Modelagem de Produção de Produtos Químicos

Daniel Celestino de Lins Neto[1]**, Gabriel Lisboa Conegero** [1]**,**
Matheus Moraes Piovesan[1]

[1]Departamento de Informática
Otimização – CI1238
Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brasil

`{dcln22, glc22, mmp22}@inf.ufpr.br`

Resumo. *Este relatório descreve a modelagem e a implementação de uma solução para um problema de otimização na produção de produtos químicos. O objetivo é maximizar os lucros de uma empresa considerando as restrições de custos e limites de produção de matérias-primas, utilizando programação linear com saída formatada para o resolvedor* `lp_solve`*.*

1. Descrição do Problema

O problema envolve a produção de n tipos de produtos químicos, utilizando m compostos. Cada produto possui um valor de venda por litro (v_i), enquanto os compostos têm custo por litro (p_j) e um limite de produção diário (q_j). A matriz c_{ij} define a quantidade de cada composto j utilizada na produção de 1 litro do produto i.

2. Modelagem

O problema foi modelado usando Progrmação Linear, onde a entrada é a instancia do problema e a saída é um PL no padrão do `pl_solve`. O processo de modelagem pode ser dividido em três partes, onde em cada uma delas foi gerado uma parte do PL (restrições e função objetivo).

2.0.1. Variáveis

Algumas variáveis foram criadas, são elas

- qtP_i: Quantidade produzida do produto i.
- qtC_i: Quantidade utilizada do componente i.

Para cada produto tem uma qtP_i e para cada componente tem uma qtC_i, então no total vamos ter

- qtP_i tal que $1 \leq i \leq n$.
- qtC_i tal que $1 \leq i \leq m$.

Para cada uma dessas variáveis foi criada a restrição de não negatividade, uma vez que não faz sentido ter quantidades negativas. Ou seja, as seguintes restrições foram criadas.

$$qtP_i \geq 0 \text{ para todo } 1 \leq i \leq n;$$
$$qtC_i \geq 0 \text{ para todo } 1 \leq i \leq m;$$

2.0.2. Função Objetivo

O problema quer maximizar o lucro da produção dos produtos, então fica claro que o que queremos maximizar é quanto vai ser ganho ao vender os produtos, como toda a produção vai ser vendida podemos apenas multiplicar o quanto foi produzido pelo valor de venda de cada produto. A função objetivo é

$$Maximizar : \sum_{i=1}^{n} qtP_i * v_i$$

Porém não estamos considerando os gastos para a produção desse produtos. Os gastos são com os componentes, ou seja quanto foi gasto para adiquirir os componentes para fabricar os produtos. Podemos considerar o custo relativo do dia, ou seja o quanto foi gasto para produzir os produtos e não o que realmente foi gasto no total. Os componentes podem ser comprados todos de uma vez para p mês, mas vamos considerar o gasto relativo do dia. A função objetivo final fica

$$Maximizar : \sum_{i=1}^{n} qtP_i * v_i - \sum_{i=1}^{m} qtC_i * p_i$$

2.0.3. Restrições na Quantidade de Componente

Cada componente tem um limite diário de quanto pode ser usado, então precisamos colocar uma restrição a quanto deve ser gasto por dia para a produção. As restrições ficaram

$$qtC_i \leq q_i \text{ para todo } 1 \leq i \leq m;$$

2.0.4. Restrições nas Receitas dos Produtos

Uma receita é o conjunto de componentes e as quantidades usadas para produzir um litro do produto i. Para que o modelo esteja correto a quantidade de componente utilizada no dia deve ser maior ou igual a quantidade necessária para produir os produtos. Então a quantidade do componente i deve ser maior ou igual a soma do quanto foi gasto do mesmo componente em cada receita. O quanto foi gasto em uma receita i pelo produto j, C_{ij}, depende da quantidade do componente requisitado na receita, c_{ij}, e do quanto foi produzido do produto i ,qtP_i.

$$C_{ij} = qtP_i * c_{ij}$$

Então para saber o total requisitado do produto j, TC_j, basta somar o gasto em todas as receitas i.

$$TC_j = \sum_{i=1}^{n} C_{ij} = \sum_{i=1}^{n} qtP_i * c_{ij}$$

Para cada componente j a quantidade utilizada deve ser maior ou igual a sua quantidade requisitada TC_j. Logo as seguintes restrições aparecem

$$\sum_{i=1}^{n} qtP_i * c_{ij} \leq qtC_j \text{ para todo } 1 \leq j \leq m;$$

2.0.5. Modelo final

Unindo as restrições e a função objetivo ficamos com o modelo final da seguinte forma:

$$
\begin{aligned}
Maximizar: & \sum_{i=1}^{n} qtP_i * v_i - \sum_{i=1}^{m} qtC_i * p_i \\
Restrições: & \\
& qtP_i \geq 0, 1 \leq i \leq n; \\
& qtC_i \geq 0, 1 \leq i \leq m; \\
& qtC_i \leq q_i, 1 \leq i \leq m; \\
& \sum_{i=1}^{n} qtP_i * c_{ij} \leq qtC_j, 1 \leq j \leq m;
\end{aligned}
$$

3. Implementação

A implementação foi desenvolvida em C++ e gera o modelo linear no formato `lp_solve`. Abaixo, destacamos os trechos mais relevantes do código:

3.1. Entrada de Dados

O programa lê os valores de entrada, incluindo o número de produtos (n) e compostos (m), os preços de venda, os custos e limites dos compostos, e a matriz de proporções:

```
// Leitura dos dados principais
cin >> n >> m;

vector<ll> venda(n);
for (auto &i : venda)
    cin >> i;

vector<ll> custo(m), limite(m);
for (ll i = 0; i < m; i++) {
    cin >> custo[i] >> limite[i];
}

vector<vector<double>> receitas(n, vector<double>(m));
for (auto &row : receitas)
    for (auto &i : row)
        cin >> i;
```

3.2. Geração do Modelo Linear

A função objetivo e as restrições são geradas no formato do `lp_solve`, conforme o exemplo abaixo:

```
// Gerar a fun  o objetivo
printf("max:␣");
for (ll i = 0; i < n; i++)
```

```
    printf("+%lld*p%lld␣", venda[i], i);
for (ll j = 0; j < m; j++)
    printf("-%lld*c%lld␣", custo[j], j);
printf(";\n");

// Restri  es de limites
for (ll j = 0; j < m; j++)
    printf("limit_constraint%lld:␣c%lld␣<=␣%lld;\n", j, j, limite[j]);

// Restri  es de receitas
for (ll j = 0; j < m; j++) {
    printf("receita_constraint%lld:␣", j);
    for (ll i = 0; i < n; i++)
        printf("+%lf*p%lld␣w", receitas[i][j], i);
    printf("<=␣c%lld;\n", j);
}
```

3.3. Exemplo de Entrada e Saída

3.3.1. Entrada

```
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
```

3.3.2. Saída Gerada

```
max: +10*p0 +7*p1 +3*p2 -1*c0 -2*c1 -5*c2 -10*c3;
limit_constraint0: c0 <= 1000;
limit_constraint1: c1 <= 2000;
limit_constraint2: c2 <= 500;
limit_constraint3: c3 <= 2000;
receita_constraint0: +0.200000*p0 +1.000000*p1 +0.400000*p2 <= c0;
receita_constraint1: +0.500000*p0 +0.100000*p1 +0.200000*p2 <= c1;
receita_constraint2: +1.000000*p0 +0.300000*p1 +0.200000*p2 <= c2;
receita_constraint3: +0.100000*p0 +0.100000*p1 +0.000000*p2 <= c3;
```