
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

DEPARTAMENTO DE MATEMÁTICA E ESTATÍSTICA – DEINFO

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

MARCONE CARLOS DE SANTANA SILVA

# PROGRAMAÇÃO INTEIRA

Recife

2019

MARCONE CARLOS DE SANTANA SILVA

# PROGRAMAÇÃO INTEIRA

Trabalho para a Disciplina Tópicos de
Otimização para obtenção da nota da 2º V.A

Professor Responsável: Cláudio Tadeu Cristino

Recife

2019

# Sumário

# 1. INTRODUÇÃO

A Programação Matemática é o ramo da Pesquisa Operacional que trata de métodos de otimização (minimização ou maximização) de uma função objetivo com um número finito de variáveis de decisão sujeita a certas restrições. Estas restrições podem ser de origem financeira, tecnológica, marketing, organizacional ou outras.

De um modo geral, Programação Matemática pode ser definida como uma representação matemática dedicada à programação ou planejamento da melhor possibilidade de alocação de recursos escassos (BRADLEY, HAX e MAGNANTI, 1977). A Programação Matemática utiliza técnicas e algoritmos para solucionar problemas modelados matematicamente.

## 2. PROGRAMAÇÃO LINEAR

Programação Linear (PL) é uma das mais importantes e mais utilizadas técnicas de Pesquisa Operacional. A simplicidade do modelo envolvido e a disponibilidade de uma técnica de solução programável em computador como o método Simplex descrito por Dantzig (1963) facilitam sua aplicação. Esta técnica é amplamente utilizada, pois possui habilidade para modelar importantes e complexos problemas de decisão e o método Simplex pela capacidade de produzir soluções rapidamente.

Um problema de PL é composto por:

1. Uma função linear formada com as variáveis de decisão chamada de Função Objetivo, cujo valor deve ser otimizado;

2. Relações de interdependência entre as variáveis de decisão que se expressam por um conjunto de equações ou inequações lineares, chamadas de restrições do modelo;

3. Variáveis de decisão que devem ser positivas ou nulas.

Modelo Formal

$$\text{maximize (ou minimize)}$$

$$z = \sum_{j \in N} c_j x_j,\ N = \{1,\ldots,n\}$$

$$\text{sujeito a}$$

$$\sum_{j \in N} a_{ij} x_j\ (\leq, = ou \geq)\ b_i,\ i \in M = \{1,2,\ldots,m\}$$

$$x_j \geq 0,\ j \in N$$

$c_j$, $a_{ij}$ *e* $b_i$ são constantes conhecidas para todo *i* e *j*, e $x_j$ são variáveis não negativas.

As restrições do problema podem ser transformadas em equações adicionando-se uma variável de folga (não negativa) $x_{n+1}$, se a i-ésima desigualdade é do tipo ≤ e subtraindo uma variável de folga (não negativa), $x_{n+k}$, se a k-ésima desigualdade é do tipo ≥. Considerando que ao serem acrescentadas as variáveis de folga, obtém-se um total de m + n variáveis, pode-se escrever o problema na forma matricial.

$$
\begin{aligned}
&\text{maximize (ou minimize)} \\
&\quad z = cx \\
&\text{sujeito a} \\
&\quad Ax = b \\
&\quad x \geq 0
\end{aligned}
$$

no qual, *c* é um vetor linha de ordem *(m+n)* , *A* é uma matriz *m × (m + n)*, *x* é um vetor coluna de ordem *(m + n)* e *b* é um vetor coluna de ordem *m*.

## 3. PROGRAMAÇÃO INTEIRA

A Programação Inteira pode ser entendida como uma caso específico da Programação Linear, onde as variáveis devem ser inteiras (ou ao menos, parte destas variáveis). A rigor, o nome mais correto para a Programação Inteira é Programação Linear Inteira.

Alguns problemas reais requerem o uso de variáveis que assumem somente valores inteiros. Quando isto acontece tem-se um problema de Programação Inteira (PI). este problema está definido na formulação a seguir:

$$
\begin{aligned}
&\text{maximize (ou minimize)} \\
&\quad z = g_0(x_1, x_2, \ldots, x_n) \\
&\text{sujeito a} \\
&\quad g_i(x_1, x_2, \ldots, x_n)(\leq ou = ou \geq) b_i,\ i \in M = \{1,2,\ldots,m\} \\
&\quad x_j \geq 0,\ j \in N = \{1,2,\ldots,n\} \\
&\quad x_j \text{ inteira},\ j \in I \subseteq N
\end{aligned}
$$

no qual, $x_j$, $j \in N$ são as variáveis, $g_i$, $i \in M \cup \{0\}$ são funções das variáveis $x_1$, $x_2$,...,$x_n$, e $b_i$, $i \in M$ são constantes conhecidas. Se $I = N$, isto é, todas as variáveis são inteiras, então o problema é dito de PI. Caso contrário, se $I \subset N$, então chama-se de problema de Programação Inteira Mista (PIM). Em muitos dos problemas abordados em PI, as funções $g_i$, $i \in M \cup \{0\}$ são lineares e o modelo pode então ser descrito como mostra a formulação :

$$
\begin{aligned}
&\text{maximize (ou minimize)} \\
&\quad z = \sum_{j \in N} c_j x_j,\ N = \{1,\ldots,n\} \\
&\text{sujeito a} \\
&\quad \sum_{j \in N} a_{ij} x_j (\leq, \geq ou =) b_i,\ i \in M = \{1,\ldots,m\} \\
&\quad x_j \geq 0,\ j \in N \\
&\quad x_j \text{ inteira},\ j \in I \subseteq N
\end{aligned}
$$

onde $x_j$ são variáveis não negativas e $c_j$, $a_{ij}$ *e* $b_i$ são constantes conhecidas, para todo *i* e *j*. Se *I = N*, isto é, todas as variáveis são inteiras, então temos um problema Programação Linear Inteira (PLI). Se *I* $\subset$ *N*, então o problema é de Programação Linear Inteira Mista (PLIM).

Muitos modelos práticos de PLI restringem algumas das variáveis inteiras para valores "0" ou "1" e, neste caso, tem-se um problema de Programação Linear Inteira Binária (PLIB). Estas variáveis são usadas para decisão: sim ("1") e não ("0").

Mesmo sendo bastante estudada, ainda não foi desenvolvido um algoritmo eficiente para resolver um PLI de tamanho razoável, ou seja, não existe um algoritmo que o resolva com um número de passos polinomial no tamanho da entrada. De fato, resolver um PLI é um problema da classe NP-Completo. Acredita-se que não exista solução eficiente para um problema dessa classe e qualquer solução correta do problema necessita de tempo exponencial no pior caso. Contudo, existem dois métodos aceitáveis, em termos computacionais.

## 3.1. EXEMPLOS DE APLICAÇÕES

Utilização de Equipamentos ⇒ $x_j$ representa a quantidade de equipamentos. Por exemplo, $x_j$ = 2.33 navios petroleiros pode não ter significado prático.

2) Tamanhos de Lotes ⇒Em algumas situações de planejamento de produção, faz-se necessário que $x_j$ = 0 ou $x_j$ > $l_j$, onde $x_j$ representa a quantidade de produtos produzidos e $l_j$ uma quantidade mínima de produtos $x_j$ para compor um lote. Esta situação é um exemplo de restrição "ou-ou" (ou faz um mínimo ou não faz nada).

3) Decisões "Sim-ou-Não"⇒ $x_j$ = 1 ou $x_j$ = 0 representando decisões sim ou não(também uma situação de "ou-ou"). Por exemplo, $x_j$ = 1 representa construir uma nova fábrica.

4) Otimização Combinatória ⇒lida com problemas de decisão de sequências, programas e itinerários. Exemplos clássicos são:

- Problema do Caixeiro Viajante ⇒Para n cidades, existem *(n-1)!* diferentes percursos (ex: n = 50, 6x1062percursos diferentes).
- Problema de Programação de Máquinas ⇒Para n itens a serem fabricados em cada uma de k máquinas existem *(n!) k* sequências possíveis (ex*: n = k* = 10, 4x1065seqüências diferentes).

## 3.2. ALGUNS PROBLEMAS TÍPICOS DE PI

### 3.2.1. PROBLEMA DE ALOCAÇÃO

Problema de Alocação (em inglês, Assignment Problem) é conhecido por este nome por ser a representação de inúmeras situações em que é necessário alocar pessoas a lugares, a tarefas ou a zonas de trabalho, máquinas a tarefas, etc. Aparece muitas vezes como se tratasse de um problema de PL mas, como veremos, as suas variáveis de decisão são binárias. Suponhamos que se pretende alocar n indivíduos a n tarefas, sabendo que a medida de eficiência de alolcar o indivíduo *i* à tarefa *j* é $c_{ij}$que tanto pode representar um lucro como um custo). Pretende-se determinar a alocação dos indivíduos às tarefas de modo a otimizar a eficiência total. As variáveis de decisão são as seguintes:

$$X_{ij} = \begin{cases} 1, se\ o\ indivíduo\ i\ for\ Alocado\ à\ tarefa\ j \\ 0, se\ o\ infivíduo\ i\ não\ for\ alocado\ à\ tarefa\ j \end{cases}$$

i = 1,...n  j= 1,...,n

O modelo de PLI é como se segue:

**min** ou **max** $\left(\sum_i \sum_j c_{ij} \cdot X_{ij}\right)$ → optimizar a eficiência total

**suj. a:**

1. cada indivíduo só pode estar alocado a uma tarefa:
$$\sum_j X_{ij} = 1, \quad \forall_{i:\ i=1,2,3,4}$$

2. cada tarefa só deve ser desempenhada por um indivíduo:
$$\sum_i X_{ij} = 1, \quad \forall_{j:\ j=A,\ B,\ C,\ D}$$

$X_{ij} \in \{0, 1\}$

Trata-se, como se pode ver, de um modelo muito simples e cujo sistema de restrições tem uma estrutura particular com certas propriedades. Situações em que o número de indivíduos é diferente do número de tarefas podem também ser representadas. Um problema que seja representado por um modelo com esta estrutura chama-se problema de alocação, independentemente da situação que estiver a ser considerada. Conforme veremos mais adiante, os problemas de alocação dispõem de um método de resolução próprio graças à sua estrutura especial.

### 3.2.2. PROBLEMA DO CAIXEIRO VIAJANTE

O Problema do Caixeiro Viajante (em inglês, Traveling Salesman Problem) é outro tipo de problema que pode ser representado por um modelo de PLI. Este problema é facilmente visto numa rede, em que as cidades correspondem aos nós ou vértices e os arcos representam as ligações entre as cidades. Consiste em encontrar um circuito que liga todas as cidades, ou seja, um conjunto de arcos que, partindo de um determinado vértice, passa por todos os outros uma e uma só vez e termina no vértice de partida. Conhecendo-se a distância ou o custo entre cada par de cidades (vértices), pretende-se determinar o circuito óptimo.

As variáveis de decisão são as seguintes:

$$X_{ij} \begin{cases} 1, se\ a\ cidade\ j\ é\ visitada\ imediatamente\ apíes\ a\ cidade\ i \\ 0\ , se\ a\ cidade\ j\ não\ é\ visitada\ imediatamente\ após\ a\ cidade\ i \end{cases}$$

I = 1, ..., n  j = 1,...,n

O modelo de PLI é como se segue:

**min** $\left(\sum_{i}^{n}\sum_{j}^{n} d_{ij} \cdot X_{ij}\right)$ →minimizar o percurso total

**suj. a:**

(1) cada uma das cidades é visitada uma e só uma vez, ou seja, cada vértice é entrado uma só vez e saído uma só vez:

$$\sum_{j}^{n} X_{ij} = 1, \quad \forall_{\text{i: i=1,..,n}}$$

$$\sum_{i}^{n} X_{ij} = 1, \quad \forall_{\text{j: j=1,..,n}}$$

(2) entre dois quaisquer subconjuntos complementares de cidades (S e $\overline{S}$) há pelo menos um arco de ligação:

$$\sum_{i \in S}\sum_{j \in \overline{S}} X_{ij} \geq 1, \quad \forall_{S \subset \text{ conjunto total das cidades a visitar}}$$

### 3.2.3. PROBLEMA DA MOCHILA

Outro problema típico de PLI é o Problema da Mochila (Knapsack Problem eminglês). A situação representada por este problema é a seguinte: um alpinista dispõe de uma mochila e de diversos objetos que podem ser "carregados" na mochila; para cada objeto é conhecido o seu peso (pi) e o benefício que dele se tira (vi). Conhecida a capacidade da mochila em termos de peso máximo, quais os objetos, dos n existentes, que devem ser escolhidos para colocar na mochila?

Repare-se que o problema seria extremamente simples de resolver, não fora a condição de as variáveis apenas poderem assumir os valores 0 (objeto não escolhido) ou 1 (objeto escolhido). Na realidade as variáveis de decisão são as seguintes:

$$X_{ij} \begin{cases} 1, se\ o\ objeto\ i\ for\ escolhido \\ 0, se\ o\ objeto\ i\ não\ for\ escolhido \end{cases}$$

I = 1 , ... n

**max** $\left(\sum_{i=1}^{n} v_i \cdot X_i\right)$ →maximizar o proveito global

**suj. a:**

não se ultrapassar o peso máximo: $\sum_{i=1}^{n} p_i \cdot X_i \leq P_{max}$

$X_i \in \{0, 1\}$

## 4. MÉTODOS DE RESOLUÇÃO DE PROBLEMAS DE PROGRAMAÇÃO INTEIRA

### 4.1. MÉTODO DE PARTIÇÃO E AVALIAÇÃO SUCESSIVAS

O método "Branch and Bound" (literalmente, método de ramificação e limitação)consiste na partição (ramificação) sucessiva do conjunto de soluções possíveis do problema de PLI em subconjuntos e na limitação (avaliação) do valor óptimo da função objetivo (limite inferior se se tratar de maximização, ou superior se se tratar de minimização), de modo a excluir os subconjuntos que não contenham a solução óptima. Partindo da constatação de que se, na solução óptima da relaxação linear dum problema de PLI, as variáveis tomam valores inteiros, então essa solução é a solução óptima do PLI, começa-se por resolver a relaxação linear do PLI inicial: se as variáveis que no problema de PLI são inteiras tomam, na solução óptima do PL, valores inteiros, então foi encontrada a solução óptima do PLI; caso contrário, divide-se o problema de PL em dois, através da introdução de restrições adicionais que fazem a partição do conjunto das soluções possíveis. Vão-se então resolvendo sucessivos problemas de PL, estabelecendo-se limites para o valor óptimo da função objetivo e, assim, eliminando diversos subconjuntos, até se alcançar a solução óptima do PLI.

### 4.2. MÉTODO DOS PLANOS DE CORTE

O método dos Planos de Corte (Cutting Planes, em inglês) foi o primeiro método a ser desenvolvido e deve-se a Gomory (1958). Consiste em introduzir sucessivamente novas restrições na relaxação linear do PLI, restrições essas que cortam o conjunto das soluções possíveis eliminando algumas delas e a própria solução óptima do PL (por isso se chamam planos de corte), sem contudo eliminar qualquer solução inteira.

## 5. PROBLEMA SOLUCIONADO

### 5.1. GERÊNCIA DE ENERGIA

Neste problema faremos gerenciamento de energia por meio do desligamento de servidores. Temos um conjunto {1…V} de VMs, sendo que cada VM *i* possui apenas a sua respectiva demanda de capacidade $d_i$. Temos também um conjunto de servidores {1…M}, com cada servidor *j* possuindo sua respectiva capacidade $c_j$. Um servidor pode ou não estar ligado, o que é modelado por meio da variável de decisão booleana $y_j$ , com 1 representando o servidor ligado e 0 desligado. A alocação da uma VM i em um servidor j é novamente modelada pela variável de decisão booleana $x_{ij}$ que assume 1 no caso da máquina virtual i estar alocada no servidor *j*, e 0 caso contrário. Assim, o que desejamos é minimizar o número de servidores ativos.

$$minimizar \sum_{i=1}^{V} y_j$$

Sujeito às seguintes restrições

$$\sum_{i=1}^{V} d_i \, x_{ij} \leq y_j c_j, para\ cada\ servidor\ j$$

$$\sum_{i=1}^{V} x_{ij} = 1\ para\ cada\ VM\ i$$

$$x_{ij} \in \{0,1\}, para\ todo\ i\ e\ j$$

$$y_{ij} \in \{0,1\}, para\ todo\ j$$

### 5.2. FERRAMENTA E SOLUÇÃO

Para a solução do problema foi usado o a linguagem de programação Python em interface com o solver do software Gurobi. No problema foram colocadas variáveis de capacidade de 20 servidores e variáveis de demanda de 90 maquinas virtuais. O script com o código da solução necessita da instalação da biblioteca *gurobipy* e da biblioteca para ler os dados da planilha *xlrd.* O projeto está disponível no Github em: https://github.com/marconeSantanaUFRPE/Otimizacao

## 6. REFERÊNCIAS BIBLIOGRÁFICAS


Nogueira, Fernando. Programação Inteira Disponível em <http://www.ufjf.br/epd015/files/2010/06/ProgramacaoInteira.pdf>.

Carvalho, Ricardo. Aplicações de Programação Linear Inteira Mista à problemas de futebol, 2015, Disponível em: <https://www.ime.usp.br/~map/tcc/2015/RicardoCarvalho.pdf>.

P.M. Mateo , David Lahoz. , Programação Linear Inteira, 2009 Disponível em : <https://ocw.unizar.es/ocw/ensenanzas-tecnicas/modelos-de-investigacion-operativa/ficheros/OCWProgEntera.pdf>

Alves, Rui. Delgado, Catarina. Programação Linear inteira. 1997, Disponível em: <https://repositorio-aberto.up.pt/bitstream/10216/74369/2/40539.pdf>