# Problema de Produção de Produtos Químicos - Resolução por meio de PL

**Ariel Evaldt Schmitt, Paulo Mateus Luza Alves**

Departamento de Informática – Universidade Federal do Paraná (UFPR)

aes20@inf.ufpr.br (GRR20203949) pmla20@inf.ufpr.br (GRR20203945)

## 1. O PROBLEMA DA PRODUÇÃO DE QUÍMICOS

O problema consiste em uma empresa química que produz uma certa quantidade $n$ de diferentes produtos químicos, e para produzir estes produtos, são utilizados $m$ compostos diferentes (matérias primas).

Com estas informações, o relacionamento entre os *i* produtos e *j* compostos ocorre da seguinte maneira:

- Cada produto *i* possui um valor de venda (por litro) $v_i$ ;
- Cada composto *j* possui um preço de compra $p_j$ ;
- Cada composto *j* possui uma quantidade máxima de volume diária (em litros) $q_j$ ;

A quantidade de cada composto (em litros) usada para produzir 1 litro de um produto é especificada como $C_{ij}$ , onde temos:

$$C_{ij} \text{ para } 1 \leq i \leq n,\ 1 \leq j \leq m$$

Dessa forma, para produzir 1 litro do produto 1, temos que utilizar

$$C_{11} + C_{12} + \ldots + C_{1m}$$

Por fim, a empresa busca maximizar o lucro obtido na produção e venda dos produtos, considerando que todo o estoque será vendido e que os demais custos de fabricação possam ser desconsiderados.

## 2. A MODELAGEM

Com base nas informações acima, chegamos no seguinte modelo, estando em função de $n$ e $m$:

$$\text{max: } \sum_{i=1}^{n} v_i * x_i - \sum_{i=1}^{n} x_i * \left(\sum_{j=1}^{m} p_j * C_{ij}\right)$$

subject to*:*

$$\sum_{i=1}^{n} C_{ij} * x_i \leq q_j,\ \forall j \in [1..m]$$

## 2.1 A FUNÇÃO OBJETIVO

A função objetivo deste Programa Linear consiste no cálculo do lucro obtido pela empresa através da venda dos produtos químicos. Logo, o primeiro somatório está representando o valor total obtido pela venda dos produtos sendo $v_i$ e $x_i$ o valor de venda e a quantidade do produto *i,* respectivamente. Por fim, o segundo somatório consiste no cálculo do custo de produção total de todos os $n$ produtos, o qual é obtido através da multiplicação da quantidade $x_i$ do produto $i$ pela soma da quantidade de cada um de seus compostos $j$ multiplicado pelo preço deste composto. Sendo assim, essa fórmula calcula o lucro da empresa, o qual é composto pelo preço de venda descontado do seu custo de produção.

## 2.2 AS RESTRIÇÕES

Para as restrições temos um conjunto de equações relacionadas ao limite diário de volume (em litros) de cada composto. Devido a isso, para cada composto $j$ temos uma equação simbolizando a quantidade gasta dele na produção dos $n$ produtos químicos, de forma que essa quantidade não possa ultrapassar o valor seu limite diário representado por $q_j$ .

# 3. A IMPLEMENTAÇÃO

O algoritmo de implementação desta modelagem foi desenvolvido em linguagem C, e para o correto funcionamento, o código foi dividido em 2 etapas, sendo elas:

1. Leitura do *stdin*, coletando todas as informações necessárias e as armazenando na struct referenciada abaixo;
2. Escrita no *stdout* no formato indicado para o programa *lp_solve* e seguindo a modelagem indicada acima.

## 3.1 A STRUCT DE PRODUÇÃO

Este *struct* é responsável por armazenar todas as informações necessárias para a construção da saída do algoritmo da forma correta.

```
typedef struct ProductionInput
{
    int productsQty, compositesQty;
    float *compositesCost;
    float *compositesLimit;
    float *productsValue;
    float **productsRecipe;
} ProductionInput;
```

Figura 1 - Estrutura de dados da produção

A estrutura possui duas variáveis inteiras, as quais armazenam a quantidade de produtos e compostos com que a empresa trabalha. Além do mais, são definidos 3 vetores de ponto

flutuante, os quais armazenam o custo de cada composto, seu limite diário e o valor de venda de cada um dos produtos, respectivamente. Por fim, temos um vetor bidimensional responsável por armazenar a composição de cada produto, ou seja, a quantidade necessária de cada composto para fabricar um determinado produto.

## 3.2 PRINCIPAIS FUNÇÕES

```
ProductionInput *read_production();
```

Esta função tem a responsabilidade de fazer a leitura do *stdin* e armazenar as informações necessárias na *struct ProductionInput*.

```
void *get_objective_function(ProductionInput *production);
```

Esta função tem a responsabilidade de construir a função objetivo e escrever em *stdout*. Portanto, a partir dos dados obtidos na função *read_production*, ela monta a função objetivo no formato de entrada para o programa *lp_solve* e de acordo com a modelagem apresentada anteriormente.

```
void *get_constraints(ProductionInput *production);
```

Esta função tem a responsabilidade de construir as funções de restrição e escrever em *stdout*. Portanto, a partir dos dados obtidos na função *read_production*, ela monta as restrições do programa linear no formato de entrada para o programa lp_solve e de acordo com a modelagem do problema.

## 4. EXEMPLOS

Utilizamos 3 exemplos para testar o nosso trabalho, sendo o primeiro deles o fornecido pelo professor na especificação do trabalho e os outros dois montados por nós.

4.1 1° EXEMPLO

Considere o problema sendo executado para 3 produtos e 4 compostos seguindo o seguinte contexto:

| | | Composto | | | | |
|---|---|---|---|---|---|---|
| | n/m | 1 | 2 | 3 | 4 | Valor |
| Produtos | 1 | 0.2 | 0.5 | 1.0 | 0.1 | 10 |
| | 2 | 1.0 | 0.1 | 0.3 | 0.1 | 7 |
| | 3 | 0.4 | 0.2 | 0.2 | 0.0 | 3 |
| | Custo | 1 | 2 | 5 | 10 | |
| | Limite | 1000 | 2000 | 500 | 2000 | |

Logo, como entrada para nosso algoritmo temos o seguinte conteúdo, que está no arquivo exemplo1.txt:

```
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
```

Logo, a partir desta entrada, temos a seguinte modelagem resultante do nosso algoritmo:

```
max: 10.000000x1 + 7.000000x2 + 3.000000x3 - 7.200000x1 - 3.700000x2 - 1.800000x3;
0.200000x1 + 1.000000x2 + 0.400000x3 <= 1000.000000;
0.500000x1 + 0.100000x2 + 0.200000x3 <= 2000.000000;
1.000000x1 + 0.300000x2 + 0.200000x3 <= 500.000000;
0.100000x1 + 0.100000x2 + 0.000000x3 <= 2000.000000;
x1 >= 0; x2 >= 0; x3 >= 0;
```

e, ao executar esta modelagem no lp_solve, obtivemos o seguinte resultado:

```
Value of objective function: 3755.31914894
Actual values of the variables:
x1                    212.766
x2                    957.447
x3                          0
```

4.2 2° EXEMPLO

Considere o problema sendo executado para 3 produtos e 4 compostos seguindo o seguinte contexto:

| | | Composto | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Produtos | n/m | 1 | 2 | 3 | 4 | 5 | 6 | Valor |
| | 1 | 1.0 | 0.5 | 0.6 | 0.2 | 0.1 | 0.0 | 10 |
| | 2 | 0.7 | 0.3 | 0.7 | 0.3 | 0.4 | 1.0 | 6 |
| | 3 | 1.2 | 0.8 | 0.3 | 0.5 | 0.1 | 0.0 | 8 |
| | 4 | 0.0 | 0.2 | 0.0 | 0.5 | 0.2 | 0.9 | 4 |
| | 5 | 0.1 | 0.2 | 0.5 | 0.0 | 0.0 | 0.1 | 9 |
| | Custo | 2 | 4 | 3 | 8 | 4 | 2 | |
| | Limite | 1200 | 2000 | 400 | 2000 | 1000 | 900 | |

Logo, como entrada para nosso algoritmo temos o seguinte conteúdo, que está no arquivo exemplo2.txt:

```
5 6
10 6 8 4 9
2 1200
4 2000
3 400
8 2000
4 1000
2 900
1.0 0.5 0.6 0.2 0.1 0.0
0.7 0.3 0.7 0.3 0.4 1.0
1.2 0.8 0.3 0.5 0.2 0.9
0.0 0.2 0.0 0.5 0.2 0.9
0.1 0.2 0.5 0.0 0.0 0.1
```

Logo, a partir desta entrada, temos a seguinte modelagem resultante do nosso algoritmo:

max: 10.000000x1 + 6.000000x2 + 8.000000x3 + 4.000000x4 + 9.000000x5 - 7.800000x1 - 10.700000x2 - 13.100000x3 - 7.400001x4 - 2.700000x5;
1.000000x1 + 0.700000x2 + 1.200000x3 + 0.000000x4 + 0.100000x5 <= 1200.000000;
0.500000x1 + 0.300000x2 + 0.800000x3 + 0.200000x4 + 0.200000x5 <= 2000.000000;
0.600000x1 + 0.700000x2 + 0.300000x3 + 0.000000x4 + 0.500000x5 <= 400.000000;
0.200000x1 + 0.300000x2 + 0.500000x3 + 0.500000x4 + 0.000000x5 <= 2000.000000;
0.100000x1 + 0.400000x2 + 0.200000x3 + 0.200000x4 + 0.000000x5 <= 1000.000000;
0.000000x1 + 1.000000x2 + 0.900000x3 + 0.900000x4 + 0.100000x5 <= 900.000000;
x1 >= 0; x2 >= 0; x3 >= 0; x4 >= 0; x5 >= 0;

e, ao executar esta modelagem no lp_solve, obtivemos o seguinte resultado:

Value of objective function: 5040.00000000

Actual values of the variables:

| | |
|---|---|
| x1 | 0 |
| x2 | 0 |
| x3 | 0 |
| x4 | 0 |
| x5 | 800 |

4.3 3° EXEMPLO

Considere o problema sendo executado para 3 produtos e 4 compostos seguindo o seguinte contexto:

| | | Composto | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | n/m | 1 | 2 | 3 | 4 | 5 | 6 | Valor |
| Produtos | 1 | 0.2 | 0.08 | 0.22 | 0.38 | 0.1 | 0.02 | 20 |
| | 2 | 0.08 | 0.16 | 0.14 | 0.34 | 0.13 | 0.15 | 15 |
| | 3 | 0.05 | 0.75 | 0.05 | 0.05 | 0.05 | 0.05 | 8 |
| | 4 | 0.25 | 0.12 | 0.18 | 0.1 | 0.05 | 0.3 | 10 |
| | 5 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.25 | 12 |
| | 6 | 0.5 | 0.1 | 0.0 | 0.15 | 0.15 | 0.1 | 14 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 0.105 | 0.13 | 0.13 | 0.22 | 0.22 | 0.195 | 4 |
| | Custo | 2 | 6 | 3 | 9 | 12 | 3 | |
| | Limite | 1000 | 1500 | 2000 | 500 | 3000 | 1400 | |

Logo, como entrada para nosso algoritmo temos o seguinte conteúdo, que está no arquivo exemplo3.txt:

7 6
20 15 8 10 12 14 4
2 1000
6 1500
3 2000
9 500
12 3000
3 1400
0.200 0.08 0.22 0.38 0.10 0.020
0.080 0.16 0.14 0.34 0.13 0.150
0.050 0.75 0.05 0.05 0.05 0.050
0.250 0.12 0.18 0.10 0.05 0.300
0.150 0.15 0.15 0.15 0.15 0.250
0.500 0.10 0.00 0.15 0.15 0.100
0.105 0.13 0.13 0.22 0.22 0.195

Logo, a partir desta entrada, temos a seguinte modelagem resultante do nosso algoritmo:

max: 20.000000x1 + 15.000000x2 + 8.000000x3 + 10.000000x4 + 12.000000x5 + 14.000000x6 + 4.000000x7 - 6.220000x1 - 6.610000x2 - 5.950000x3 - 4.160000x4 - 5.550000x5 - 5.050000x6 - 6.585000x7;
0.200000x1 + 0.080000x2 + 0.050000x3 + 0.250000x4 + 0.150000x5 + 0.500000x6 + 0.105000x7 <= 1000.000000;
0.080000x1 + 0.160000x2 + 0.750000x3 + 0.120000x4 + 0.150000x5 + 0.100000x6 + 0.130000x7 <= 1500.000000;
0.220000x1 + 0.140000x2 + 0.050000x3 + 0.180000x4 + 0.150000x5 + 0.000000x6 + 0.130000x7 <= 2000.000000;
0.380000x1 + 0.340000x2 + 0.050000x3 + 0.100000x4 + 0.150000x5 + 0.150000x6 + 0.220000x7 <= 500.000000;
0.100000x1 + 0.130000x2 + 0.050000x3 + 0.050000x4 + 0.150000x5 + 0.150000x6 + 0.220000x7 <= 3000.000000;
0.020000x1 + 0.150000x2 + 0.050000x3 + 0.300000x4 + 0.250000x5 + 0.100000x6 + 0.195000x7 <= 1400.000000;
x1 >= 0; x2 >= 0; x3 >= 0; x4 >= 0; x5 >= 0; x6 >= 0; x7 >= 0;

e, ao executar esta modelagem no lp_solve, obtivemos o seguinte resultado:

Value of objective function: 26633.33333333

Actual values of the variables:

| | |
|---|---|
| x1 | 0 |
| x2 | 0 |
| x3 | 0 |
| x4 | 3333.33 |
| x5 | 1111.11 |
| x6 | 0 |
| x7 | 0 |