Relatório Primeiro Trabalho Prático

CI1238 - Otimização

Victor Ribeiro Garcia

1. Introdução

Para maximizar os lucros, uma empresa precisa determinar a quantidade de cada produto a ser produzido, levando em consideração as proporções e limitações dos compostos, os preços de venda e os limites de volume diário. Além disso, a empresa parte do pressuposto de que toda a sua produção será vendida. Esse tipo de problema é conhecido como um problema de otimização linear.

2. Modelagem

A abordagem para resolver esse tipo de problema é utilizar a programação linear. A ideia é formular um modelo matemático que represente a situação descrita e, em seguida, aplicar algoritmos de otimização para encontrar a solução ótima.

Neste caso, vamos denotar a quantidade a ser produzida de cada produto i como xi (em litros). O objetivo é maximizar o lucro total, que pode ser expresso como:

Lucro Total = *v1 * x1 + v2 * x2 + ... + vn * xn*

Onde *vi* é o valor de venda por litro do produto *i*.

No entanto, existem algumas restrições que devem ser consideradas. Primeiro, a quantidade de cada composto j utilizado na produção de um litro do produto i é dada por cij. Portanto, a quantidade total de cada composto j utilizada deve respeitar o limite diário qj. Isso pode ser expresso como:

*c11 * x1 + c21 * x2 + ... + cm1 * xm <= q1*
*c12 * x1 + c22 * x2 + ... + cm2 * xm <= q2*
...
*c1n * x1 + c2n * x2 + ... + cmn * xm <= qn*

Essas restrições garantem que a quantidade de cada composto utilizado não exceda o limite diário disponível. Além disso, a quantidade produzida de cada produto i deve ser não negativa: xi >= 0, para todo i

Portanto, o problema de maximização de lucros pode ser formulado como um problema de programação linear, sujeito às restrições mencionadas acima.

3. Implementação

As variáveis **produtos** e **compostos** são inteiras e armazenam a quantidade de produtos e compostos, respectivamente, conforme o código 1.

Código 1

```c
// quantidade de produtos e compostos
int produtos, compostos;
fscanf(stdin, "%d", &produtos);
fscanf(stdin, "%d", &compostos);
```

O *array* **valores** tem tamanho produtos e armazena os valores de venda dos produtos, conforme o código 2.

Código 2

```c
// valor de venda de cada produtos
int valores[produtos];
for (int i = 0; i < produtos; i++)
{
    fscanf(stdin, "%d", &valores[i]);
}
```

Os *arrays* **custo_compostos** e **limites_compostos** têm tamanho **compostos** e são usados para armazenar o custo e o limite diário de volume (em litros) de cada composto, respectivamente, conforme o código 3.

Código 3

```c
// custo e o limite diário de volume de cada composto
int custo_compostos[compostos];
int limites_compostos[compostos];
for (int i = 0; i < compostos; i++)
{
    fscanf(stdin, "%d", &custo_compostos[i]);
    fscanf(stdin, "%d", &limites_compostos[i]);
}
```

O *array* **coeficientes_obj** é um *array* de tamanho **produtos** que armazenará os coeficientes usados na função objetivo. A variável auxiliar **custo** será usada para armazenar o custo de fabricação do produto. A matriz bidimensional **matriz_produtos** armazena as quantidades de compostos necessárias para fabricar cada produto. O código realiza um cálculo envolvendo a matriz (matriz_produtos) e os custos dos compostos (custo_compostos) para calcular os coeficientes a serem usados na função objetivo. **Cada coeficiente é calculado como a diferença entre o valor de venda do produto i** (valores[i]) **e o custo de fabricação do produto i** (resultado da soma das multiplicações da quantidade de

cada composto j utilizado na fabricação do produto i (matriz_produtos[i][j]) pelo custo correspondente de cada composto j (custo_compostos[j])), conforme o código 4.

Código 4

```
// armazena os coeficientes usados na função objetivo
float coeficientes_obj[produtos];
// variável auxiliar usada para armazenar o custo de fabricação do produto i
float custo = 0;
// quantidade de compostos necessarios para fabricar cada produto
float matriz_produtos[produtos][compostos];
// calcula os coeficientes da função objetivo
for (int i = 0; i < produtos; i++)
{
    for (int j = 0; j < compostos; j++)
    {
        fscanf(stdin, "%f", &matriz_produtos[i][j]);
        custo = custo + matriz_produtos[i][j] * custo_compostos[j];
    }
    coeficientes_obj[i] = valores[i] - custo;
    custo = 0;
}
```

O loop for é usado para imprimir a função objetivo no formato: "***max: coeficiente1x0 + coeficiente2x1 + ... + coeficienteN-1xN-1;***", onde os coeficientes são recuperados do *array* coeficientes_obj e os índices correspondem aos produtos, conforme o código 5.

Código 5

```
// imprimir a função objetivo
printf("max : ");
for (int i = 0; i < produtos; i++)
{
    if (i != produtos - 1)
        printf("%.2fx%d + ", coeficientes_obj[i], i);
    else
        printf("%.2fx%d;\n", coeficientes_obj[i], i);
}
printf("\n");
```

Os dois loops for aninhados são usados para imprimir as restrições no formato "***coeficiente1x0 + coeficiente2x1 + ... + coeficienteN-1xN-1 <= limite;***", onde os coeficientes são recuperados da matriz_produtos e os índices correspondem aos produtos. Os limites são obtidos do *array* limites_compostos, conforme o código 6.

Código 6

```
// imprimir as restrições
for (int i = 0; i < compostos; i++)
{
    for (int j = 0; j < produtos; j++)
        if (j != produtos - 1)
            printf("%.2fx%d + ", matriz_produtos[j][i], j);
        else
            printf("%.2fx%d ", matriz_produtos[j][i], j);

    printf("<= %d;\n", limites_compostos[i]);
}
```

Em resumo, o código lê informações sobre produtos, compostos, custos, limites e quantidades necessárias para fabricação, e em seguida imprime um modelo de programação linear para maximização dos lucros, com a função objetivo e as restrições correspondentes.