Universidade do Minho
Licenciatura em Engenharia Informática

Investigação Operacional

Trabalho 1

Março 2024

a97646 Pedro Silva Ferreira
a98352 Enzo Gabriel Barros Vieira
a100480 Nuno Alberto Gonçalves Aguiar
a100549 Luís Carlos Fragoso Figueiredo
a100656 Gustavo Manuel Marinho Barros

Introdução

Com este relatório, acompanha-se o desenvolvimento do 1º projeto da UC de Investigação Operacional proposto no ano letivo de 2023/24, relativo à aplicação dum modelo *one-cut*, de autoria de Harald Dyckhoff, a um problema de empacotamento.

Pergunta 0: Dados Utilizados

Maior número de aluno do grupo: 100656
(xABCDE)
A = 0, B = 0, C = 6, D = 5, E = 6

contentores	
comprimento	**quantidade**
11	ilimitada
10	1
7	6

itens	
comprimento	**quantidade**
1	0
2	8
3	10
4	8
5	5

Tabelas 1 e 2 - Comprimento e quantidade de contentores e de itens necessários, respetivamente

Soma dos comprimentos dos itens a empacotar:
0*1 + 8*2 + 10*3 + 8*4 + 5*5 = 103

Pergunta 1: Formulação do Problema

O problema proposto é um problema de empacotamento de itens de vários tamanhos em contentores também de diferentes. Como descrito na tabela 1, cada contentor de um certo tamanho tem uma quantidade limitada associada, com a exceção dos contentores de comprimento 11, que podem ser usados tantos quantos forem necessários. Cada dimensão de item tem também uma quantidade limitada descrita na tabela 2. É obrigatório garantir a atribuição de todos os itens a contentores, não sendo possível exceder nem a capacidade, nem a quantidade disponível dos mesmos. O objetivo do problema é empacotar os itens de modo a minimizar a soma dos comprimentos dos contentores usados para esse efeito, seguindo o modelo de Dyckhoff.

As variáveis de decisão foram feitas e pensadas de acordo com o modelo de Dyckhoff, ou seja o “one-cut model” que no caso baseia-se em, basicamente, cortes recursivos.

Apesar deste se tratar de um problema de empacotamento, é possível resolver através deste modelo, pois os dois problemas são bastante idênticos, sendo até possível pensar no problema de empacotamento como um problema de corte.

Neste caso, as variáveis $y_{l,k}$ significam o número de secções de contentores de tamanho l com um item de comprimento K e um espaço de sobra de dimensão l - K. Ou seja, a variável $y_{11,4}$ significa que dentro de um contentor de 11 se coloca um objeto de tamanho 4. Ao colocar um objeto dentro de um contentor de 11 sobra ainda 7 de espaço, pelo que se tem adicionalmente a variável $y_{7,4}$ que significa colocar no espaço de resíduo 7 da operação anterior (ou num contentor de comprimento 7, visto que medidas standard e residuais não são distinguidas segundo este modelo). A solução do problema irá traduzir-se na quantidade a utilizar de cada variável $y_{l,k}$.

Para resolver o problema devemos minimizar a função linear objetivo do modelo que representa a soma total dos comprimentos dos contentores usados. Esta função tem vários tipos de restrições, explicadas no próximo ponto.

Pergunta 1 (Cont.)

O modelo one-cut tem como base as premissas apresentadas de seguida, que constituem o modus operandi de formulação de problemas de corte/empacotamento no geral. Embora originalmente escritos sob o ponto de vista do corte, podemos adaptá-la à semântica dum problema de empacotamento:

A. O número de operações de empacotamento não é limitado.

B. Cada empacotamento resulta em dois novos espaços em que pelo menos uma tem uma dimensão pedida. (Esta premissa é mais relevante no corte, pois impede que se corte stock em duas medidas não pedidas. No caso aqui seria impensável fazer um empacotamento segundo uma medida que não seja a do item em questão.)

C. Espaço residual dum corte podem ser alvo de subsequente empacotamento, de modo a encaixar mais 1+ itens pedidos.

D. Aos olhos da operação de empacotamento, um contentor vazio com, por exemplo, 7 espaços, não difere dum contentor com 10 espaços mas com 3 já ocupados por um item.

E. Dimensões standard dos contentores estão disponíveis em qualquer quantidade.

F. Todos os itens devem ser empacotados.

G. Os custos de empacotamento estão linearmente dependentes do consumo de contentores. (Irrelevante neste trabalho pois não se atribui custo à operação de empacotar)

H. Quaisquer dos itens a empacotar é menor que o maior contentor em termos de unidades de espaço.

I. O contentor menos espaçoso é maior que o menor item, e não há situações em que um item tenha exatamente a mesma medida dum contentor.

Pergunta 2: Modelo de Programação Linear

- ### Variáveis de Decisão

  $y_{l,k}$ em que:
  $l \in L \{11,10,7\} \cup R \{8, 9, 6, 5, 4, 3\}$
  $k \in K \{5,4,3,2\}$

  $y_{l,k} >= 0$

- ### Parâmetros

  Estão disponíveis um número de contentores de comprimento 11 infinito, 1 contentor de tamanho 10 e 6 contentores de tamanho 7.
  Quer-se armazenar 8 unidades de comprimento 2, 10 unidades de comprimento 3, 8 unidades de comprimento 4 e 5 unidades de comprimento 5.

- ### Função Objetivo

  Trata-se de um problema de minimização, neste caso, de minimizar o número de contentores utilizados de modo a armazenar itens com uma soma de comprimentos igual a 103.

  $$\text{min: } z = 11y_{11,5} + 11y_{11,4} + 11y_{11,3} + 11y_{11,2} + 10y_{10,5} + 10y_{10,4} + 10y_{10,3} + 7yb_{7,5} + 7yb_{7,4}$$

Pergunta 2 (Cont.)

- ## Restrições

Considere-se, por exemplo, que se recorria erradamente a um padrão de empacotamento que envolve um resíduo sem antes existir o empacotamento do item de onde esse resíduo provém. Tal situação é colmatada especificamente com esta bateria de restrições:

```
y11_5 - y6_5         >= 0;
y11_4 - y7_4         >= 0;
y10_4 - y6_4         >= 0;
y11_3 - y8_3         >= 0;
y8_3  - y5_3         >= 0;
y10_3 - y7_4 + y11_4 >= 0;
y7_4  - y4_3         >= 0;
y11_2 - y9_2         >= 0;
y9_2  - y7_2         >= 0;
y5_3  - y3_2 + y11_3 >= 0;
y5_3  - y3_2         >= 0;
y10_2 - y8_2         >= 0;
y8_2  - y6_4 + y10_2 >= 0;
y6_4  - y4_2 + y10_2 >= 0;
yb7_5 - yb5_2        >= 0;
yb5_2 - yb3_2        >= 0;
```

As restrições que impedem que se ultrapasse a quantidade de contentores disponíveis é a soma das variáveis de empacotamentos iniciais:

```
y10_5 + y10_4 +y10_3 +y10_2 <= 1;
yb7_5 + yb7_4 +yb7_3 +yb7_2 <= 6;
/* Sem restrição p/ contentores 11 porque são infinitos */
```

As restrições que garantem que o objetivo de empacotar o número necessário de objetos são as seguintes:

```
y11_5 + y6_5 + 2y10_5 + yb7_5 - yb5_2 = 5;
y11_4 + y7_4 +  y10_4 + y6_4  + yb7_4 - y4_2  - y4_3  - yb4_3 = 8;
y11_3 + y8_3 +  y5_3  + y10_3 + y7_4  + y4_3  + yb7_4 + yb4_3 - y3_2 = 10;
y11_2 + y9_2 +  y7_2  + y5_3  + y3_2  + y10_2 + y8_2  + y6_4  + 2y4_2 + yb7_5 + yb5_2 + yb3_2
= 8;
```

Optámos por omitir algumas variáveis que tinham o mesmo significado prático. Ex: $y_{7,4}$ e $y_{7,3}$ tem igual efeito segundo certo empacotamento. Subtrações de variáveis impedem a interferência destas com a forma real de como os itens foram empacotados, garantindo que são coerentes com as quantidades concretas de itens.

Finalmente, restringiu-se as variáveis a quantidades inteiras.

```
int y11_5, y6_5, y10_5, yb7_5, y11_4, y7_4, y10_4, y6_4, yb7_4, y11_3, y8_3, y5_3, y10_3,
y4_3, yb7_4, yb4_3, y11_2, y9_2, y7_2, y5_3, y3_2, y10_2, y8_2, y4_2, yb5_2, yb3_2;
```

Pergunta 3: Ficheiro de Input

```
/* Objective function */
min:
11y11_5 + 11y11_4 + 11y11_3 + 11y11_2 + 10y10_5 + 10y10_4 + 10y10_3 + 10y10_2 + 7yb7_5 + 7yb7_4;

/* Restrictions */

y10_5 + y10_4 +y10_3 +y10_2 <= 1;
yb7_5 + yb7_4 +yb7_3 +yb7_2 <= 6;

y11_5 - y6_5         >= 0;
y11_4 - y7_4         >= 0;
y10_4 - y6_4         >= 0;
y11_3 - y8_3         >= 0;
y8_3  - y5_3         >= 0;
y10_3 - y7_4 + y11_4 >= 0;
y7_4  - y4_3         >= 0;
y11_2 - y9_2         >= 0;
y9_2  - y7_2         >= 0;
y5_3  - y3_2 + y11_3 >= 0;
y5_3  - y3_2         >= 0;
y10_2 - y8_2         >= 0;
y8_2  - y6_4 + y10_2 >= 0;
y6_4  - y4_2 + y10_2 >= 0;
yb7_5 - yb5_2        >= 0;
yb5_2 - yb3_2        >= 0;

y11_5 + y6_5 + 2y10_5 + yb7_5 - yb5_2 = 5;
y11_4 + y7_4 +  y10_4 + y6_4  + yb7_4 - y4_2  - y4_3  - yb4_3 = 8;
y11_3 + y8_3 +  y5_3  + y10_3 + y7_4  + y4_3  + yb7_4 + yb4_3 - y3_2 = 10;
y11_2 + y9_2 +  y7_2  + y5_3  + y3_2  + y10_2 + y8_2  + y6_4  + 2y4_2 + yb7_5 + yb5_2 + yb3_2 = 8;


int y11_5, y6_5, y10_5, yb7_5, y11_4, y7_4, y10_4, y6_4, yb7_4, y11_3, y8_3, y5_3, y10_3, y4_3,
yb7_4, yb4_3, y11_2, y9_2, y7_2, y5_3, y3_2, y10_2, y8_2, y4_2, yb5_2, yb3_2;
```

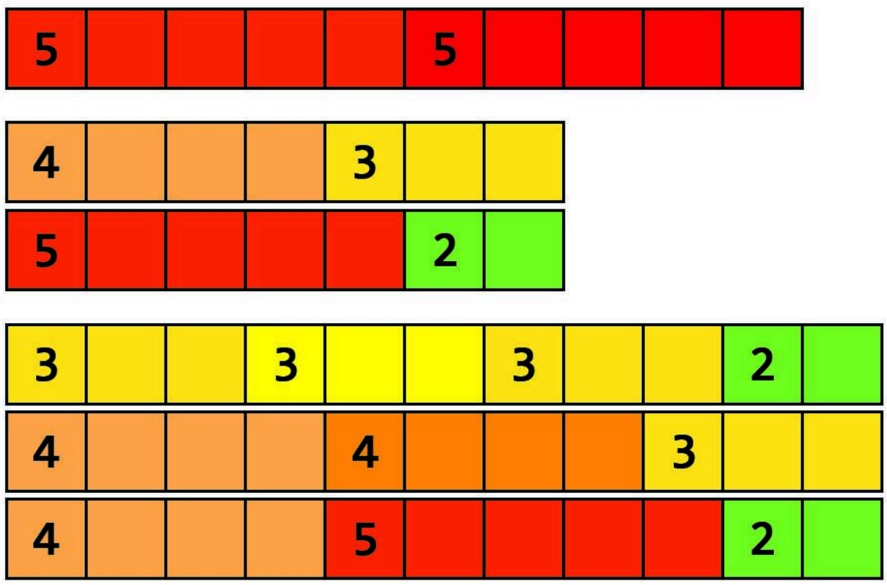
Pergunta 4: Output do LPSolve

```
Variable yb7_4 declared integer more than once, ignored on line 33
Variable y5_3 declared integer more than once, ignored on line 33

Model name:  'LPSolver' - run #1
Objective:   Minimize(R0)

SUBMITTED
Model size:        22 constraints,      26 variables,           78 non-zeros.
Sets:                                    0 GUB,                  0 SOS.

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.


Relaxed solution       103.333333333 after         21 iter is B&B base.

Feasible solution                108 after         30 iter,         1 nodes (gap 3.8%)
Improved solution                104 after         32 iter,         3 nodes (gap 0.0%)

Optimal solution                 104 after         32 iter,         3 nodes (gap 0.0%).

Relative numeric accuracy ||*|| = 0

 MEMO: lp_solve version 5.5.2.11 for 32 bit OS, with 64 bit REAL variables.
      In the total iteration count 32, 0 (0.0%) were bound flips.
      There were 2 refactorizations, 0 triggered by time and 1 by density.
       ... on average 16.0 major pivots per refactorization.
      The largest [LUSOL v2.2.1.0] fact(B) had 63 NZ entries, 1.0x largest basis.
      The maximum B&B level was 3, 0.1x MIP order, 3 at the optimal solution.
      The constraint matrix inf-norm is 2, with a dynamic range of 2.
      Time to load data was 0.003 seconds, presolve used 0.006 seconds,
       ... 0.029 seconds in simplex solver, in total 0.038 seconds.
```

Variables	MILP ...	MILP ...	result
	108	104	104
y11_5	0	0	0
y11_4	5	4	4
y11_3	1	2	2
y11_2	0	0	0
y10_5	0	1	1
y10_4	0	0	0
y10_3	0	0	0
y10_2	0	0	0
yb7_5	6	4	4
yb7_4	0	0	0
yb7_3	0	0	0
yb7_2	0	0	0
y6_5	0	0	0
y7_4	5	4	4
y6_4	0	0	0
y8_3	1	2	2
y5_3	1	2	2
y4_3	2	0	0
y9_2	0	0	0
y7_2	0	0	0
y3_2	0	0	0
y8_2	0	0	0
y4_2	0	0	0
yb5_2	1	1	1
yb3_2	0	1	1
yb4_3	0	0	0

Pergunta 5: Interpretação da solução óptima

O somatório da ocupação bruta dos contentores usados resultou em 104 unidades. Mais especificamente, foram 4 contentores de 11, 1 contentor de 10 e 4 contentores de 7.
Vejam-se, abaixo, alguns exemplos de padrões de empacotamento possíveis segundo a tabela de variáveis gerada pelo LPSolve (apresentada na P4):



Pergunta 6: Validação do Modelo

Primeiramente, verificamos todos os limites físicos dos contentores perante os itens disponíveis e nada indicou que pudesse haver qualquer impedimento na credibilidade do modelo.

As estatísticas apresentadas no output do LPSolve sugerem uma suficiente aproximação do modelo a uma solução teoricamente ótima. A mínima diferença entre a solução viável (que obteve z = 108 após 30 iterações) e a solução melhorada (que com mais 2 iterações só diminuiu z por 4 unidades) fala por si própria.

Conclusão

Este projeto viabilizou a criação de um algoritmo para otimizar o empacotamento de itens em contentores de vários comprimentos, utilizando o software LPSolve. Foram conduzidas análises abrangentes para determinar a melhor maneira de alocar os itens nos contentores, visando minimizar o comprimento total dos contentores utilizados na solução ideal.

O modelo de programação linear desenvolvido definiu as variáveis de decisão, os parâmetros, a função objetivo e as restrições necessárias para resolver eficazmente o problema de empacotamento. O LPSolve foi capaz de resolver o modelo linear, fornecendo uma solução ótima que organizou os itens nos contentores de forma eficiente, resultando num comprimento total mínimo de 104 unidades.

Em resumo, este projeto destacou a importância da programação linear e do LPSolve na resolução de problemas complexos de otimização, permitindo alcançar soluções ótimas para o empacotamento de itens em contentores com diferentes capacidades.