TRABALHO PRÁTICO 03
USO DE SOFTWARES DE OTIMIZAÇÃO LINEAR

Objetivo.

Este trabalho tem como objetivo aplicar a teoria apresentada em sala na implementação de modelos de programação linear inteira (mista) utilizando softwares comerciais para a resolução e, também, desenvolver habilidades de implementação e integração desses softwares com códigos escritos pelo próprio estudante.

Problema das Múltiplas Mochilas.

O ***Problema das Múltiplas Mochilas Binárias*** (ou 0-1) pode ser descrito da seguinte forma: dados n itens não fracionáveis cujos pesos $w_i > 0$ e benefícios $p_i > 0$ são conhecidos para todo $i = 1, \dots n$; e dadas m mochilas de capacidade limitada $b_j > 0$, também conhecidas, para todo $j = 1, \dots m$. O objetivo é escolher os itens que serão levados em cada mochila, sem que a capacidade individual das mochilas seja extrapolada, de forma a maximizar a soma total dos benefícios dos itens levados nas mochilas. Só existe uma única unidade de cada item e esses itens não podem ser fracionados.

O Problema das Múltiplas Mochilas Binárias pode ser escrito como um problema de programação linear inteira. Para isso, são definidas variáveis binárias de decisão $x_{i,j} \in \{0, 1\}$, para todo $i = 1, \dots, n$ e para todo $j = 1, \dots, m$, de forma que $x_{i,j} = 1$ se o item i for selecionado para ser levado na mochila j, ou $x_{i,j} = 0$ caso contrário. Assim, o Problema das Múltiplas Mochilas Binárias, escrito como um problema de programação linear inteira, pode ser expresso da seguinte forma:

$$\text{Maximizar} \quad \sum_{j=1}^{m}\sum_{i=1}^{n} p_i x_{i,j} \qquad (1)$$

$$\text{sujeito a:} \quad \sum_{i=1}^{n} w_i x_{i,j} \leq b_j, \qquad \forall j = 1, \dots, m \qquad (2)$$

$$\sum_{i=1}^{m} x_{i,j} \leq 1, \qquad \forall i = 1, \dots, n \qquad (3)$$

$$x_{i,j} \in \{0, 1\}, \qquad \forall i = 1, \dots n; \forall j = 1, \dots, m \qquad (4)$$

em que: a Equação (1) representa a maximização da soma dos benefícios associados aos itens selecionados para serem levados em alguma mochila; a Equação (2) define o conjunto de restrições que garante que, para cada mochila j, a soma dos pesos dos itens levados por ela não será maior do que a sua capacidade b_j; a Equação (3) define o conjunto de restrições que garante que um item i só pode ser levado em uma única mochila, caso seja levado; e, por fim, a Equação (4) define o domínio das variáveis de decisão.

Tarefa.

A partir da descrição do Problemas das Múltiplas Mochilas Binárias, apresentada acima, escreva um programa capaz de resolver uma instância desse problema utilizando o *solver* ***Gurobi***. O programa deverá ser capaz de ler os dados da instância do problema a partir de um arquivo de texto que terá o seu caminho passado para o programa como um argumento de linha de comando. O log de execução do Gurobi deve ser exibido e, além disso, ao final do processo de otimização, deve ser apresentada a solução ótima encontrada por ele, ou seja, quais itens serão levados em cada uma das mochilas.

Estrutura do Arquivo de Entrada.

O arquivo texto terá a seguinte estrutura:

- A primeira linha do arquivo contém dois valores inteiros, n e m, separados por um espaço. O valor n representa o número de itens e m representa o número mochilas.
- A segunda linha do arquivo contém m números reais, separados por espaços, representando as capacidades de cada uma das mochilas.
- As n linhas seguintes contém os dados os itens disponíveis. Cada linha é composta de dois valores reais, em que o primeiro valor representa o benefício e o segundo valor representa o peso do item.

Exemplo de Arquivo de Entrada e Saída Esperada.

O quadro abaixo apresenta o arquivo texto de entrada e um exemplo da saída esperada em que é exibido o log de execução do Gurobi e a solução ótima encontrada.

Arquivo

```
5 2
9 6
5 3
10 2
7 9
6 2
6 10
```

Exemplo de Saída

```
Academic license 2538141 - for non-commercial use only - registered to an___@cefetmg.br
Gurobi Optimizer version 11.0.3 build v11.0.3rc0 (linux64 - "Ubuntu 22.04.4 LTS")

CPU model: Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads

Academic license 2538141 - for non-commercial use only - registered to an___@cefetmg.br
Optimize a model with 7 rows, 10 columns and 20 nonzeros
Model fingerprint: 0x79d39a56
Variable types: 0 continuous, 10 integer (10 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+01]
  Objective range  [5e+00, 1e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 9e+00]
Found heuristic solution: objective 21.0000000
Presolve removed 2 rows and 3 columns
Presolve time: 0.00s
Presolved: 5 rows, 7 columns, 13 nonzeros
Variable types: 0 continuous, 7 integer (7 binary)

Root relaxation: objective 2.600000e+01, 5 iterations, 0.00 seconds (0.00 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0   26.00000    0    1   21.00000   26.00000  23.8%     -    0s
H    0     0                      23.0000000   26.00000  13.0%     -    0s

Explored 1 nodes (5 simplex iterations) in 0.00 seconds (0.00 work units)
Thread count was 4 (of 4 available processors)

Solution count 2: 23 21

Optimal solution found (tolerance 1.00e-04)
Best objective 2.300000000000e+01, best bound 2.300000000000e+01, gap 0.0000%

----------------------------------------------------
Solução Ótima
----------------------------------------------------
Valor da Função Objetivo: 23.0
Itens na Mochila 0: [2]
Itens na Mochila 1: [1, 3]
```

Critérios de Avaliação.

- Uso da API do Gurobi feito de forma correta.
- A solução ótima apresentada estar correta.
- Capacidade do programa em ler o arquivo de entrada corretamente.
- Capacidade do programa em apresentar a saída corretamente.
- Qualidade de código, incluindo a legibilidade (indentação, nomes de variáveis, ...) e comentários.

Formato de Entrega.

Deve ser entregue um único arquivo compactado no formato ZIP. O nome do arquivo compactado deve seguir o padrão `nome-sobrenome-matricula.zip`, por exemplo `andre-maravilha-123456.zip`. Esse arquivo deverá ter o seguinte conteúdo:

- Um diretório `src` com o código-fonte.
- Um arquivo `LEIAME.txt` com instruções de compilação (no caso de linguagens compiladas) e execução.