Programação Linear - Projecto Exemplo

Análise e Síntese de Algoritmos @ IST

December 2024

1 Problema

O Professor Natalino Caracol foi contratado pela empresa UbiquityInc, em Rovaniemi na Lapónia, para desenvolver um programa que permita estimar o lucro máximo que pode ser obtido com a produção e venda de brinquedos durante o Natal.

A empresa produz diariamente um conjunto de n brinquedos de madeira $\{b_1, \ldots, b_n\}$, onde cada brinquedo b_i tem um lucro lb_i. Para além de um limite máximo de produção de cada brinquedo devido a restrições de linha de montagem, a empresa está limitada a uma quantidade máxima total de brinquedos que podem ser produzidos por dia, devido a restrições de corte da floresta boreal. Adicionalmente, este Natal a empresa decidiu, para além de vender cada brinquedo individualmente, vender também uma gama de m pacotes especiais, $\{p_1, ..., p_m\}$. Cada pacote contém três brinquedos distintos, sendo o lucro resultande da sua venda superior à soma dos lucros individuais dos brinquedos que o constituem.

O objectivo consiste em indicar ao Rüdolf, CEO da UbiquityInc, qual o lucro máximo que se pode obter diariamente. A UbiquityInc, tratará posteriormente do problema da distribuição.

Input Contém informação sobre os n brinquedos e m pacotes, da seguinte forma:

- Uma linha contendo três inteiros: n indicando o número de diferentes brinquedos passíveis de serem produzidos, m indicando o número de pacotes especiais, e max indicando o número máximo de brinquedos que podem ser produzidos por dia;
- Uma lista de n linhas, em que cada linha contém dois inteiros lb_i e c_i, indicando o lucro e a capacidade de produção do brinquedo b_i;
- Uma lista de m linhas, em que cada linha contém o lucro do pacote i, lp_i, e os três brinquedos que o constitutem, b_1, b_2, e b_3.

Output O objectivo é calcular o lucro máximo que o Rüdolf pode obter diariamente.

2 Solução - Modelo

Identificação das Variáveis do Problema Usamos dois conjuntos de variáveis:

- Variáveis que denotam o número de unidades a produzir de cada *brinquedo isoladamente* (os que não entram nos pacotes): $\{x_i \mid_{i=1}^{n}\}$
- Variáveis que denotam o número de unidades a produzir de cada *pacote*: $\{y_i \mid_{i=1}^{m}\}$

Modelação da Função Objectivo e Restrições do Problema O objectivo é maximizar o lucro total, devendo por isso ter em conta tanto os brinquedos produzidos isoladamente como os pacotes:

$$\mathbf{max} \sum_{i=1}^{n} lb_i.x_i \ + \ \sum_{i=1}^{m} lp_i.y_i$$

Temos de modelar as restrições seguintes:

- *Restrição de produção de cada brinquedo:* Consideramos n restrições, uma por cada tipo i de brinquedo a produzir:

$$x_i + \sum \{y_k \mid_{k=1}^{m} \ \wedge \text{ o pacote } p_k \text{ contém o brinquedo } b_i\} \leq c_i$$

- *Restrição da produção global:*

$$\sum_{i=1}^{n} x_i + 3. \sum_{i=1}^{m} y_i \leq max$$

Combinando as restrições com a função objectivo obtemos o programa linear final:

$$\begin{aligned}
&\mathbf{max} \textstyle\sum_{i=1}^{n} lb_i.x_i \ + \ \sum_{i=1}^{m} lp_i.y_i \\
&\quad x_i + \textstyle\sum \{y_k \mid_{k=1}^{m} \wedge \text{ o pacote } k \text{ contém o brinquedo } i\} \leq c_i \\
&\quad\quad \mathbf{para} \ i \in \{1, ..., n\} \\
&\quad \textstyle\sum_{i=1}^{n} x_i + 3. \sum_{i=1}^{m} y_i \leq max \\
&\quad x_i \geq 0 \mid_{i=1}^{n}, y_i \geq 0 \mid_{i=1}^{m}
\end{aligned}$$

Complexidade do Modelo Pretendemos agora determinar a complexidade do modelo proposto em função dos parâmetros do problema: número de brinquedos, n, e número de pacotes, m:

- *Número de variáveis:* $n + m$, uma variável por brinquedo e uma variável por pacote. Dizemos portanto que o número de variáveis é $\Theta(n + m)$.
- *Número de restrições:* $n + 1 + n + m$, uma restrição de limite de produção por brinquedo (n), a restrição de produção global (1), e a uma restrição de não-negatividade por variável ($n + m$). Dizemos portanto que o número de restrições é $\Theta(n + m)$.

3 Solução - Implementação

Implementamos este modelo linear em **Python com recurso à biblioteca PuLP** para resolução de problemas LP (`https://pypi.org/project/PuLP/`). Exemplos disponíveis em `https://github.com/coin-or/pulp/tree/master/examples`. Passos a seguir:

- Criar a instância do programa linear:

  ```
  prob = LpProblem("P3", LpMaximize)
  ```

  Usamos a flag `LpMaximize` para indicar que se trata de um problema de maximização.

- Definir as variáveis do problema:

  ```
  xs_names = ["x" + str(i) for i in range(1, n+1)]
  ys_names = ["y" + str(i) for i in range(1, m+1)]

  for x in xs_names :
    xs[x] = LpVariable(x, 0, None, LpInteger)
  for y in ys_names :
    ys[y] = LpVariable(y, 0, None, LpInteger)
  ```

  Onde os dicionários `xs` e `ys` guardam respectivamente as variávies PuLP $\{x_i\ |_{i=1}^{n}\}$ e $\{y_i\ |_{i=1}^{p}\}$.

- Definir a função objectivo:

  ```
  prob += lpSum([lb[x] * xs[x] for x in xs_names])
        + lpSum([lp[y] * ys[y] for y in ys_names])
        , "objectivo"
  ```

  Onde os dicionários `lb` e `lp` mapeiam respectivamente o nome de cada variável de brinquedo e de pacote no lucro correspondente.

- Definir as restrições de produção de cada brinquedo:

  ```
  for x in xs_names :
    prob += xs[x] + lpSum(ys[y] for y in packs[x])
            <= c[x], f"quantidade por brinquedo {x}"
  ```

  Onde o dicionário `c` mapeia o nome de cada variável de producto na capacidade de produção do mesmo e o dicionário `packs` mapeia o nome de cada variável de produto nos nomes das variáveis correspondentes aos pacotes que incluem esse produto.

- Definir a restrição de produção global:

  ```
  prob += lpSum([xs[x] for x in xs_names])
       +  lpSum([ys[y]*3 for y in ys_names])
          <= MAX, "restricao global"
  ```

- Calcular a solução do programa linear e imprimir o respectivo valor:

  ```
  prob.solve(GLPK(msg=0))
  print(int(pulp.value(prob.objective)))
  ```

- Para debugging pode ser conveniente imprimir os valores das variáveis:

  ```
  for x in xs:
    print(f"{x}={value(xs[x])}")
  for y in ys:
    print(f"{y}={value(ys[y])}")
  ```

Observação: Para se definir com facilidade o programa linear em PuLP é conveniente manter a informação sobre as variáveis do problema adequadamente organizada em estruturas auxiliares que devem ser construídas num passo de pre-processamento. No caso do exemplo em questão, utilizámos os dicionários:

- `lb` e `lp` para guardar os lucros associados aos brinquedos e aos pacotes;
- `c` para guardar as capacidades de produção de cada brinquedo; e
- `packs` para associar cada brinquedo aos pacotes nos quais participa.

É essencial que escolham com cuidado as estruturas de suporte à vossa solução. Escolhas inadequadas podem resultar em código ineficiente e/ou desnecessariamente complicado.