1. INTRODUÇÃO

A programação linear é amplamente utilizada para otimização de processos, seja para maximizar ou minimizar uma função linear de variáveis, chamada de função-objetivo. O desenvolvimento de técnicas algébricas para a solução de inequações lineares é algo com bastante tempo, no final da década de 1940 o matemático George Dantzing criou o primeiro algoritmo simplex. No mesmo período, outro matemático desenvolveu a teoria da dualidade que teve implicações na Economia.

A representação de modelos tem uma importância fundamental ao estudar programação linear e que para isso o entendimento de **função objetivo**, as **restrições** e o **tipo de otimização** se faz necessário para a utilização de um problema. Alguns problemas de programação linear pode-se utilizar um tipo solução chamada de gráfica que analisa o domínio da função-objetivo em um plano, mas a solução geral para a resolução de problemas é pela forma algébrica.

2. CONCEITOS

A otimização é a alocação de recursos que por muitas das vezes limitadas entre atividades que concorrem pela mesma atividade procurando atender um certo objetivo. As escolhas sistemáticas de valores de um problema

Algumas definições são fundamentais para o entendimento do simplex, os teoremas abaixo são parte para o embasamento matemático do algoritmo.

1. TEOREMA: "O conjunto de todas as soluções compatíveis de um problema de programação linear é um conjunto convexo".
2. TEOREMA: "Toda solução básica compatível do sistema Ax = b é um ponto extremo do conjunto convexo C ".
3. TEOREMA: "Se a função objetivo possui máximo (ou mínimo) finito, então existe um ponto extremo do conjunto convexo C, que produz o valor ótimo da função".
4. TEOREMA: "Se a função objetivo assume valor máximo (mínimo) em mais de um ponto extremo, então a função tem o mesmo valor para qualquer combinação convexa desses pontos extremos".

As restrições são as limitações presentes nos recursos dos problemas e até mesmo de limitações físicas que se impõem às variáveis descritivas das atividades.

3. MÉTODO SIMPLEX

O método Simplex é um processo iterativo que permite melhorar a solução da função objetivo em cada etapa. O processo finaliza quando não é possível continuar melhorando este valor, ou seja, quando se obtenha a solução ótima (o maior ou menor valor possível, segundo o caso, para que todas as restrições sejam satisfeitas).

O algoritmo foi desenvolvido para solução de problemas envolvendo **maximização** na condição **menor que**, com isso o usuário informará a solução básica compatível para o melhor funcionamento do programa. Logo abaixo, mostra-se como deve ser formato de matriz aumentada em que o arquivo deve estar para que o programa possa fazer o upload e resolvê-lo.

As questões estão escritas conforme a estrutura abaixo:

$$z = a_{11}\mathbf{x_1} + a_{21}\mathbf{x_2} + \ldots + a_{1n}\mathbf{x_n}$$
$$a_{21}\mathbf{x_1} + a_{22}\mathbf{x_2} + \ldots + a_{2n}\mathbf{x_n} = \mathbf{b_2}$$
$$a_{m1}\mathbf{x_1} + \ldots + a_{mn}\mathbf{x_n} = \mathbf{b_m}$$

Após, estruturar em forma de matriz aumentada segue a matriz 1 abaixo:

Matriz 1. Exemplo.

$$\begin{bmatrix} z & a11 & \cdots & a1n & 0 \\ 0 & a21 & a22 & a2n & b2 \\ \vdots & a31 & \ddots & \vdots & b3 \\ 0 & am1 & \cdots & amn & bm \end{bmatrix}$$

Após formatar o problema, o algoritmo pedirá para o usuário selecionar o arquivo no diretório específico do computador onde está a questão, e depois no console do Scilab aparecerá o resultado do **z ótimo** e posterior pedirá para salvar em arquivo o resultado do Simplex em forma de matriz aumentada.

3.1. ALGORITMO EM SCILAB

O programa importa a questão no formato .txt utilizando a função *uigetfile()* que o usuário pré-formato em matriz aumentada do problema. A função *fscanfMat()* ler uma matriz em arquivo .txt proveniente da função *uigetfile()*.

```
arq = uigetfile("*.txt",pwd(),"Escolha um arquivo: ");
matriz_aumentada = fscanfMat(arq);
```

O algoritmo possui uma função chamada simplex que tem como entrada uma matriz e exibe os resultados da matriz de coeficientes, vetor solução e o valor ótimo da função objetivo.

```
function metodo = simplex(matriz_aumentada)
```

As variáveis abaixo são para separar a matriz de coeficientes e vetor solução, cria um vetor de fator limitante e outro de variáveis básicas.

```
dim = size(matriz_aumentada);
A = matriz_aumentada(:,1:dim(2)-1);
b = matriz_aumentada(:,dim(2));
q = zeros(1,length(b)-1);
linha1 = A(1,:);
vb = zeros(1,length(b)-1);
```

O programa adotou a seguinte estratégica de finalizar as interações para o valor ótimo:

1. Cria uma variável auxiliar;
2. Dentro de um laço de repetição faz as interações;
3. A cada interação procura se há valores negativos na primeira linha da matriz A;
4. Caso não tenha, recebe o tamanho mais 1 para o laço while se torna falso.

```
m = -1;
while m < length(linha1)

negativos = find(linha1 < 0)
if length(negativos) <= 0 then
m = length(A(1,:)) + 1;
end

end
```

O passo a seguir é para a escolha da variável para entrar na base. Para isso, o processo de maximização precisa achar o endereço em que tenha o menor valor da primeira linha da matriz A, mas pode acontecer de ter o menor mais de uma vez que para isso a escolha se faz do primeiro endereço da esquerda para direita.

```
a = min(linha1);
index = find(linha1 == a);
if length(index) > 1 then
index = index(1);
end
```

A escolha de tirar uma variável da base se faz da razão de elemento a elemento entre o vetor b e a coluna que possui o menor valor da linha 1 da matriz A. Caso haja divisão por zero a posição do vetor q (fator limitante) recebe um valor alto de 100000 para não interferir no passo adiante.

```
for i = 1:length(b)-1
if A(i+1,index) <= 0 then
q(i) = 100000;
else
q(i) = b(i+1) / A(i+1,index);
end
end
```

A escolha do menor valor do fator limitante é essencial para o processo de “escalonamento” da matriz aumentada. O vetor de variável básica é indexada através do índice do menor fator limitante com atribuição do posição do menor valor da primeira linha.

```
ql = min(q);
l = find(q == ql);
vb(l) = index - 1;
```

O código abaixo faz a operação mais importante do método simplex que percorre todas as linhas e colunas a fim de zerar os elementos e transformar as variáveis não básicas em básicas.

```
for j = 1: length(A(:,1))
if j == l+1 then
A(j,:) = A(j,:);
b(j) = b(j);
else
pivo = (A(j,index)/A(l+1,index))
b(j) = b(j) - pivo*b(l+1);
for k = 1:length(linha1)
A(j ,k) =  A(j,k) - pivo * A(l+1,k);
end
end
end
```

Ao final das interações os resultados são salvos em arquivo.txt e mostrados no console do Scilab.

```
printf('O valor otimo de z = %5.1f',b(1));
disp('Os índices das variáveis básicas: ');
disp(vb);
salvar(A,b);

endfunction
```

A função *salvar()* recebe uma matriz e um vetor como entrada e exporta em formato arquivo.txt no diretório do computador escolhido pelo usuário.

```
function guardar = salvar(B,c);

a = [B c];
arq = uigetfile("*.txt",pwd(),"Escolha um arquivo: ");
fprintfMat(arq,a,"%5.1f");
endfunction
```

4. APLICAÇÕES

As questões a seguir são parte da 1º Lista de Exercícios em que possuem numeração e o enunciado. Os resultados foram obtidos a partir do algoritmo simplex desenvolvido no Scilab.

Os problemas propostos são de programação sendo de maximizar com restrições menor que tendo ou não variáveis sem restrições de sinal.

Para se aplicar o método Simplex, é necessário que o problema satisfaça os requisitos:

a) Todas as variáveis são não negativas (só podem assumir valores positivos ou nulos);
b) Todas as restrições são equações (ou restrições do tipo '=');
c) Todos os termos independentes são positivos.

Após o programa compilar, os resultados são apresentados no console do Scilab, bem como um arquivo em formato .txt na forma de matriz aumentada.

2. Resolva o seguinte problema de programação linear:

Máx. Z = $3\mathbf{x_1} + 2\mathbf{x_2} + 5\mathbf{x_3}$ sujeita a

$2\mathbf{x_1} + 3\mathbf{x_2} + 4\mathbf{x_3} \leq 10$

$5\mathbf{x_1} + 6\mathbf{x_2} + 2\mathbf{x_3} \leq 12$

e

$\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3} \geq 0$.

A matriz 2 está na forma de matriz aumentada proveniente padronização do modelo canônico para resolver o problema.

Matriz 2. Questão 2.

$$\begin{bmatrix} 1 & -3 & -2 & -5 & 0 & 0 & 0 \\ 0 & 2 & 3 & 4 & 1 & 0 & 10 \\ 0 & 5 & 6 & 2 & 0 & 1 & 12 \end{bmatrix}$$

Resolução:

Após o método simplex resolver o problema a matriz 3 apresenta os resultados finais.

Matriz 3. Solução da questão 2.

$$\begin{bmatrix} 1 & 0 & 2.31 & 0 & 1.19 & 0.13 & 13.38 \\ 0 & 0 & 0.18 & 1 & 0.31 & -0.12 & 1.62 \\ 0 & 1 & 1.12 & 0 & -0.12 & 0.25 & 1.75 \end{bmatrix}$$

O valor ótimo de **zo** = 13.4.

As variáveis básicas:

x_3 = 13/8
x_1 = 7/4

Números de interações: 2

5. Resolva o seguinte problema de programação linear:

Máx. $Z = 4x_1 + 2x_2 + 2x_3$ sujeita a

$x_1 + x_2 + 2x_3 \leq 4$

$4x_1 - 5x_2 + 3x_3 \leq 30$

e $x_1, x_3 \geq 0$.

Notar que x_2 não tem restrição de sinal.

Matriz 4. Questão 3.

$$\begin{bmatrix} 1 & -4 & -2 & 2 & -2 & 0 & 0 & 0 \\ 0 & 1 & 1 & -1 & 2 & 1 & 0 & 4 \\ 0 & 4 & -5 & 5 & 3 & 0 & 1 & 30 \end{bmatrix}$$

Resolução:

Matriz 5. Resolução da questão 3.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 4.89 & 3.11 & 0.22 & 19.11 \\ 0 & 1 & 0 & 0 & 1.44 & 0.56 & 0.11 & 5.56 \\ 0 & 0 & 1 & -1 & 0.55 & 0 & 0.11 & 1.55 \end{bmatrix}$$

O valor ótimo: **zo** = 19.11.

As variáveis básicas:

$\mathbf{x_1} = 5.56$
$\mathbf{x_2} = 26/8$

Números de interações: 2

7. Resolva, pelo método simplex, o seguinte problema de programação linear:

Máx. $Z = \mathbf{x_1} + 3\mathbf{x_2}$ sujeita a

$-\mathbf{x_1} + 2\mathbf{x_2} \leq 4$

$\mathbf{x_1} + \mathbf{x_2} \leq 6$

$\mathbf{x_1} + 3\mathbf{x_2} \leq 9$

e $\mathbf{x_1}, \mathbf{x_3} \geq 0$.

Matriz 6. Questão 7.

$$\begin{bmatrix} 1 & -1 & -3 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 & 4 \\ 0 & 1 & 1 & 0 & 1 & 0 & 6 \\ 0 & 1 & 3 & 0 & 0 & 1 & 9 \end{bmatrix}$$

Resolução:

Matriz 7. Questão 7.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 9 \\ 0 & 0 & 1 & 0.2 & 0 & 0.2 & 2.6 \\ 0 & 0 & 0 & 0.4 & 1 & -0.6 & 2.2 \\ 0 & 1 & 0 & -0.6 & 0 & 0.4 & 1.2 \end{bmatrix}$$

O valor ótimo: **zo** = 9.

As variáveis básicas:

x_2 = 2.6
x_4 = 1.2
x_3 = 2.2

Números de interações: 2

5. CONCLUSÃO

O algoritmo simplex mostrou-se uma forma bastante eficiente para se resolver problemas de programação linear que envolvam problema de questões concorrentes. O desenvolvimento do código permitiu o aprendizado do simplex de forma mais genérica, pois tem que ser capaz de resolver qualquer tipo de problema com as limitações já citadas.

6. REFERÊNCIA

"Programação linear – Wikipédia, a enciclopédia livre." Acessado em agosto 20, 2019. https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_linear.

"Pesquisa operacional/Introdução à Programação Linear - Wikilivros." https://pt.wikibooks.org/wiki/Pesquisa_operacional/Introdu%C3%A7%C3%A3o_%C3%A0_Programa%C3%A7%C3%A3o_Linear. Acessado em 20 ago. 2019.

"Linear programming: Simplex method example - PHPSimplex." http://www.phpsimplex.com/en/simplex_method_example.htm. Acessado em 20 ago. 2019.

7. APÊNDICE

```
clc
clear all

arq = uigetfile(); // Função para pegar um arquivo
matriz_aumentada = fscanfMat(arq);// pega o arquivo de formato especificado

function metodo = simplex(matriz_aumentada)

dim = size(matriz_aumentada);// vetor que tem a qnt de linhas por qnt de colunas
A = matriz_aumentada(:,1:dim(2)-1);// Matriz de coeficientes
b = matriz_aumentada(:,dim(2));// vetor de resultados
q = zeros(1,length(b)-1); // vetor do Fator Limitante
linha1 = A(1,:);// primeira linha da matriz de coeficientes
vb = length(q)-1:1:length(linha1)-1;

interacao = 0;

m = -1;// para iniciar o while

while m < length(linha1)

a = min(linha1); // menor valor da primeira linha  (variavel nao basica)

//***** SÓ PRA SABER O INDICE DE UMA LINHA DE MATRIX
interacao = interacao + 1;
index = find(linha1 == a);

//*****************
if length(index) > 1 then
index = index(1);
end

// tá bom para achar o vetor de fator limitante
for i = 1:length(b)-1

if A(i+1,index) <= 0 then // tanto divisão por zero e o denominador negativo deve
ser excluido
q(i) = 100000; // VALOR BEM GRANDE PARA SER EXCLUÍDO NA FUNÇÃO min
else
q(i) = b(i+1) / A(i+1,index);
end
end

ql = min(q);// Fator Limitante
l = find(q == ql); // índice  que identifica a posição no conjunto B que deve deixar
a base.
vb(l) = index - 1;
//************** ROLE PRA ZERAR ELEMENTOS ******************
for j = 1: length(A(:,1))

if j == l+1 then
A(j,:) = A(j,:);
b(j) = b(j);
else
pivo = (A(j,index)/A(l+1,index));
```

```
b(j) = b(j) - pivo*b(l+1);

for k = 1:length(linha1)
A(j ,k) =  A(j,k) - pivo * A(l+1,k);
end
end
end
//*********************

// ****** PARA DECIDIR A HORA DE ACABAR ******
linha1 = A(1,:);
negativos = find(linha1 < 0);
if length(negativos) <= 0 then
m = length(A(1,:)) + 1;
end

//****************************************
end
printf('O valor ótimo de z = %5.1f',b(1));
printf('\n Numero de interações = %5.1f',interacao);
disp('Os indices da base: ');
disp(vb(1:interacao+1));
disp('Matriz de coeficientes: ');
disp(A);
disp('Vetor solução: ');
disp(b);
salvar(A,b);

metodo = 0;
endfunction
simplex(matriz_aumentada);

function guardar = salvar(B,c)

a = [B c];
arq = uigetfile();

fprintfMat(arq,a,"%5.1f");
guardar = 0;

endfunction
```