Investigação Operacional - Trabalho Prático 1

Rodrigo Monteiro, Diogo Abreu, Miguel Gramoso, and Miguel Marinha

Universidade do Minho, 4710-057 Braga, Portugal
e-mail: {a100706,a100646,a100835,a100651}@alunos.uminho.pt

Abstract. Neste estudo é determinada uma solução ótima relativamente a um problema de empacotamento de um conjunto de itens em contentores. De acordo com o maior número de estudante do grupo e uma fórmula devidamente apresentada posteriormente, foram determinados os comprimentos e quantidade de itens e contentores.
Com base nisso e na medida de eficiência estabelecida, foram definidos os parâmetros e restrições, sendo o objetivo de minimizar a soma do comprimento dos contentores utilizados, conforme esses dados estabelecidos. Com recurso ao LPSolve, e implicitamente ao método Simplex, obteve-se uma solução ótima válida.

Keywords: Linear Programming· Bin Packing Problem · LPSolve

Table of Contents

List of Tables

List of Figures

1 Formulação do Problema

Neste problema, pretende-se determinar como empacotar um conjunto de itens em contentores, de modo a obter uma solução ótima com base numa medida de eficiência: a soma dos comprimentos dos contentores que são utilizados. Esta classe de problema é geralmente referida como um problema de empacotamento em uma dimensão/*1D bin packing problem*.

Assim, considera-se uma solução admissível, se todos os itens forem atribuídos a contentores sem exceder a capacidade de nenhum deles, tendo em conta que são utilizados contentores de diferentes capacidades.

De acordo com o maior número de inscrição do grupo (100835), obtemos os seguintes dados relativos ao comprimento e à quantidade dos itens e dos contentores:

$$
\begin{aligned}
&A = 0\\
&B = 0, \text{par} \Rightarrow k_1 = 0\\
&C = 8, \text{par} \Rightarrow k_2 = C + 2 \iff k_2 = 8 + 2 = 10\\
&D = 3, \text{ímpar} \Rightarrow k_3 = 10\\
&E = 5, \text{ímpar} \Rightarrow k_4 = E + 8 \iff k_4 = 5 + 8 = 13
\end{aligned}
$$

Soma dos comprimentos dos itens $= 0 \times 1 + 10 \times 2 + 10 \times 3 + 13 \times 4 + 5 \times 5 = 127$

Comprimento	Quantidade
1	$k_1 = 0$
2	$k_2 = 10$
3	$k_3 = 10$
4	$k_4 = 13$
5	$k_5 = 5$

Table 1. Itens

Comprimento	Quantidade disponível
11	ilimitada
10	B+1 = 1
7	D+1 = 4

Table 2. Contentores

Depois de uma análise do problema, reparamos na semelhança a um problema de corte (*cutting stock problem*) e, portanto, determinamos que as variáveis de decisão necessárias consistem nas diversas possíveis combinações/*padrões* de itens que podem ser empacotados nos contentores de três comprimentos diferentes (11, 10 e 7), sendo que a ordem dos itens é irrelevante, que podem haver combinações com um ou mais itens do mesmo comprimento, e que, tal como se referiu antes, não podem exceder a capacidade do respetivo contentor.

O modelo utilizado para cumprir o objetivo e a medida de eficiência terá, então, de restringir a quantidade usada de certos itens e contentores, tendo como base os padrões de empacotamento de itens para as impor essas restrições.

2 Modelo

2.1 Variáveis de decisão

As variáveis x_i representam combinações de itens para os contentores de comprimento 11 (com somas menores ou iguais a 11). Ou seja, qualquer conjunto S tal que $S = [a_1, a_2, ...a_n]$ em que para qualquer $1 \leq i \leq n$, $a_i \in [2, 3, 4, 5]$ e $\sum_{i=1}^{n} a_i \leq 11$:

- $x_1 : 2$
- $x_2 : 3$
- $x_3 : 4$
- $x_4 : 5$
- $x_5 : 2 + 2$
- $x_6 : 2 + 3$
- $x_7 : 2 + 4$
- $x_8 : 2 + 5$
- $x_9 : 3 + 3$
- $x_{10} : 3 + 4$
- $x_{11} : 3 + 5$
- $x_{12} : 4 + 4$
- $x_{13} : 4 + 5$
- $x_{14} : 5 + 5$
- $x_{15} : 2 + 2 + 2$
- $x_{16} : 2 + 2 + 3$
- $x_{17} : 2 + 2 + 4$
- $x_{18} : 2 + 2 + 5$
- $x_{19} : 2 + 3 + 3$
- $x_{20} : 2 + 3 + 4$
- $x_{21} : 2 + 3 + 5$
- $x_{22} : 2 + 4 + 4$
- $x_{23} : 2 + 4 + 5$
- $x_{24} : 3 + 3 + 3$
- $x_{25} : 3 + 3 + 4$
- $x_{26} : 3 + 3 + 5$
- $x_{27} : 3 + 4 + 4$
- $x_{28} : 2 + 2 + 2 + 2$
- $x_{29} : 2 + 2 + 2 + 3$
- $x_{30} : 2 + 2 + 2 + 4$
- $x_{31} : 2 + 2 + 2 + 5$
- $x_{32} : 2 + 2 + 3 + 3$
- $x_{33} : 2 + 2 + 3 + 4$
- $x_{34} : 2 + 3 + 3 + 3$
- $x_{35} : 2 + 2 + 2 + 2 + 2$
- $x_{36} : 2 + 2 + 2 + 2 + 3$

As variáveis y_i representam combinações de itens para os contentores de comprimento 10 (com somas menores ou iguais a 10). Ou seja, qualquer conjunto S tal que $S = [a_1, a_2, ...a_n]$ em que para qualquer $1 \leq i \leq n$, $a_i \in [2, 3, 4, 5]$ e $\sum_{i=1}^{n} a_i \leq 10$:

- $y1$: 2
- $y2$: 3
- $y3$: 4
- $y4$: 5
- $y5$: 2+2
- $y6$: 2+3
- $y7$: 2+4
- $y8$: 2+5
- $y9$: 3+3
- $y10$: 3+4
- $y11$: 3+5
- $y12$: 4+4
- $y13$: 4+5
- $y14$: 5+5
- $y15$: 2+2+2
- $y16$: 2+2+3
- $y17$: 2+2+4
- $y18$: 2+2+5
- $y19$: 2+3+3
- $y20$: 2+3+4
- $y21$: 2+3+5
- $y22$: 2+4+4
- $y23$: 3+3+3
- $y24$: 3+3+4
- $y25$: 2+2+2+2
- $y26$: 2+2+2+3
- $y27$: 2+2+2+4
- $y28$: 2+2+3+3
- $y29$: 2+2+2+2+2

As variáveis z_i representam combinações de itens para os contentores de comprimento 7 (com somas menores ou iguais a 7). Ou seja, qualquer conjunto S tal que $S = [a_1, a_2, ...a_n]$ em que para qualquer $1 \leq i \leq n$, $a_i \in [2, 3, 4, 5]$ e $\sum_{i=1}^{n} a_i \leq 7$:

- $z_1 : 2$
- $z_2 : 3$
- $z_3 : 4$
- $z_4 : 5$
- $z_5 : 2 + 2$
- $z_6 : 2 + 3$
- $z_7 : 2 + 4$
- $z_8 : 2 + 5$
- $z_9 : 3 + 3$
- $z_{10} : 3 + 4$
- $z_{11} : 2 + 2 + 2$
- $z_{12} : 2 + 2 + 3$

Naturalmente, os conjuntos utilizados numa solução ótima serão os mais *eficientes*, ou seja, aqueles com menos desperdício de espaço.

2.2 Parâmetros

Os parâmetros estão apresentados nas tabelas 1 e 2. Existe um número ilimitado de contentores de comprimento 11 (`bin_x`), 1 contentor de comprimento 10 (`bin_y`), e 4 contentores de comprimento 7 (`bin_z`). Além disso, existem 10 itens de comprimento 2, 10 itens de comprimento 3, 13 itens de comprimento 4 e 5 itens de comprimento 5.

2.3 Função Objetivo

O objetivo é minimizar a soma do comprimento dos contentores utilizados.
$\min_z = 11 \times bin\_x + 10 \times bin\_y + 7 \times bin\_z$

2.4 Restrições

Primeiramente, para garantir que não é excedida a quantidade disponível de cada contentor, são impostas restrições, um limite superior, nos valores de `bin_y` e de `bin_z` — não é necessário impor um limite em `bin_x` visto que há uma quantidade ilimitada de contentores de comprimento 11. O valor de `bin_x` corresponde a $\sum_{i=1}^{36} x_i$, ou seja, corresponde à soma das quantidades usadas de padrões de empacotamento relativos aos contentores de comprimento 11. A mesma lógica é utilizada para `bin_y` e para `bin_z`, portanto, $bin\_y = \sum_{i=1}^{29} y_i$ e $bin\_z = \sum_{i=1}^{12} z_i$.

Para além disso, também é necessário impor restrições na quantidade de itens. De modo a obter um modelo mais inteligível foram criadas, por exemplo, as variáveis `x_itens2, y_itens2` e `z_itens2`, que correspondem à quantidade de itens de comprimento 2 utilizada em contentores de tamanho 11, 10 e 7, respetivamente. Assim, a restrição fica sobre a soma dessas variáveis:

```
x_itens2 + y_itens2 + z_itens2 >= 10;
x_itens3 + y_itens3 + z_itens3 >= 10;
x_itens4 + y_itens4 + z_itens4 >= 13;
x_itens5 + y_itens5 + z_itens5 >= 5;
```

Os valores de `[x_itens2, ..., z_itens5]` são obtidos através da soma do número de variáveis/padrões utilizados relativos a um determinado contentor multiplicados pela quantidade de um dado item necessário para esse padrão. Por exemplo, `z_itens2 = z1 + 2 z5 + z6 + z7 + z8 + 3 z11 + 2 z12`, ou seja, o número de itens de comprimento 2 utilizado em contentores de comprimento 7 é igual à soma do número de padrões `z1` utilizados, visto que este contém um item de comprimento 2, com o número de padrões `z5` utilizados multiplicado por 2, visto que este padrão contém dois itens de comprimento 2, etc.

2.5 Ficheiro de Input

```
/* Função Objetivo */
min: 11 bin_x + 10 bin_y + 7 bin_z;

/* Restrições a itens */
x_itens2 + y_itens2 + z_itens2 >= 10;
x_itens3 + y_itens3 + z_itens3 >= 10;
x_itens4 + y_itens4 + z_itens4 >= 13;
x_itens5 + y_itens5 + z_itens5 >= 5;

x_itens2 = x1 + 2 x5 + x6 + x7 + x8 + 3 x15 + 2 x16 + 2 x17 + 2 x18 + x19 + x20 + x21 + x22 + x23 + 4
x28 + 3 x29 + 3 x30 + 3 x31 + 2 x32 + 2 x33 + x34 + 5 x35 + 4 x36;
y_itens2 = y1 + 2 y5 + y6 + y7 + y8 + 3 y15 + 2 y16 + 2 y17 + 2 y18 + y19 + y20 + y21 + y22 + 4 y25 + 3
y26 + 3 y27 + 2 y28 + 5 y29;
z_itens2 = z1 + 2 z5 + z6 + z7 + z8 + 3 z11 + 2 z12;

x_itens3 = x2 + x6 + 2 x9 + x10 + x11 + x16 + 2 x19 + x20 + x21 + 3 x24 + 2 x25 + 2 x26 + x27 + x29 + 2
x32 + x33 + 3 x34 + x36;
y_itens3 = y2 + y6 + 2 y9 + y10 + y11 + y16 + 2 y19 + y20 + y21 + 3 y23 + 2 y24 + y26 + 2 y28;
z_itens3 = z2 + z6 + 2 z9 + z10 + z12;

x_itens4 = x3 + x7 + x10 + 2 x12 + x13 + x17 + x20 + 2 x22 + x23 + x25 + 2 x27 + x30 + x33;
y_itens4 = y3 + y7 + y10 + 2 y12 + y13 + y17 + y20 + 2 y22 + y24 + y27;
z_itens4 = z3 + z7 + z10;

x_itens5 = x4 + x8 + x11 + x13 + 2 x14 + x18 + x21 + x23 + x26 + x31;
y_itens5 = y4 + y8 + y11 + y13 + 2 y14 + y18 + y21;
z_itens5 = z4 + z8;

/* Restrições a contentores */
bin_x = x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + x17 +
x18 + x19 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29 + x30 + x31 + x32 + x33 + x34 +
x35 + x36;

bin_y = y1 + y2 + y3 + y4 + y5 + y6 + y7 + y8 + y9 + y10 + y11 + y12 + y13 + y14 + y15 + y16 + y17 +
y18 + y19 + y20 + y21 + y22 + y23 + y24 + y25 + y26 + y27 + y28 + y29;

bin_z = z1 + z2 + z3 + z4 + z5 + z6 + z7 + z8 + z9 + z10 + z11 + z12;

bin_x >= 0;
bin_y <= 1;
bin_z <= 4;

/* Restrições do valor das variáveis a inteiros */

int x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, x17, x18, x19, x20, x21,
x22, x23, x24, x25, x26, x27, x28, x29, x30, x31, x32, x33, x34, x35, x36;
int y1, y2, y3, y4, y5, y6, y7, y8, y9, y10, y11, y12, y13, y14, y15, y16, y17, y18, y19, y20, y21,
y22, y23, y24, y25, y26, y27, y28, y29;
int z1, z2, z3, z4, z5, z6, z7, z8, z9, z10, z11, z12;
```

Fig. 1. Ficheiro de input (.lp)

3 Solução Ótima

A solução ótima, para este problema, é aquela em que se verifica o menor espaço necessário para empacotar os items em causa nos contentores. Para reduzir o espaço total necessário devem ter prioridade as soluções que reduzem ao máximo ou eliminem completamente espaço livre no interior dos contentores.

Itens	Tamanho	Quantidade	Total per Item
itens1	1	0	0
itens2	2	10	20
itens3	3	10	30
itens4	4	13	52
itens5	5	5	25
Total:		38	127

Fig. 2. Tamanho dos items

De todas as combinações possíveis para empacotar os itens, queremos o conjunto de combinações cujo tamanho total seja o mais próximo possível de 127, e se possível, que tenham tamanho total igual a 127. Através do LPSolve obtemos 3 soluções. A solução A é a solução ótima produzida com as restrições e parâmetros definidos e discutidos anteriormente. As soluções B e C foram obtidas modificando essas restrições, de modo a verificar a variação da solução: a solução B não utiliza contentores de comprimento 7 e a solução C não utiliza contentores de comprimento 10 (uma solução que não utiliza contentores de comprimento 11 é impossível sem a modificação dos parâmetros).

	A	B	C
bin 11	10	11	9
bin 10	1	1	0
bin 7	1	0	4
Total	127	131	127

Fig. 3. Soluções A, B e C

```
Value of objective function: 127.0

Actual values of the variables:
bin_x                          10
bin_y                           1
bin_z                           1
x_itens2                        9
y_itens2                        1
x_itens3                        9
z_itens3                        1
x_itens4                       10
y_itens4                        2
z_itens4                        1
x_itens5                        5
x23                             5
x27                             2
x33                             1
x34                             2
y22                             1
z10                             1
```

Fig. 4. Output A reduzido do LPSolve

Ao comparar as três soluções podemos excluir a solução B, tendo em causa que as restantes duas possuem um tamanho inferior, de valor 127.

A diferença nas duas soluções remete-se ao total de contentores necessários, sendo a solução A a que utiliza menos contentores, 12 contentores contra os 13 da solução C. Por esse motivo pode-se considerar a solução A uma melhor solução, apesar de ambas respeitarem a medida de eficiência estabelecida.

```
Branch & Bound depth: 13
Nodes processed: 44
Simplex pivots: 114
Number of equal solutions: 1

Relaxed solution    127 after   24 iter is B&B base.

Feasible solution   142 after   36 iter,    6 nodes  (gap 11.7%)
Improved solution   138 after   37 iter,    7 nodes  (gap 8.6%)
Improved solution   134 after   82 iter,    29 nodes (gap 5.5%)
Improved solution   131 after   93 iter,    34 nodes (gap 3.1%)
Improved solution   127 after   114 iter,   44 nodes (gap 0.0%)

Optimal solution    127 after   114 iter,   44 nodes (gap 0.0%).
```

Fig. 5. Informação do output A com as flags -v e -t

4 Validação do Modelo

A validação do modelo é feita com o intuito de comprovar que o nosso programa, neste caso as variáveis e restrições impostas, de facto fazem o que é suposto e que desta forma é-se chegado ao resultado pretendido.
Em projetos em que se tenta simular algum aspeto da vida real a validação de modelos é necessária e crítica para se ter a garantia de que o programa faz o que se pretende, pois tal como em todas as simulações, é necessário abordar o problema em causa de uma maneira inteligente e sistemática, usando todas as ferramentas ao nosso dispor para que o produto final seja o melhor possível.
No nosso caso, porém, não estamos a tentar aproximar necessariamente um ambiente real, mas sim chegar a uma solução a partir de parâmetros conhecidos e limitações absolutas, os quais já foram extensivamente abordados no capítulo 1 e 2.
Desta maneira, obtivemos várias soluções (ver capítulo 3). A figura abaixo mostra uma das soluções geradas pelo LPSolve a partir do modelo usado, que de facto obedece às restrições impostas no capítulo 2.4.

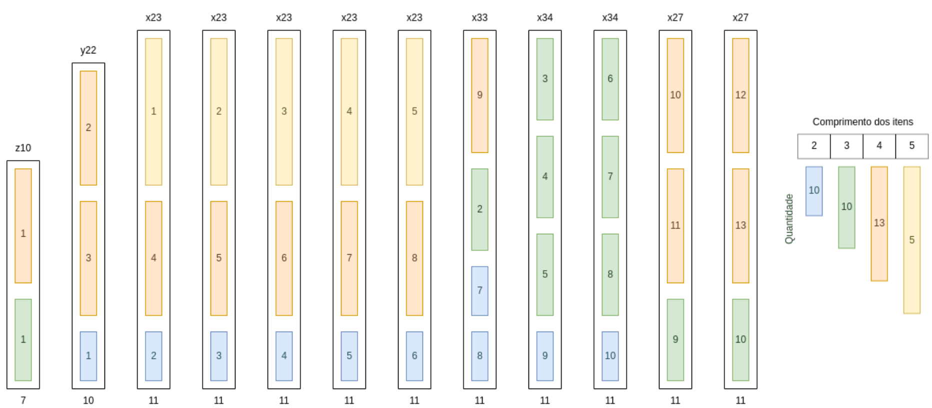

Fig. 6. Demonstração da solução do problema

Tal como mostra a figura 6, o modelo empacota 10 itens de comprimento 2, 10 itens de comprimento 3, 13 itens de comprimento 4, e 5 caixas de comprimento 5 usando 1 contentor de comprimento 7 (máximo 4), 1 contentor de comprimento 10 (máximo 1), e 10 contentores de comprimento 11 (ilimitados).

References

1. Verification and validation of computer simulation models: https://en.wikipedia.org/wiki/Verification_and_validation_of_computer_simulation_models. Acessado em Março, 2023
2. lp_solve 5.5.2.5 documentation https://web.mit.edu/lpsolve/doc/. Acessado em Março, 2023