Trabalho 1 - Otimização

Pedro Amaral Chapelin

- ## Descrição do problema

O problema da produção de produtos químicos consiste em um problema de programação linear de maximização. Ele nos apresenta que certa empresa produz (**N**) produtos, sendo que cada produto é formado de (**M**)compostos. Cada composto tem seu valor necessário para ser produzido (**Pj**), juntamente com um limite em litros (**Qj**) para limitação do mesmo. Além disso, cada composto apresenta a quantidade em litros que existe dele mesmo em cada produto (**Ci**j - *Sendo que i indica o produto, e j o composto*). Por fim, cada produto tem seu valor de venda (**Vi**).

Exemplificando em forma de tabela:

```
                    |      compostos      |
                    | m1   m2   ..   my   | valor
              ------|---------------------|---------
                n1  | c11  c12  ..  c1y   | v1
                n2  | c21  c22  ..  c2y   | v2
produtos  -> ...    | ..   ..   ..  ..    | ...
                nt  | ct1  ct2  ..  cty   | vt
              ------|---------------------|---------
         custo      | p1   p2   ..  py    |
         limite     | q1   q2   ..  qy    |
                    |                     |
```

- Sendo *t* o máximo de n, e *y* o máximo de m.

- ## Modelagem

Ao interpretarmos bem o problema, podemos concluir que o que deve ser maximizado é o lucro com a venda dos produtos.

Logo, o Lucro seria o valor de cada produto, menos o quanto foi gasto com cada composto.

Quanto ganhou = $\sum_{i\,=\,0}^{t} V_i \times N_i$

Quanto gastou = $\sum_{i=0}^{t} N_i \times \left(\sum_{i=0}^{y} \left(C_{ij} \times P_j\right)\right)$

Lucro = **Quanto ganhou** - **Quanto gastou**.

Além disso, temos que obedecer a algumas restrições impostas, que dizem respeito ao limite de cada composto nos produtos. Logo, modelando com essas informações:

max:

$$\left(\sum_{i=0}^{t} V_i \times N_i\right) - \left(\sum_{i=0}^{t} N_i \times \left(\sum_{i=0}^{y} (C_{ij} \times P_j)\right)\right)$$

s.a.:

Para cada composto **j** (coluna da tabela), existe a restrição:

$\left(\sum_{i=0}^{t} (C_i \times N_i)\right) \leq Q_j$

Com as condições propostas pelo exemplo do enunciado:

max:

$2{,}8N_0 + 3{,}3N_1 + 1{,}8N_2$

s.a.:

$0{,}2N_0 + 1N_1 + 0{,}4N_2 \leq 1000;$

$0{,}5N_0 + 0{,}1N_1 + 0{,}2N_2 \leq 2000;$

$1N_0 + 0{,}3N_1 + 0{,}2N_2 \leq 500;$

$0{,}1N_0 + 0{,}1N_1 \leq 2000;$

$N_0, N_1, N_2 \geq 0$

Na forma equacional, adicionando variáveis de folga:

max:

$2{,}8N_0 + 3{,}3N_1 + 1{,}8N_2$

s.a.:

$0{,}2N_0 + 1N_1 + 0{,}4N_2 + N_3 = 1000;$

$0{,}5N_0 + 0{,}1N_1 + 0{,}2N_2 + N_4 = 2000;$

$1N_0 + 0{,}3N_1 + 0{,}2N_2 + N_5 = 500;$

$0{,}1N_0 + 0{,}1N_1 + N_6 = 2000;$

$N_0, N_1, N_2, N_3, N_4, N_5, N_6 \geq 0$

Segundo o livro *Understanding and Using Linear Programming(*Jiří Matoušek e Bernd Gärtner), esses passos são necessários para que possamos usar o método *Simplex* para resolver o problema, pois ao fazer isso, temos uma região viável que pode assumir vários valores, mas queremos o mais extremo à nossa *Função Objetivo* e que esteja dentro das restrições propostas pelo problema.

- **Resolução do problema**

Juntamente a este relatório, estarão os arquivos fontes que receberão os dados do problema de uma maneira especificada, e os colocarão do jeito que são as entradas do programa *lp_solve*, programa que implementará o *Simplex* e devolverá a solução ótima para este PL, caso exista.

Pegando como exemplo o a entrada citada anteriormente (*entrada1.in*):

```
3 4
10 7 3
1 1000
2 2000
5 500
10 2000
0.2 0.5 1.0 0.1
1.0 0.1 0.3 0.1
0.4 0.2 0.2 0.0
```

Basicamente o arquivo principal ***DotLPMaker.c****,* vai pegar todos os dados nesse arquivo e guardá-los em devidas estruturas, essa é a etapa 1 do processo. Após isso, a etapa 2 consiste em escrever os dados em um novo arquivo (*entrada.lp*), da maneira que fique no modelo das entradas do resolvedor lp_solve.

Saída para este exemplo: (*coloquei os x com um índice a mais para não ter que iniciar do 0.*)

```
max: 2.8x1 + 3.3x2 + 1.2x3;

0.2x1 + 1.0x2 + 0.4x3 <= 1000;

0.5x1 + 0.1x2 + 0.2x3 <= 2000;

1.0x1 + 0.3x2 + 0.2x3 <= 500;

0.1x1 + 0.1x2 + 0.0x3 <= 2000;

x1 >= 0;

x2 >= 0;

x3 >= 0;
```

Desta maneira, podemos jogar esse arquivo no lp_solve, de modo que retornará a resposta esperada:

```
Value of objective function: 3755.31914894

Actual values of the variables:

x1                      212.766

x2                      957.447

x3                            0
```