Relatório Trabalho Prático I
Programação Linear

Otimização (ci1238)
Departamento de Informática
Universidade Federal do Paraná - UFPR
Eduardo Gabriel Kenzo Tanaka (GRR20211791)
Curitiba - PR

I. Introdução

Trabalho que tem por objetivo modelar e implementar, por programação linear, o problema da produção de produtos químicos, o qual é apresentado abaixo:

Uma empresa química produz *n* tipos diferentes de produtos. Para produzir estes produtos usa diferentes proporções de *m* diferentes compostos (matérias primas). Cada produto *i* tem valor de venda (por litro), *vi*. Cada composto *j* usado tem um preço (por litro), *pj*, e um limite diário de volume (em litros), *qj* . A quantidade (em litros) de uso de cada composto *j* na produção de 1 litro do produto *i* é dada por *cij*. Ou seja, para produzir 1 litro do produto *i* são necessários *ci1* litros do composto 1, *ci2* litros do composto 2, e assim até *cim* litros do composto m.

A empresa assume que toda sua produção será vendida. Levando em consideração os dados, e que os demais custos de produção não dependem de quais produtos são feitos, a empresa quer maximizar os lucros.

II. Modelagem

1. Variáveis

Primeiramente, retirando as informações do problema, de forma a ficar mais claro a função de cada variável:

n: quantidade de produtos, $i \in [1..n]$
m: quantidade de compostos, $j \in [1..m]$
pj: preço pelo litro de composto *j*
qj: limite de litros produzidos do composto *j*
vi: valor de venda de 1 litro de produto *i*
cij: quantidade do composto *j* utilizado para produzir 1 litro do produto *i*

Uma vez que o objetivo é maximizar o lucro, as incógnitas serão a quantidade, em litros, de cada produto que deve ser produzido, denominadas, nesta implementação, de *x1, x2, ..., xn.*

2. Função Objetivo

Sabendo que a empresa quer maximizar os lucros, para encontrar a função objetivo, é preciso saber qual o lucro de cada produto *i.*

$$lucro = valor\ de\ venda - custo\ do\ produto$$

O valor de venda por litro é dado como entrada (vi), já o custo do produto é dado pela soma dos custos dos compostos utilizados em sua formulação. O custo por litro do produto *i* é sumarizado pelo somatório abaixo.

$$\sum_{j=1}^{m} (pj \cdot cij)$$

Por sua vez, o lucro por litro do produto *i* será:

$$vi - \sum_{j=1}^{m} (pj \cdot cij)$$

E o lucro do produto para *xi* litros de produto será:

$$[\, vi - \sum_{j=1}^{m} (pj \cdot cij)\,] \cdot xi$$

A função objetivo é obtida somando o lucro de todos produtos, ou seja:

$$\sum_{i=1}^{n} [(vi - \sum_{j=1}^{m} (pj \cdot cij)) \cdot xi]$$

3. Restrições

As restrições são as quantidades limite de composto que podem ser produzidas, ou seja, a soma da quantidade de composto *j* utilizado em cada produto tem que ser menor que o limite de produção *qj*. Dessa forma, serão *m* inequações de restrição – um para cada composto –, conforme ilustrado abaixo:

$$\sum_{i=1}^{n} (xi \cdot cij) \leq qj, \; j = 1, \ldots, m$$

Além disso, existem as restrições de positividade, uma vez que a quantidade de produto produzido não pode ser negativa. Serão mais *n* inequações da forma:

$$xi \geq 0, \; i = 1, \ldots, n$$

4. Programa Linear

A partir dessas informações chega-se ao programa linear abaixo:

maximizar:

$$\sum_{i=1}^{n} [(vi - \sum_{j=1}^{m} (pj \cdot cij)) \cdot xi]$$

sujeito à:

$$\sum_{i=1}^{n} (xi \cdot cij) \leq qj, \; j = 1, 2, \ldots, m$$

$$xi \geq 0, \; i = 1, 2, \ldots, n$$

III. Implementação

1. Estruturas

O programa para processar os dados conforme a modelagem acima foi implementado criando duas *structs*: uma para os produtos, contendo o preço de venda, o lucro e quantidade de cada composto utilizado em sua formulação; e outra *struct* para as informações do PL, contendo o custo e o limite de produção de cada composto *j* e um vetor da *struct* de produto.

```
typedef struct produto_t {
	double lucro;
	double precoVenda;
	double *quantComposto;
} produto_t;
```

```
typedef struct PL {
	unsigned int n, m;
	produto_t *produtos;
	double *custoComposto;
	double *limiteComposto;
} parametros_pl;
```

2. Entrada

A chamada do programa pode ser feita especificando o arquivo de informações e o arquivo de entrada para o *lp_solve*, na forma:

```
./producao <dados> <entrada_lp_solve>
```

Onde *<dados>* é o caminho para o arquivo de entrada com as informações do *PL* e *<entrada_lp_solve>* é o caminho para o arquivo de entrada do *lp_solve* de extensão ".lp" que será gerado. Caso nenhum arquivo seja especificado, por padrão, o programa lerá o arquivo "ex1.txt", que possui o exemplo do enunciado do trabalho, e gerará o arquivo "ex1-input.lp" no diretório */egkt21/*.

3. Main

Na função *main*, primeiramente, abre-se o arquivo com as informações para leitura e o arquivo de *input* do *lp_solve* para escrita. Depois disso, lê-se *n* e *m, inicializa a PL* e coleta as informações do arquivo de entrada e armazena na estrutura de *PL*. Com os dados coletados, é então calculado o lucro a ser maximizado. Por último, o *PL* é montado e impresso no arquivo de *input* do *lp_solve*.

4. Exemplos

Os exemplos de entrada utilizados estão no subdiretório */exemples.* O primeiro exemplo testado foi o apresentado no enunciado do trabalho:

```
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
```

Neste caso, o resultado ótimo foi um máximo de 3755.319, com x1 = 212.766; x2 = 957.447 e x3 = 0.

O segundo exemplo:

```
4 5
34 9 15 19
1 1600
3 2000
5 550
12 600
15 3500
0.2 0.5 1.0 0.2 1.4
2.0 0.1 0.9 0.1 0.6
0.4 0.5 0.2 0.0 0.7
0.3 1.8 1.0 0.0 0.0
```

Obteve um resultado ótimo com a função objetivo valendo 4565.000, x1 = x2 = 0, x3 = 550.

O terceiro exemplo:

```
4 2
10 7 3 2
1 1100
5 550
0.2 0.5
1.0 0.1
0.4 0.2
0.1 0.3
```

Obteve um ótimo em função objetivo 11733.333, quando x1 = x2 = 916.667 e x3 = x4 = 0.