *(A. C. Yao, 1980 [2]) Qualquer algoritmo online para o Problema do Empacotamento não pode ter razão de competitividade menor que* $3/2$.

Demonstração. Seja $x = \frac{1}{6} - 2\varepsilon$, $y = \frac{1}{3} + \varepsilon$ e $z = \frac{1}{2} + \varepsilon$, e sejam L_1, L_2 e L_3 três listas de itens. A lista L_1 contém n itens de tamanho x, a lista L_2 contém n itens de tamanho y, e a lista L_3 contém n itens de tamanho z, onde n é um múltiplo de 12.

Temos então que $OPT(L_1) = \frac{1}{6}n$, $OPT(L_1L_2) = \frac{1}{2}n$, e $OPT(L_1L_2L_3) = n$.

Analisamos agora o empacotamento de L_1 feito por um algoritmo online A qualquer.

Seja α_i o número de recipientes contendo i itens de L_1. Como não cabem mais que 6 itens de L_1 em um recipiente, temos que $A(L_1) = \sum_{i=1}^{6} \alpha_i$. Além disso, $n = \sum_{i=1}^{6} i\alpha_i$.

Considerando agora o empacotamento de L_1L_2. Seja $\alpha_{i,j}$ o número de recipientes contendo i itens de L_1 e j itens de L_2. Assim, como não cabem mais que 2 itens de L_2 em um recipiente, $\alpha_i = \sum_{j=0}^{2} \alpha_{i,j}$, e o número total de recipientes é $A(L_1L_2) = \sum_{i=1}^{6} \alpha_i + \alpha_{0,1} + \alpha_{0,2}$. Além disso, $n = \sum_{i=0}^{6} \alpha_{i,1} + 2\alpha_{i,2}$.

Vejamos agora o empacotamento de $L_1L_2L_3$. Após o empacotamento de L_1 e L_2, os recipientes não tem espaço para um item de L_3 caso já contenham mais que 3 itens de L_1, mais que 1 item de L_2, ou um item de L_2 e mais que 1 item de L_1. Além disso, qualquer recipiente só pode conter um item de L_3. Portanto temos que

$$A(L_1L_2L_3) \geq \sum_{i=4}^{6} \alpha_{i,0} + \sum_{i=0}^{6} \alpha_{i,2} + \sum_{i=2}^{6} \alpha_{i,1} + n.$$

Como recipientes contendo um item de L_2 não podem conter mais que 4 itens de L_1, $\alpha_{i,1} = 0$ para $i > 4$. Da mesma forma, recipientes com 2 itens de L_2 não podem conter mais que 2 itens de L_1, portanto $\alpha_{i,2} = 0$ para $i > 2$. As equações vistas até então podem ser reescritas da seguinte forma:

$$A(L_1) = \alpha_{1,1} + \sum_{i=1}^{2} \alpha_{i,2} + \sum_{i=1}^{3} \alpha_{i,0} + \sum_{i=2}^{4} \alpha_{i,1} + \sum_{i=4}^{6} \alpha_{i,0} \tag{1}$$

$$n \leq \alpha_{1,1} + 2\sum_{i=1}^{2} \alpha_{i,2} + 3\sum_{i=1}^{3} \alpha_{i,0} + 4\sum_{i=2}^{4} \alpha_{i,1} + 6\sum_{i=4}^{6} \alpha_{i,0} \tag{2}$$

$$A(L_1L_2) = \alpha_{1,1} + \sum_{i=1}^{2} \alpha_{i,2} + \sum_{i=1}^{3} \alpha_{i,0} + \sum_{i=2}^{4} \alpha_{i,1} + \sum_{i=4}^{6} \alpha_{i,0} + \alpha_{0,1} + \alpha_{0,2} \tag{3}$$

$$n = \alpha_{1,1} + 2\sum_{i=1}^{2} \alpha_{i,2} + \sum_{i=2}^{4} \alpha_{i,1} + \alpha_{0,1} + 2\alpha_{0,2} \tag{4}$$

$$A(L_1L_2L_3) \geq \sum_{i=1}^{2} \alpha_{i,2} + \sum_{i=2}^{4} \alpha_{i,1} + \sum_{i=4}^{6} \alpha_{i,0} + \alpha_{0,2} + n. \tag{5}$$

Seja ρ a razão de competitividade do algoritmo. Se $\rho < 3/2$, então $A(L_1) \leq \frac{1}{6}n\rho < \frac{1}{4}n$, $A(L_1L_2) \leq \frac{1}{2}n\rho < \frac{3}{4}n$ e $A(L_1L_2L_3) \leq n\rho < \frac{3}{2}n$. Somando essas três inequações temos:

$$\frac{5}{2}n > 2\alpha_{1,1} + 3\sum_{i=1}^{2}\alpha_{i,2} + 2\sum_{i=1}^{3}\alpha_{i,0} + 3\sum_{i=2}^{4}\alpha_{i,1} + 3\sum_{i=4}^{6}\alpha_{i,0} + \alpha_{0,1} + 2\alpha_{0,2} + n.$$

Subtraindo n dos dois lados obtemos

$$\frac{3}{2}n > 2\alpha_{1,1} + 3\sum_{i=1}^{2}\alpha_{i,2} + 2\sum_{i=1}^{3}\alpha_{i,0} + 3\sum_{i=2}^{4}\alpha_{i,1} + 3\sum_{i=4}^{6}\alpha_{i,0} + \alpha_{0,1} + 2\alpha_{0,2}.$$

Subtraindo agora dos dois lados a equação (4),

$$\frac{1}{2}n > \alpha_{1,1} + \sum_{i=1}^{2}\alpha_{i,2} + 2\sum_{i=1}^{3}\alpha_{i,0} + 2\sum_{i=2}^{4}\alpha_{i,1} + 3\sum_{i=4}^{6}\alpha_{i,0}.$$

Finalmente, dividindo a inequação (2) por 2 e a subtraindo do resultado anterior, temos que $0 > \frac{1}{2}\alpha_{1,1} + \frac{1}{2}\sum_{i=1}^{3}\alpha_{i,0}$, o que é uma contradição, já que as quantidades de recipientes são valores não negativos. Portanto, $\rho \geq 3/2$. □

Nos algoritmos para o Problema do Empacotamento, consideramos que um recipiente está *fechado* quando não será colocado mais nenhum item nele. Um recipiente que ainda pode receber novos itens é chamado de recipiente *aberto*. Um algoritmo online que sempre mantém no máximo um número constante de recipientes abertos é chamado de *algoritmo de espaço limitado*.

Lema 2. *(C. C. Lee e D. T. Lee, 1985 [3]) Se um algoritmo online para o Problema do Empacotamento é de espaço limitado, então sua razão de competitividade não pode ser menor que* 1,6910 . . . .

Demonstração. Suponha que o algoritmo em questão mantenha no máximo T recipientes abertos durante sua execução. Seja $n \geq 2$ a quantidade de itens em uma instância, e i um

divisor de n tal que $i \geq 2$, e seja

$$x_j = \begin{cases} \frac{1}{k_j+1} + \varepsilon, \text{ se } 1 \leq j < i, \\ \frac{1}{k_i} - (i-1)\varepsilon, \text{ se } j = i \end{cases}$$

onde $k_1 = 1$ e $k_{j+1} = k_j(k_j + 1)$, para $2 \leq j < i$. Seja também I uma instância contendo, em ordem não crescente, n/i itens de tamanho x_j para cada valor de $j = 1, \ldots, i$.

Note que a regra de recorrência da definição de k_j implica que

$$\frac{1}{k_j} - \frac{1}{k_{j+1}} = \frac{1}{k_j} - \frac{1}{k_j(k_j+1)} = \frac{1}{k_j+1}.$$

Portanto,

$$\begin{aligned} \sum_{j=1}^{i} x_j &= \left(\sum_{j=1}^{i-1} \frac{1}{k_j+1} + \varepsilon \right) + \frac{1}{k_i} - (i-1)\varepsilon \\ &= \left(\sum_{j=1}^{i-1} \frac{1}{k_j} - \frac{1}{k_{j+1}} \right) + \frac{1}{k_i} \\ &= \frac{1}{k_1} - \frac{1}{k_i} + \frac{1}{k_i} = 1. \end{aligned}$$

Assim, é possível empacotar os itens de forma ótima colocando um item de tamanho x_j em cada recipiente, para $j = 1, \ldots, i$, e portanto $OPT(I) = n/i$. Porém, como um algoritmo online decide o recipiente em que cada item será colocado na ordem em que os itens aparecem na instância, e os itens estão em ordem não crescente, qualquer algoritmo online abrirá inicialmente um novo recipiente para cada item de tamanho x_1, já que seu tamanho é $1/2 + \varepsilon$. A partir disso, para cada $j = 2, \ldots, i$, como cabem no máximo k_j itens de tamanho x_j em um recipiente, e há n/i itens desse tamanho na instância, serão necessários no mínimo $\frac{(n/i)-k_jT}{k_j}$ novos recipientes para empacotar esses itens.

Portanto qualquer algoritmo online usa no mínimo $\sum_{j=1}^{i} \frac{(n/i)-k_jT}{k_j}$ recipientes para essa

instância, então a razão de competitividade assintótica não pode ser melhor que $\sum_{j=1}^{i} 1/k_j$, e como i pode ser arbitrariamente grande, temos $\sum_{j=1}^{\infty} 1/k_j = 1,6910\ldots$. □

Apresentamos a seguir uma técnica que será muito utilizada nas análises dos algoritmos estudados.

Definição 1. *Para a análise de um algoritmo $A(I)$, por vezes é possível definir uma função de ponderação $W_A(a)$, que associa cada item a a um valor real seguindo duas propriedades:*

$$\sum_{a \in I} W_A(a) \geq A(I) - c \tag{6}$$

$$\sum_{a \in B} W_A(a) \leq \alpha \tag{7}$$

onde I é uma instância do Problema do Empacotamento, B é qualquer subconjunto de itens de I cuja soma dos tamanhos não ultrapassa 1 (ou seja, qualquer subconjunto que possa ser empacotado em um único recipiente), e c e α são constantes não negativas.

Para facilitar a notação, podemos definir que, para qualquer conjunto S de itens, $W_A(S) = \sum_{a \in S} W_A(a)$.

Lema 3. *Se W_A é uma função que respeita as propriedades (1) e (2), então o valor de uma solução dada pelo algoritmo é $A(I) \leq \alpha OPT(I) + c$.*

Demonstração. Com a propriedade (2) temos que, se $\{B_1^*, B_2^*, \ldots, B_{OPT(I)}^*\}$ são os recipientes de uma solução ótima, então $W_A(I) = \sum_{j=1}^{OPT(I)} W_A(B_j^*) \leq \alpha OPT(I)$. Assim, com a propriedade (1), temos que $A(I) \leq \alpha OPT(I) + c$, e assim encontramos um limitante superior para o algoritmo. □

Para facilitar a notação nesse texto, definimos também a função $SIZE(S) = \sum_{a \in S} a$, para qualquer conjunto S de itens. Essa função será frequentemente usada para descrever o quanto um recipiente foi ocupado, ou seja, a soma dos tamanhos dos itens nele colocados.

3 Next Fit (NF)

O algoritmo *Next Fit* (NF) é um algoritmo online de espaço limitado, que empacota o primeiro item no primeiro recipiente e, a partir disso, verifica se cada próximo item cabe no recipiente no qual o último item foi colocado. Se sim, o item é colocado nesse recipiente também; do contrário, fecha-se o recipiente, abre-se um novo e o item é colocado neste, como descrito no pseudo-código a seguir, onde B_j são os recipientes da solução e b o número de recipientes usados.

```
NEXT FIT(I)
1  b = 1
2  para i = 1 ate n
3       se a_i cabe no recipiente B_b
4            Empacote a_i em B_b
5       senão
6            Feche recipiente B_b
7            b = b + 1
8            Empacote a_i em B_b
```

A Figura 3 mostra um exemplo de instância empacotada por esse algoritmo. O algoritmo empacota os dois primeiros itens no primeiro recipiente. Após isso, o primeiro recipiente tem apenas 0,1 de espaço livre, que não é suficiente para empacotar o terceiro item, portanto esse recipiente é fechado e é aberto um segundo recipiente, em que são colocados o terceiro e quarto itens. Após isso o segundo recipiente não tem mais espaço livre, portanto é fechado, e é aberto um terceiro recipiente para o último item.

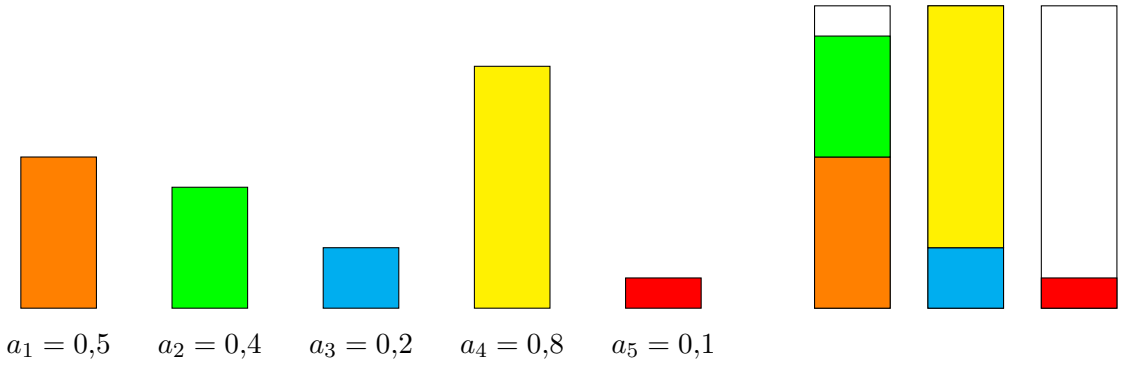

Figura 3: Exemplo de instância do Problema do Empacotamento empacotada por *Next Fit*.

Teorema 2. *(D. S. Johnson, 1974 [4]) Para qualquer instância I do Problema do Empacotamento, o algoritmo Next Fit devolve uma solução cuja quantidade de recipientes utilizada é $NF(I) \leq 2OPT(I) + 1$.*

Demonstração. Definimos uma função de ponderação para o *Next Fit* como sendo o dobro do tamanho do item, isto é, $W_{NF}(a) = 2a$.

Uma propriedade importante do *Next Fit* é que, em qualquer solução que use mais que um recipiente, um recipiente só é aberto quando um item não cabe no recipiente anterior, portanto o tamanho total dos itens em qualquer par de recipientes B_j e B_{j+1} é maior que 1. Assim, temos que

$$W_{NF}(I) = \sum_{j=1}^{NF(I)} \sum_{a \in B_j} 2a \geq 2 \sum_{j=1}^{\lfloor NF(I)/2 \rfloor} \sum_{a \in B_{2j-1} \cup B_{2j}} a \geq 2\lfloor NF(I)/2 \rfloor \geq NF(I) - 1$$

e assim a função de ponderação usada obedece a propriedade (1), com $c = 1$.

Além disso, $\sum_{a \in B} W_A(a) = 2 \sum_{a \in B} a \leq 2$, já que a soma dos tamanhos em um recipiente é menor ou igual a 1. Portanto a função também obedece a propriedade (2), com $\alpha = 2$, e temos então, pelo Lema 3 que o total de recipientes usado pelo algoritmo é $NF(I) \leq 2OPT(I) + 1$. □

4 First Fit (FF)

O algoritmo *First Fit* é um algoritmo online (porém de espaço ilimitado, ao contrário do *NF*), que empacota cada item no primeiro recipiente aberto em que esse item cabe. Um novo recipiente é aberto apenas quando o item não cabe em nenhum dos recipientes abertos no momento, como descrito no pseudo-código abaixo, onde B_j são os recipientes e b o número de recipientes abertos em cada iteração.

```
First Fit(I)
1  b = 1
2  para i = 1 ate n
3       se existe j ≤ b tal que a_i cabe em B_j
4            Seja j* o menor j tal que a_i cabe em B_j
5            Empacote a_i no recipiente B_j*
6       senão
7            b = b + 1
8            Empacote a_i no recipiente B_b
```

A Figura 4 mostra um exemplo de instância empacotada por esse algoritmo. O algoritmo empacota os dois primeiros itens no primeiro recipiente. Após isso, o primeiro recipiente tem apenas 0,1 de espaço livre, que não é suficiente para empacotar o terceiro ou o quarto item, portanto é aberto um segundo recipiente, em que são colocados esses dois itens. O quinto item tem tamanho 0,1 e portanto cabe no primeiro recipiente, logo é colocado neste. Nesse exemplo, a solução encontrada é ótima, mas isso não ocorre em geral.

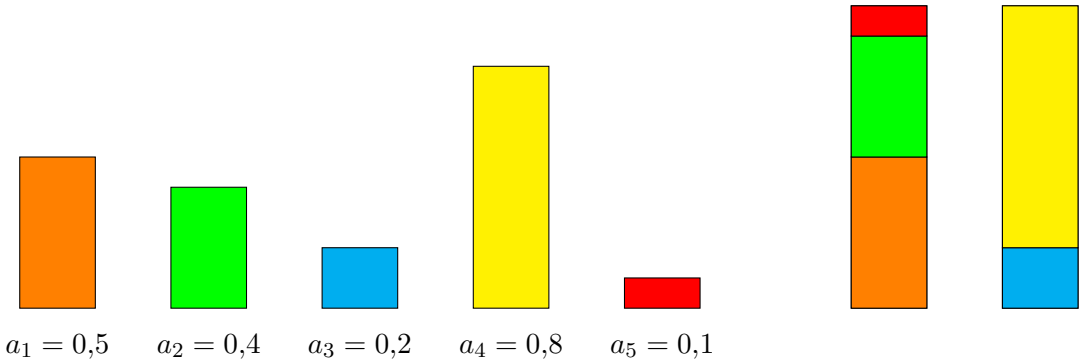

Figura 4: Exemplo de instância do Problema do Empacotamento empacotada por First Fit.

Definimos a seguinte função de ponderação para o algoritmo:

$$W_{FF}(a) = \begin{cases} \frac{6}{5}a & 0 \leq a < 1/6 \\ \frac{9}{5}a - 1/10 & 1/6 \leq a < 1/3 \\ \frac{6}{5}a + 1/10 & 1/3 \leq a \leq 1/2 \\ 1 & 1/2 < a < 1 \end{cases}$$

Lema 4. *A função W_{FF} respeita a propriedade (2) de função de ponderação quando $\alpha = \frac{17}{10}$, ou seja, $W_{FF}(B) \leq 17/10$, onde B é qualquer subconjunto de itens cuja soma dos tamanhos não ultrapassa 1.*

Demonstração. A prova é simples quando o tamanho de nenhum item em B ultrapassa $1/2$, pois nesse caso:

- Para os itens a tais que $a < 1/6$, $\frac{W_{FF}(a)}{a} = \frac{6}{5} < \frac{17}{10}$;
- Para os itens a tais que $1/6 \leq a < 1/3$, $\frac{W_{FF}(a)}{a} = \frac{9}{5} - \frac{1}{10a} \leq 9/5 - 3/10 = 15/10$;
- Para os itens a tais que $1/3 \leq a \leq 1/2$, $\frac{W_{FF}(a)}{a} = \frac{6}{5} + \frac{1}{10a} \leq 6/5 + 3/10 = 15/10$;

Portanto $\frac{W_{FF}(a)}{a} \leq 17/10$ para todo $a \leq 1/2$, então $W_{FF}(B) \leq \frac{17}{10} \sum_{a \in B} a \leq 17/10$.

Se B possui um item de tamanho maior que $1/2$, então seus outros itens $a_1 \leq \cdots \leq a_t$ têm que ser menores que $1/2$, e precisamos mostrar que $\sum_{i=1}^{t} W_{FF}(a_i) \leq 7/10$.

Para reduzir o número de casos que precisam ser analisados, podemos assumir que $a_i \leq 1/3,\ i = 1, \ldots, t$. Isso porque todo item maior que $1/3$ pode ser tratado como dois itens, $a_i^1 = 1/3$ e $a_i^2 = a_i - 1/3 < 1/6$, sem nenhuma alteração no somatório dos pesos, já que $W_{FF}(a_i) = W_{FF}(a_i^1) + W_{FF}(a_i^2)$.

Similarmente, podemos assumir que só há no máximo um item menor que $1/6$. Pares $\{a_i, a_j\}$ de itens menores que $1/6$ podem ser combinados em um único item $a_c < 1/3$, sem redução no somátorio dos pesos, pois $W_{FF}(a_c) \geq W_{FF}(a_i) + W_{FF}(a_j)$.

Com essas reduções, temos que B contém no máximo 3 itens menores que $1/2$, já que a soma deles também tem que ser menor que $1/2$ por já haver um item maior que $1/2$ em B. Resta então analisar cada caso:

Caso 1: $t = 1$

- Subcaso 1: $a_1 < 1/6$.

  Temos que $W_{FF}(a_1) < 1/5 < 7/10$;

- Subcaso 2: $1/6 \leq a_1 \leq 1/3$.

  Temos que $W_{FF}(a_1) \leq 3/5 - 1/10 < 7/10$;

Caso 2: $t = 2$

- Subcaso 1: $a_1 < 1/6 \leq a_2 \leq 1/3$.

  Temos que $W_{FF}(a_1) + W_{FF}(a_2) \leq 1/5 + 3/5 - 1/10 = 7/10$;

- Subcaso 2: $1/6 \leq a_1, a_2 \leq 1/3$.

  Temos que $W_{FF}(a_1) + W_{FF}(a_2) \leq \frac{9}{5}(a_1 + a_2) - 1/5$.

  Como $a_1 + a_2 < 1/2$, temos $W_{FF}(a_1) + W_{FF}(a_2) < 9/10 - 1/5 = 7/10$;

Caso 3: $t = 3$

- Subcaso 1: $a_1 < 1/6 \leq a_2, a_3 \leq 1/3$.

  Temos que $W_{FF}(a_1) + W_{FF}(a_2) + W_{FF}(a_3) \leq \frac{6}{5}a_1 + \frac{9}{5}(a_2 + a_3) - 2/10$.

  Como $a_2 + a_3 < 1/2 - a_1$, temos

$$\begin{aligned} W_{FF}(a_1) + W_{FF}(a_2) + W_{FF}(a_3) &\leq \frac{6}{5}(a_1 + a_2 + a_3) + \frac{3}{5}(a_2 + a_3) - 2/10 \\ &< 6/10 + 3/10 - \frac{3}{5}a_1 - 2/10 \\ &< 7/10 - \frac{3}{5}a_1 < 7/10. \end{aligned}$$

- Subcaso 2: $1/6 \leq a_1, a_2, a_3 \leq 1/3$.

  Temos que $W_{FF}(a_1) + W_{FF}(a_2) + W_{FF}(a_3) < \frac{6}{5}(a_1 + a_2 + a_3) - 3/10$.

  Como $a_1 + a_2 + a_3 < 1/2$, temos $W_{FF}(a_1) + W_{FF}(a_2) + W_{FF}(a_3) < 3/10 < 7/10$.

Assim, o resultado segue. □

Para as provas dos próximos lemas é necessário definir o conceito de *granularidade* de um recipiente.

Definição 2. *Um recipiente B_i possui granularidade g_i se, para cada $B_j, j < i$, temos que $SIZE(B_j) \geq 1 - g_i$, e existe $B_j, j < i$ tal que $SIZE(B_j) = 1 - g_i$.*

A granularidade representa o maior espaço disponível nos recipientes antes de B_i, ou seja, B_i possui apenas itens maiores que g_i, pois itens menores cabem nos recipientes anteriores, e portanto não teriam sido empacotados em B_i pelo *FF*.

O lema a seguir utiliza desse conceito para limitar inferiormente a função de ponderação para certos recipientes.

Lema 5. *Para qualquer recipiente B_i com $g_i < 1/2$ contendo itens $a_1 \geq \cdots \geq a_k$, se $SIZE(B_i) \geq 1 - g_i$ então $W_{FF}(B_i) \geq 1$.*

Demonstração. Se $a_1 > 1/2$ o resultado é imediato pois $W_{FF}(a_1) = 1$. Se $a_1 \leq 1/2$ temos três casos:

Caso 1: $g_i \leq 1/6$. Então, pela hipótese, $\sum_{j=1}^{k} a_j \geq 1 - g_i \geq 5/6$. Como $a_j \leq 1/2$ para $j = 1, \ldots, k$, então $W_{FF}(a_j)/a_j \geq 6/5$, e portanto $\sum_{j=1}^{k} W_{FF}(a_j) \geq 6/5 \cdot 5/6 = 1$.

Caso 2: $1/6 < g_i \leq 1/3$. Nesse caso, $k \geq 2$, pois do contrário $a_1 \geq 1 - g_i \geq 2/3$, o que contradiz $a_1 \leq 1/2$.

- Subcaso 1: $a_1 \geq a_2 \geq 1/3$. Nesse caso, $W_{FF}(a_1) + W_{FF}(a_2) \geq 2(6/5 \cdot 1/3 + 1/10) = 1$.

- Subcaso 2: $a_1 \geq 1/3 \geq a_2$. Nesse caso, como todo item no recipiente é maior que g_i, temos

$$\begin{aligned} W_{FF}(B_i) &= W_{FF}(a_1) + W_{FF}(a_2) + \sum_{j=3}^{k} W_{FF}(a_j) \\ &\geq \frac{6}{5}a_1 + \frac{1}{10} + \frac{9}{5}a_2 - \frac{1}{10} + \frac{6}{5}\sum_{j=3}^{k} a_j \\ &= \frac{6}{5}\sum_{j=1}^{k} a_j + \frac{3}{5}a_2 \\ &\geq \frac{6}{5}(1 - g_i) + \frac{3}{5}g_i \\ &= \frac{6}{5} - \frac{3}{5}g_i \geq 1. \end{aligned}$$

- Subcaso 3: $1/3 \geq a_1$. Nesse caso, $k \geq 3$, pois do contrário $a_1 + a_2 \geq 1 - g_i \geq 2/3$, o

que seria uma contradição já que $a_2 \leq a_1 \leq 1/3$. Então

$$\begin{aligned} W_{FF}(B_i) &= W_{FF}(a_1) + W_{FF}(a_2) + \sum_{j=3}^{k} W_{FF}(a_j) \\ &\geq \frac{9}{5}a_1 - \frac{1}{10} + \frac{9}{5}a_2 - \frac{1}{10} + \frac{6}{5}\sum_{j=3}^{k} a_j \\ &= \frac{6}{5}\sum_{j=1}^{k} a_j + \frac{3}{5}a_1 + \frac{3}{5}a_2 - \frac{1}{5} \\ &\geq \frac{6}{5}(1-g_i) + \frac{6}{5}g_i - \frac{1}{5} = 1. \end{aligned}$$

Caso 3: $1/3 < g_i < 1/2$. Novamente $k \geq 2$, pois do contrário $a_1 \geq 1 - g_i > 1/2$. Vale a mesma análise feita no subcaso 1 do caso anterior, já que todos os itens são maiores que g_i, e portanto maiores que $1/3$. □

O próximo lema também utiliza a granularidade dos recipientes para limitar superiormente o quão cheio certos recipientes podem estar.

Lema 6. *Se um recipiente B_i possui $g_i \leq 1/2$, contém itens $a_1 \geq \cdots \geq a_k$, tais que $W_{FF}(B_i) = 1 - \beta$, para $\beta > 0$, então ou $k = 1$ e $a_1 \leq 1/2$, ou $SIZE(B_i) \leq 1 - g_i - \frac{5}{6}\beta$.*

Demonstração. Se $k = 1$ e $a_1 > 1/2$ então $W_{FF}(a_1) = 1$, o que contradiz a hipótese.

Se $k \geq 2$, então seja γ tal que $SIZE(B_i) = 1 - g_i - \gamma$. Se $\gamma \leq 0$, então poderíamos aplicar o Lema 5 e obter $W_{FF}(B_i) \geq 1$, o que contradiz a hipótese. Então $0 \leq \gamma \leq 1 - g_i$.

Sejam $d_1, \ldots, d_6$ seis novos itens, todos de tamanho $\gamma/6$, e seja C um novo recipiente que contém todos os itens de B_i, e também os itens $d_1, \ldots, d_6$. Então como a somatória dos tamanhos dos itens em C é $SIZE(B_i) + \sum_{j=1}^{6} d_j = 1 - g_i$, podemos aplicar o Lema 5 à C, e portanto $W_{FF}(C) \geq 1$.

Como $d_j = \gamma/6 \leq 1/6$, então $W_{FF}(d_j) = \frac{6}{5}d_j$, $j = 1, \ldots, 6$. Então $1 \leq W_{FF}(C) = W_{FF}(B_i) + \frac{6}{5}\sum_{j=1}^{6} d_j = 1 - \beta + \frac{6}{5}\gamma$. Portanto $\gamma \geq \frac{5}{6}\beta$, e $SIZE(B_i) = 1 - g_i - \gamma \leq 1 - g_i - \frac{5}{6}\beta$. □

Com isso podemos provar o último lema, relacionado à propriedade (1) da função de ponderação.

Lema 7. *Sendo $B_1, \ldots, B_{FF(I)}$ os recipientes da solução dada pelo First-Fit, temos que $W_{FF}(I) \geq FF(I) - 2$, ou seja, a função $W_{FF}(a)$ respeita a propriedade (1) de funções de ponderação, com $c = 2$.*

Demonstração. Para os recipientes que contém um item maior que $1/2$, $W_{FF}(B) \geq 1$. Para os outros recipientes, $W_{FF}(B_i) \leq 7/10 < 1$, como visto na prova do Lema 4. Então para esses recipientes definimos $W_{FF}(B_i) = 1 - \beta_i$, $0 < \beta_i < 1$.

Pela definição de granularidade, $SIZE(B_{i-1}) \geq 1 - g_i$. Podemos aplicar o Lema 6, obtendo $SIZE(B_{i-1}) \leq 1 - g_{i-1} - \frac{5}{6}\beta_{i-1}$. Portanto, $g_i \geq 1 - SIZE(B_{i-1}) \geq g_{i-1} + \frac{5}{6}\beta_{i-1}$, e logo $\frac{6}{5}(g_i - g_{i-1}) \geq \beta_{i-1}$.

Sejam $B_{i_1}, \ldots, B_{i_l}$ os recipientes que não contém itens maiores que $1/2$, com $i_1 \leq \cdots \leq i_l$. A granularidade de qualquer um desses recipientes é menor ou igual a $1/2$, do contrário eles só conteriam um único item maior que $1/2$. Portanto temos que

$$\sum_{k=1}^{l-1} \beta_{i_k} \leq \frac{6}{5} \sum_{k=2}^{l} g_{i_k} - g_{i_{k-1}} = \frac{6}{5}(g_{i_l} - g_{i_1}) \leq \frac{6}{5} \cdot \frac{1}{2} < 1.$$

Além disso, $\beta_{i_l} < 1$, portanto $\sum_{k=1}^{l} \beta_{i_k} \leq 2$.

Seja m a quantidade de recipientes com um item maior que $1/2$. Temos então que $m + l = FF(I)$. Então

$$\begin{aligned} W_{FF}(I) &\geq m + \sum_{k=1}^{l} W_{FF}(B_{i_k}) \\ &= m + \sum_{k=1}^{l} 1 - \sum_{k=1}^{l} \beta_{i_k} \\ &= m + l - \sum_{k=1}^{l} \beta_{i_k} \\ &\geq FF(I) - 2. \end{aligned}$$

Assim, o resultado segue. □

Juntos, os Lemas 4 e 7 provam que a quantidade de recipientes usados na solução dada pelo algoritmo é $FF(I) \leq \frac{17}{10}OPT(I) + 2$ [5].

5 First Fit Decreasing (FFD)

O algoritmo *First Fit Decreasing* é uma variação offline do *First Fit* onde os itens da instância são colocados em ordem não-crescente antes de serem empacotados.

FirstFitDecreasing(I)

1 Ordene I em ordem não-crescente

2 FirstFit(I)

Lema 8. *Seja I uma instância para a qual $FFD(I) > \alpha OPT(I) + c$, com $\alpha \geq 1$ e $c \geq 1$, e seja I' uma nova instância obtida ao descartar de I todos os itens cujos tamanhos não excedem $(\alpha - 1)/\alpha$. Temos então que $FFD(I') > \alpha OPT(I') + c$.*

Demonstração. Se $FFD(I) > FFD(I')$, então o último recipiente do empacotamento de I contém apenas itens que não excedem $(\alpha - 1)/\alpha$, e portanto todos os recipientes anteriores já estavam cheios pelo menos até $1/\alpha$.

Portanto $(1/\alpha)(FFD(I) - 1) < SIZE(I) \leq OPT(I)$, logo $FFD(I) < \alpha OPT(I) + 1$, o que contradiz a hipótese. Então $FFD(I) = FFD(I')$. E o descarte de itens feito para obter a instância I' não pode ter aumentado o valor da solução ótima, ou seja, $OPT(I) \geq OPT(I')$, logo $FFD(I') > \alpha OPT(I') + c$. □

Lema 9. *Seja $I = I_1 \cup I_2$, em que I_2 contém todos os itens cujos tamanhos pertencem ao intervalo $(0, 1/m]$, para algum $m > 1$. Se $FFD(I_1) \leq OPT(I) + k$ para algum $k \geq 0$, então $FFD(I) \leq \frac{m+1}{m}OPT(I) + k + 1$.*

Demonstração. Dividimos o empacotamento feito pelo FFD em quatro partes, como mostra a Figura 5:

- P_1 contém os recipientes R_j tais que $j \leq \min\{OPT(I), FFD(I) - 1\}$;
- P_2 contém os recipientes R_j tais que $OPT(I) < j \leq \min\{OPT(I) + k, FFD(I) - 1\}$;
- P_3 contém os recipientes R_j tais que $OPT(I) + k < j \leq FFD(I) - 1$;
- P_4 contém o último recipiente.

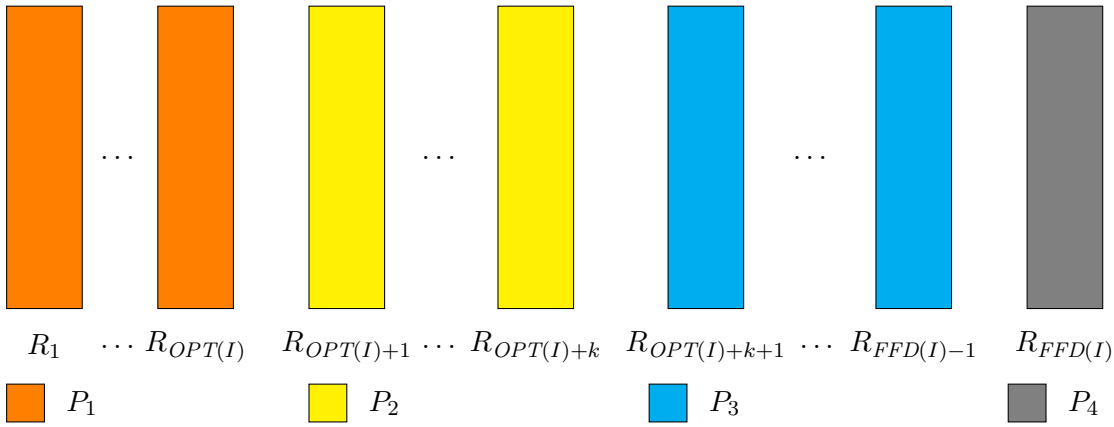

Figura 5: Divisão do empacotamento feito pelo FFD.

Se $P_3 = \emptyset$ a prova é trivial pois $FFD(I) = |P_1| + |P_2| + |P_4| = OPT(I) + k + 1$. Senão, $|P_3| = FFD(I) - OPT(I) - k - 1$.

Seja d o maior espaço vazio nos recipientes de P_1, e $SIZE(P_i) = \sum_{R_j \in P_i} SIZE(R_j)$. Então $SIZE(P_1) \geq (1 - d)|P_1| = (1 - d)\,OPT(I)$.

Como $FFD(I_1) \leq OPT(I) + k$, nenhum item de um recipiente de P_3 pertence a I_1. Então P_3 contém apenas itens de I_2, e em cada recipiente cabem pelo menos m desses itens, e todos esses itens são maiores que d, do contrário teriam sido colocados nos recipientes anteriores. Então $SIZE(P_3) > md|P_3| = md(FFD(I) - OPT(I) - k - 1)$.

Logo temos

$$\begin{aligned} OPT(I) &\geq SIZE(I) \\ &\geq SIZE(P_1) + SIZE(P_3) \\ &\geq (1-d)OPT(I) + md(FFD(I) - OPT(I) - k - 1) \\ &\geq OPT(I) - (m+1)dOPT(I) + md(FFD(I) - k - 1). \end{aligned}$$

Portanto $(m+1)dOPT(I) \geq md(FFD(I) - k - 1)$, e assim

$$FFD(I) \geq \frac{m+1}{m} OPT(I) + k + 1. \qquad \square$$

Nos lemas a seguir, dividimos a instância em cinco sublistas:

- A contém os itens cujos tamanhos estão no intervalo $(1/2, 1]$.
- B contém os itens cujos tamanhos estão no intervalo $(1/3, 1/2]$.
- C contém os itens cujos tamanhos estão no intervalo $(1/4, 1/3]$.
- D contém os itens cujos tamanhos estão no intervalo $(1/5, 1/4]$.
- E contém os itens cujos tamanhos estão no intervalo $(2/11, 1/5]$.

Lema 10. *Se I é uma instância para a qual $FFD(I) > \alpha OPT(I) + c$, e I' é uma nova instância obtida ao descartar de I os itens menores que o último item a ser empacotado em um recipiente que não contém nenhum item de A, então $FFD(I') > \alpha OPT(I') + c$.*

Demonstração. Como os itens de A são os maiores, são empacotados primeiro, portanto os recipientes que os contém são os primeiros da solução dada pelo FFD. Qualquer item colocado nesses recipientes não muda o número total de recipientes usados. Portanto

$FFD(I) = FFD(I')$. E o descarte de itens não pode aumentar o valor da solução ótima, portanto $FFD(I') > \alpha OPT(I') + c$. □

O lema a seguir mostra que para o caso particular de instâncias apenas com itens maiores que 1/3, o FFD obtem a solução ótima.

Lema 11. *Se I contém itens apenas no intervalo $(1/3, 1]$, ou seja, as sublistas C, D e E são vazias, então $FFD(I) = OPT(I)$.*

Demonstração. Um recipiente pode conter apenas um único item da sublista A, portanto a quantidade de recipientes gastos para empacotar os itens de A é a mesma tanto em uma solução ótima quanto na solução do FFD. Nesses recipientes, cabe apenas um item de B.

Seja $B^* = \{b_1, b_2, \ldots, b_{k^*}\}$ a lista de itens que pertencem a B e que foram empacotados no mesmo recipiente que um item de A em uma solução ótima, em ordem não crescente de tamanho, e a_j o item de A empacotado no mesmo recipiente que $b_j, j = 1, \ldots, k^*$. Seja também B^F a lista de itens que pertencem a B e que foram empacotados no mesmo recipiente que um item de A na solução do FFD. Provaremos por indução que $|B^F| \geq |B^*|$, ou seja, que pelo menos a mesma quantidade de itens de B é empacotada nesses recipientes pela solução ótima e pela solução do FFD.

Seja $B_0^F = \emptyset$, e $B_j^F = \{b \in B^F | b \geq b_j\}$ para $1 \leq j \leq k$. Nossa hipótese de indução é $|B_j^F| \geq j$. A hipótese vale para $j = 0$, já que $|B_0^F| = 0$.

Como os itens de B^* estão em ordem não crescente, $a_i + b_j \leq 1, 1 \leq i \leq j$. Portanto, se b_j não foi empacotado junto a nenhum item de A pelo FFD, ou seja, $b_j \notin B^F$, então em cada recipiente contendo um item de A já havia um item de B maior que b_j, então $|B_j^F| \geq j$. E se $b_j \in B^F$, então pela hipótese de indução $|B_j^F| \geq |B_{j-1}^F| + 1 \geq j$.

Então quando $j = k^*$, $|B^F| \geq |B^*|$. Portanto não há mais itens de B empacotados em recipientes sem itens de A na solução do FFD do que na solução ótima, e o FFD empacota esses itens de forma ótima (dois por recipiente), ou seja, $FFD(I) = OPT(I)$. □

Lema 12. *Se $I = A \cup B \cup C \cup D \cup E$, e o algoritmo FFD não precisa abrir novos recipientes ao empacotar os itens de C e D, então $FFD(I) \leq \frac{11}{9} OPT(I) + 4$.*

Demonstração. Pelo Lema 11, $FFD(A \cup B) = OPT(A \cup B) \leq OPT(I)$. Se a quantidade de recipientes continua a mesma após empacotar os itens de C e D, então temos que $FFD(A \cup B \cup C \cup D) = FFD(A \cup B) \leq OPT(I)$. Aplicando então o Lema 9 com $m = 5$, temos $FFD(I) \leq \frac{6}{5} OPT(I) + 1 < \frac{11}{9} OPT(I) + 4$. □

O restante da prova da razão de aproximação se baseia em particionar os itens da instância em conjuntos de um ou dois elementos. Seja π qualquer partição desse tipo, e também $\pi_1 = \{x \in \pi_1 | \{x\} \in \pi\}$ e $\pi_2 = \{(x, y) \in \pi_2 | \{x, y\} \in \pi, x > y\}$. A partir disso definimos as seguntes funções de ponderação, considerando apenas itens cujo tamanho pertence a um intervalo $(1/N, 1/2]$:

- $w_1(x) = 1/k$, se $x \in (1/(k+1), 1/k]$, para $2 \leq k \leq N$.
- $w_2(x, y) = \begin{cases} w_1(x) + \frac{k-1}{k} w_1(y), & x \in (1/(k+1), 1/k] \wedge kx + y \leq 1 \\ w_1(x) + w_1(y), & x \in (1/(k+1), 1/k] \wedge kx + y > 1 \end{cases}, \quad 2 \leq k \leq N$
- $w_{12}(\pi) = \sum_{(x,y) \in \pi_2} w_2(x, y) + \sum_{x \in \pi_1} w_1(x)$.
- $W_{FFD}(I) = \min_\pi \{w_{12}(\pi)\}$.

Definimos também uma função $desconto(x, y) = w_1(x) + w_1(y) - w_2(x, y)$. Então separamos os itens em dois conjuntos, P e S. O conjunto P contém, para cada intervalo $(1/(k+1), 1/k]$, com $k \geq 2$, os itens cujo tamanho pertencem a esse intervalo e que são ou o primeiro item colocado em um recipiente pelo FFD, ou foram colocados em um recipiente que até então só continha itens do mesmo intervalo. O conjunto S contém os itens restantes.

Lema 13. *Se I é uma instância contendo apenas itens no intervalo $(1/N, 1/2]$, então $\sum_{x \in P} w_1(x) \geq FFD(I) - \sum_{k=2}^{N-1} (k-1)/k$.*

Demonstração. Se um recipiente R_k contém k itens no intervalo $(1/(k+1), 1/k]$ então $\sum_{x \in R_k} w_1(x) = 1$. Para cada k, há apenas um recipiente R_k que é aberto para empacotar um item desse intervalo e empacota menos de k itens desse intervalo. Portanto, para esse recipientes $1/k \leq \sum_{x \in R_k} w_1(x) \leq 1$. Assim, $\sum_{x \in P} w_1(x)) \geq FFD(I) - \sum_{k=2}^{N-1}(k-1)/k$. □

A partir do Lema 13 e de extensa análise do conjunto π_2 e da função *desconto* derivamos outro lema:

Lema 14. *Se I é uma instância contendo apenas itens no intervalo $(1/N, 1/2]$, então $w_{12}(\pi) \geq FFD(I) - (N-2)$.*

Como a razão de aproximação que queremos provar é maior ou igual a 7/6, usando o Lema 8 podemos definir $N = 7$ e assim considerar apenas itens no intervalo $(1/7, 1/2]$. Portanto, pelo Lema 14 temos que $W_{FFD}(I) \geq FFD(I) - 5$.

Pode-se provar, através de extensa análise de todas as possíveis configurações para cada recipiente, o seguinte teorema:

Teorema 3. *Se I é uma instância contendo apenas itens no intervalo $(1/N, 1/2]$, então $FFD(I) \leq \frac{71}{60} OPT(I) + 5$.*

A função de ponderação pode então ser modificada para incluir itens maiores que 1/2, resultando na razão de aproximação a seguir:

Teorema 4. *(D. S. Johnson, 1973 [6]) Para qualquer instância I, $FFD(I) \leq \frac{11}{9} OPT(I) + 4$.*

6 APTAS - La Vega e Lueker

Nesse algoritmo, primeiro consideramos apenas os itens maiores que $\varepsilon/2$ em uma instância I do problema, e aplicamos neles um esquema de agrupamento linear da seguinte forma: sendo k um parâmetro a ser definido, o primeiro grupo G_1 contém os k maiores itens, o segundo grupo G_2 contém os próximos k maiores, e assim por diante, até o último grupo G_m,

que conterá os h menores itens da entrada, sendo $h \leq k$.

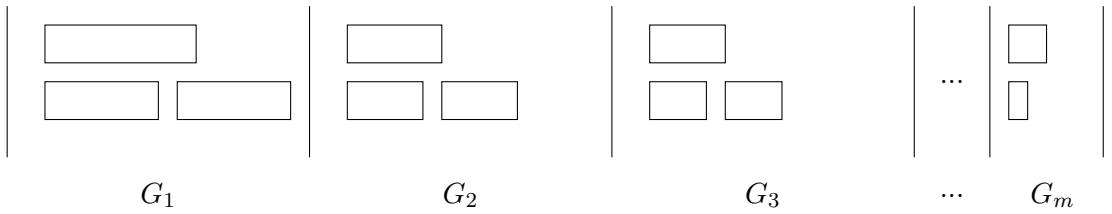

Figura 6: Ilustração de um agrupamento linear com $k = 3$.

A partir desse agrupamento, construimos uma nova entrada I' descartando o primeiro grupo (ou seja, os k maiores itens) e arredondando os tamanhos de todos os itens dos outros grupos para o tamanho do maior item desse grupo. Assim, existem no máximo n/k tamanhos distintos em I', e como todos os itens são maiores que $\varepsilon/2$, cada recipiente pode empacotar menos que $2/\varepsilon$ itens.

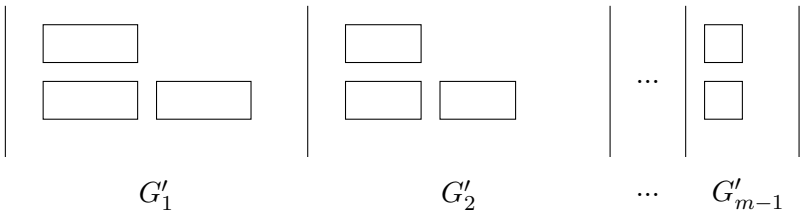

Figura 7: Exemplo de instância I' construída a partir do agrupamento mostrado na figura anterior.

Assim, como o número de tamanhos distintos e a quantidade máxima de itens em cada recipiente é constante, I' pode ser empacotado por um algoritmo de programação dinâmica, descrevendo com um vetor n/k-dimensional a quantidade de itens de cada tamanho distinto colocados em um determinado recipiente B, e então utilizando a seguinte recorrência:

$$OPT(I) = 1 + \min_{B \in C} OPT(I - B)$$

onde C é o conjunto de todos os recipientes possíveis com os itens da instância.

Note que, como cada recipiente só pode empacotar menos que $2/\varepsilon$ itens, existem menos de $(\frac{2}{\varepsilon}+1)^{n/k}$ configurações possíveis para o vetor n/k-dimensional. Portanto esse algoritmo é de tempo polinomial contanto que n/k seja limitado por algum valor constante, o que faremos escolhendo k como $\lfloor \varepsilon SIZE(I) \rfloor$, e assumindo que $\lfloor \varepsilon SIZE(I) \rfloor > 1$, portanto $\lfloor \varepsilon SIZE(I) \rfloor \geq \varepsilon SIZE(I)/2$. Como consideramos apenas os itens maiores que $\varepsilon/2$, $SIZE(I) > n\varepsilon/2$, e logo $n/k \leq 2n/\varepsilon SIZE(I) \leq 4/\varepsilon^2$.

Podemos assumir $\lfloor \varepsilon SIZE(I) \rfloor > 1$ pois do contrário $SIZE(I) \leq 1/\varepsilon$, então haveria no máximo $(1/\varepsilon)/(\varepsilon/2) = 2/\varepsilon^2$ itens grandes na instância e ela poderia ser resolvida de forma ótima pela programação dinâmica sem precisar do agrupamento linear.

A partir desse empacotamento de I', deriva-se um empacotamento para I adicionando-se no máximo k novos recipientes. Por último, usamos *First Fit* para empacotar os itens menores ou iguais a $\varepsilon/2$.

LaVegaELueker

1 Faça agrupamento linear dos itens maiores que $\varepsilon/2$ e cria instância I'
2 Empacote I' por programação dinâmica
3 Derive empacotamento para itens não descartados de I
4 Empacote k maiores itens em no máximo k recipientes
5 Empacote itens menores que $\varepsilon/2$ com *First Fit*

Lema 15. *Se I' é a entrada obtida a partir de uma aplicação de esquema de agrupamento linear à uma entrada I com grupos de tamanho k, então $OPT(I') \leq OPT(I) \leq OPT(I') + k$.*

Demonstração. A primeira desigualdade é provada mostrando-se que todo empacotamento de I pode ser transformado em um empacotamento de I' sem aumentar o número de recipientes. Sejam $G_1, \ldots, G_m$ os grupos de I, e $G'_1, \ldots, G'_{m-1}$ os grupos arredondados de I'. Para todo $i = 1, \ldots, m-1$, o grupo G'_i tem uma quantidade de itens menor ou igual que a quantidade de itens em G_i. Além disso, pela forma que o agrupamento foi feito (pegando os maiores itens primeiro) e por I' ter descartado os k maiores itens, todos os itens em G'_i

tem tamanho menor ou igual aos itens de G_i, portanto dado um empacotamento para I, os itens em G'_i podem ser empacotados na mesma quantidade de recipientes que os itens em G_i foram empacotados. Então $OPT(I') \leq OPT(I)$.

Para provar a segunda desigualdade fazemos o oposto, ou seja, transformamos um empacotamento de I' em um empacotamento de I, adicionando no máximo k recipientes a mais. Pela forma como foi feito o arredondamento dos itens de I', para $i = 1, \ldots, m-1$ todos os itens em G'_i tem o mesmo tamanho do maior item em G_{i+1}, e além disso, G'_i e G_{i+1} tem a mesma quantidade de itens. Portanto dado um empacotamento para I', os itens em G_{i+1} podem ser empacotados na mesma quantidade de recipientes que os itens em G'_i foram empacotados. Resta apenas empacotar então os itens em G_1. No pior caso, será necessário um novo recipiente para cada item em G_1, e G_1 possui k itens, portanto $OPT(I) \leq OPT(I') + k$. □

Lema 16. *Um empacotamento dos itens maiores que γ em l recipientes pode ser expandido para um empacotamento de toda a entrada em* $\max\{l, \frac{1}{1-\gamma} SIZE(I) + 1\}$ *recipientes.*

Demonstração. Empacotamos os itens maiores que γ nos l recipientes, e então aplicamos o algoritmo *First Fit* nos itens menores. Se ao fim o FF não abriu nenhum recipiente novo, então temos l recipientes abertos e todos os itens empacotados. Senão, todos os recipientes exceto o último foram incapazes de receber uma peça pequena, ou seja, estão cheios no mínimo até $1 - \gamma$ de seu espaço. Então, se $m+1$ é o número de recipientes usados, $m(1-\gamma) \leq SIZE(I)$, e portanto $m + 1 \leq \frac{SIZE(I)}{1-\gamma} + 1$. □

Com esses lemas podemos provar que o algoritmo é um APTAS para o Problema do Empacotamento.

Teorema 5. *(F. de La Vega e G. S. Lueker, 1981 [7]) Para qualquer $\varepsilon > 0$, a solução do algoritmo utiliza no máximo $(1+\varepsilon)OPT(I) + 1$ recipientes.*

Demonstração. Segundo o lema 15, se l é a quantidade de recipientes usada no empacota-

mento derivado do agrupamento linear, então

$$\begin{aligned} l &\leq OPT(I') + k \\ &\leq OPT(I) + k \\ &= OPT(I) + \lfloor \varepsilon SIZE(I) \rfloor \\ &\leq OPT(I) + \varepsilon SIZE(I) \\ &\leq (1+\varepsilon) OPT(I). \end{aligned}$$

Após a aplicação do *First Fit* nos itens menores, o total de recipientes usado é, segundo o lema 16, $\max\{l, (1+\varepsilon)OPT(I) + 1\}$. □

7 APTAS - Karmarkar e Karp

Sejam $s_1, \ldots, s_m$ os tamanhos distintos dos itens de uma instância, e b_i a quantidade de itens de tamanho s_i, $i = 1, \ldots, m$. Representamos a quantidade de itens de tamanho s_i incluídas em um recipiente de uma solução por t_i. Dessa forma, o conteúdo de um recipiente pode ser descrito como uma *configuração* $t_1, \ldots, t_m$ tal que $\sum_{i=1}^{m} t_i s_i \leq 1$.

Denotamos todas as configurações possíveis para um recipiente como $T_1, \ldots, T_N$, t_{ij} a quantidade de itens de tamanho s_i na configuração T_j, e x_j o número de recipientes seguindo a configuração T_j, $j = 1, \ldots, N$. Assim, o Problema do Empacotamento pode ser representado pelo seguinte programa linear inteiro, conhecido como *programa linear de configuração*, introduzido por Gilmore e Gomory [8]:

$$\begin{aligned} \text{minimizar} \quad & \sum_{j=1}^{N} x_j \\ \text{sujeito a} \quad & \sum_{j=1}^{N} t_{ij} x_j \geq b_i, \qquad i = 1, \ldots, m \\ & x_j \in \mathbb{N}, \qquad j = 1, \ldots, N \end{aligned}$$

Nesse algoritmo, itens da instância I são ordenados em ordem não-crescente e agrupados de acordo com um esquema de agrupamento harmônico, onde cada grupo recebe itens até seu tamanho total ser maior que 2. Seja r a quantidade de grupos formados, e n_i a quantidade de itens em cada grupo G_i, $i = 1, \ldots, r$. Como os itens foram agrupados em ordem não-crescente, qualquer item do grupo i é maior ou igual a qualquer item do grupo $i+1$, de onde concluímos que $n_i \leq n_{i+1}$, $i = 2, \ldots, r-1$.

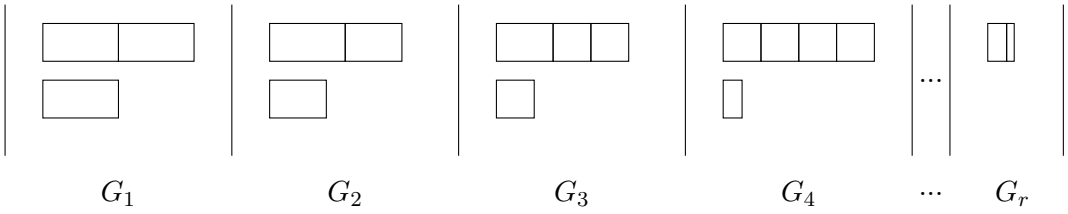

Figura 8: Ilustração de um agrupamento harmônico. A primeira linha de itens em cada grupo exceto G_r tem tamanho total no máximo 2.

A partir desse agrupamento, construimos uma nova entrada I' descartando G_1, G_r, e os $n_i - n_{i-1}$ menores itens dos grupos G_i restantes ($i = 2, \ldots, r-1$), e arredondando os tamanhos dos itens de cada grupo para o valor do maior item do grupo. Ao fim, cada grupo de I' contém a mesma quantidade de itens que seu antecessor continha em I. E como os itens dos grupos foram arredondados para o maior tamanho, qualquer empacotamento da instância I' pode ser transformado em um empacotamento dos itens não descartados de I.

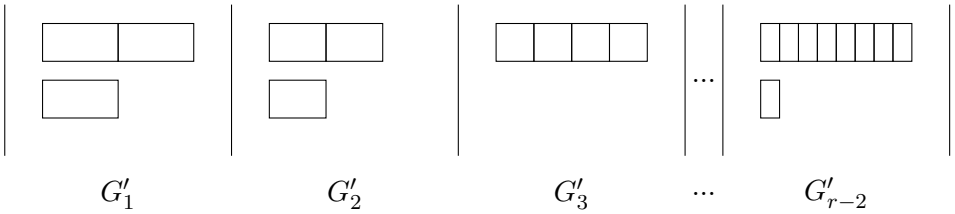

Figura 9: Exemplo de instância I' construída a partir do agrupamento mostrado na figura anterior.

Empacotamos os itens descartados com *First-Fit*, e então resolvemos o programa linear para a instância I'. Sendo x a solução ótima encontrada, para i de 1 a N empacotamos $\lfloor x_i \rfloor$ recipientes seguindo a configuração T_i. Esse arredondamento da solução linear pode deixar alguns itens de fora, e esses são então empacotados recursivamente pelo algoritmo, parando apenas quando o número de itens for menor que alguma constante pré-definida.

KarmarkarEKarp(I)

```
1  se SIZE(I) < CONSTANTE
2      Empacote peças com First Fit
3  senão
4      Faça agrupamento harmônico e cria instância I'
5      Empacote itens descartados com First Fit
6      Ache solução ótima x para PL de configuração
7      para i = 1 ate N
8          Empacote ⌊x_i⌋ recipientes seguindo a configuração T_i
9      I_2 = itens restantes
10     KarmarkarEKarp(I_2)
```

Lema 17. *O tamanho total dos itens descartados para a construção da instância I' é $O(\log SIZE(I))$.*

Demonstração. O tamanho da instância é $SIZE(I) = \sum_{i=1}^{m} s_i b_i$. Considerando a restrição $\sum_{j=1}^{N} t_{ij} x_j \geq b_i$, temos que $SIZE(I) \leq OPT_{LP}(I) \leq OPT(I)$, onde $OPT_{LP}(I)$ é o valor da solução ótima do programa linear e $OPT(I)$ é o valor da solução ótima do programa inteiro.

Pela forma que foi construída, a instância I' possui no máximo $SIZE(I)/2$ tamanhos distintos, e o tamanho total de cada grupo é no máximo 3. Cada grupo G_i da instância original I descarta seus $n_i - n_{i-1}$ menores itens. Por serem os menores, seus tamanhos não podem ultrapassar a média de tamanho dos itens do grupo, $3/n_i$. Portanto o tamanho desses itens no total é no máximo $\frac{3}{n_i}(n_i - n_{i-1}) \leq \sum_{j=n_{i-1}+1}^{n_i} 3/j$.

Então, o tamanho total de todos os itens descartados de todos os grupos não pode passar de $\sum_{j=1}^{n_r} 3/j = O(H_{n_r}) = O(\log n_r)$, sendo H_{n_r} o n_r-ésimo número harmônico $1 + 1/2 + \cdots + 1/n_r$.

Usando o Lema 16, podemos ignorar itens menores que um certo γ sem mudar a magnetude do erro. Assumimos então que todos os itens tem no mínimo tamanho $1/SIZE(I)$. Então $n_r \leq \frac{3}{1/SIZE(I)} = 3SIZE(I) = O(SIZE(I))$.

Portanto o tamanho total dos itens descartados é $O(\log SIZE(I))$. □

Sejam I_1 os itens empacotados a partir da solução arredondada do PL na primeira chamada do algoritmo, e I_2 os itens empacotados recursivamente.

Lema 18. *Para qualquer instância I para a qual o esquema de agrupamento harmônico produz outra instância I', que é então dividida em I_1 e I_2, valem as desigualdades $OPT_{LP}(I_1) + OPT_{LP}(I_2) \leq OPT_{LP}(I') \leq OPT(I)$.*

Demonstração. Para cada item em I' existe um item correspondente em I de tamanho maior ou igual. Portanto um empacotamento de I pode ser transformado em um empacotamento de I' (qualquer item de I que não tem correspondência em I' é simplesmente ignorado). Então $OPT(I)$ é um limite superior para $OPT(I')$. Similarmente, toda configuração para um recipiente em I induz uma configuração de recipiente em I', portanto $OPT_{LP}(I') \leq OPT(I)$.

Pela forma como I_1 e I_2 foram construidas, se x é a solução ótima do programa linear para a instância I', então $\lfloor x_j \rfloor$, $j = 1, \ldots, N$ é uma solução viável do PL para I_1 e $x_j - \lfloor x_j \rfloor$, $j = 1, \ldots, N$ é uma solução viável do PL para I_2. Portanto a soma dos valores das soluções ótimas para esses dois PLs é limitada superiormente por $\sum_{j=1}^{N} \lfloor x_j \rfloor + \sum_{j=1}^{N} x_j - \lfloor x_j \rfloor = \sum_{j=1}^{N} x_j = OPT_{LP}(I')$. □

Teorema 6. *(Karmarkar e Karp, 1982 [9]) Para qualquer instância I do Problema do Empacotamento, a solução do algoritmo usa ao todo não mais que $OPT(I) + O(\log^2 SIZE(I))$ recipientes.*

Demonstração. Segundo o Lema 18, o valor da solução do empacotamento de I_1 e I_2 não ultrapassa o valor ótimo da solução de I, então o único erro introduzido no algoritmo vem do empacotamento dos itens que não pertencem nem a I_1 nem a I_2, ou seja, os itens descartados durante o agrupamento. Esse erro é introduzido a cada nível de recursão, portanto precisamos achar um limitante para o número de chamadas recursivas.

O tamanho da primeira instância a ser empacotada recursivamente, $SIZE(I_2)$, é limitado por $\sum_{j=1}^{N} x_j - \lfloor x_j \rfloor$. Essa soma pode ser limitada pelo número de restrições do PL para a instância I', ou seja, o número de tamanhos distintos de itens nessa instância. Como a instância I' é composta de grupos de itens de mesmo tamanho cujos tamanhos totais eram maiores que 2 mesmo antes de ser arredondados, o número de tamanhos distintos é no máximo $SIZE(I)/2$.

Portanto, $SIZE(I_2) \leq SIZE(I)/2$, ou seja, o tamanho da instância original cai pelo menos pela metade a cada recursão, o que leva a um número $O(\log SIZE(I))$ recursões no máximo, cada uma empacotando $O(\log SIZE(I))$ itens descartados, de acordo com o Lema 17. No pior caso (cada item ocupa sozinho um recipiente), são gastos $O(\log SIZE(I))$ recipientes a mais para esses itens em cada recursão, resultando em um erro aditivo total de $O(\log^2 SIZE(I))$ recipientes.

Como $SIZE(I) \leq OPT(I)$, o algoritmo gasta ao todo $OPT(I) + O(\log^2 SIZE(I))$ recipientes no máximo. □

8 Agradecimentos

Agradecemos à Universidade Estadual de Campinas (UNICAMP) pela bolsa de iniciação científica concedida através do Programa Institucional de Bolsas de Iniciação Científica (PIBIC). O projeto também teve apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) por meio dos processos de números #308689/2017-8 e #425340/2016-3, e da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), por meio dos processos de número #2015/11937-9 e #2016/23552-7.

Referências

[1] SIMCHI-LEVI, D. New worst-case results for the bin-packing problem. In: . c2006.

[2] YAO, A. C.-C. New algorithms for bin packing. *J. ACM*, New York, NY, USA, v. 27, n. 2, p. 207–227, Apr. 1980.

[3] LEE, C. C.; LEE, D. T. A simple on-line bin-packing algorithm. *J. ACM*, New York, NY, USA, v. 32, n. 3, p. 562–572, July 1985.

[4] JOHNSON, D. S. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, v. 8, n. 3, p. 272 – 314, 1974.

[5] JOHNSON, D. S.; ULLMAN, J. D.; GAREYI, M. R.; GRAHAMII, R. L. Worst-case performance bounds for simple one-dimensional packing algorithms*.

[6] JOHNSON, D. S. *Near-optimal bin packing algorithms*. 1973. Tese (Doutorado em Física) - Massachusetts Institute of Technology, 1973.

[7] FERNANDEZ DE LA VEGA, W.; LUEKER, G. S. Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica*, v. 1, n. 4, p. 349–355, Dec 1981.

[8] C E GILMORE, R.; GOMORY, R. A linear programming approach to the cutting stock problem i. *Oper Res*, v. 9, 01 1961.

[9] KARMARKAR, N.; KARP, R. M. An efficient approximation scheme for the one-dimensional bin-packing problem. In: . SFCS '82. Washington, DC, USA: IEEE Computer Society, c1982. p. 312–320.