SPOLM 2008

ISSN 2175-6295 Rio de Janeiro- Brasil, 05 e 06 de agosto de 2008.

IMPLEMENTAÇÃO E APLICAÇÃO DO MÉTODO SIMPLEX NO "R PROJECT"

Amanda da Silva Lira
UFRPE – Universidade Federal Rural de Pernambuco
Rua Dom Manuel de Medeiros s/n, 50171-900 – Recife, PE, Brasil
amandaslest@yahoo.com.br

Alessandro Henrique da Silva Santos
UFRPE – Universidade Federal Rural de Pernambuco
Rua Dom Manuel de Medeiros s/n, 50171-900 – Recife, PE, Brasil
alessandrohss@yahoo.com.br

RESUMO

A Programação Linear é a parte da Pesquisa Operacional desenvolvida, inicialmente, por grupos acadêmicos que assessoraram as forças militares durante a 2ª Guerra Mundial a fim de encontrar soluções voltadas para o emprego eficiente do radar, uso de canhões anti-aéreos, táticas de bombardeio a submarinos, escolta navais e etc. Com o fim da guerra, os conhecimentos desenvolvidos sofreram manutenções mas redirecionadas para desafios ligados a engenharia civil. O marco definitivo da Pesquisa Operacional foi a publicação do Método Simplex, por G. Dantzig, em 1947. Desde então, a programação linear permanece até hoje como a mais básica e útil de todas as técnicas da Pesquisa Operacional. Em meio aos avanços da programação linear e da informática, o método simplex passou a ser implementado em diversas plataformas de análise de dados. Dentre essas plataformas, o programa gratuito R project o qual possui fácil acesso e manipulação, ainda não possuí tal método de programação linear. Este trabalho apresenta o algoritmo do método simplex implementado na plataforma R project e, além disso, faz diversas aplicações através de exemplos retirados de livros didáticos. Ao fim do trabalho é apresentado o código fonte do algoritmo utilizado.

Palavras-Chaves: Programação Linear, Método Simplex, R project.

ABSTRACT

The Linear Programming is the part of the developed Operational Research, initially, for academic groups that advised the military forces during to 2nd World war in order to find solutions gone back to the efficient job of the radar, use of anti-aerial cannons, bombing tactics to submarines, it escorts naval and etc. With the end of the war, the developed knowledge suffered maintenances but redirected for linked challenges the civil engineering. The definitive mark of the Operational Research was the publication of the Simplex Method, by G. Dantzig, in 1947. Ever since, the linear programming stays until today as the most basic and useful of all the techniques of the Operational Research. Amid the progresses of the linear programming and of the computer science, the simplex method passed to be implemented in

several platforms of analysis of data. Among those platforms, the program free R project which possesses easy access and manipulation, still didn't possess such method of linear programming. This work presents the algorithm of the simplex method implemented in the platform R project and makes several applications through solitary examples of text books. To the end of the work the code source of the used algorithm is presented.
Keywords: Linear programming, Simplex method; R project.

1. INTRODUÇÃO

A Pesquisa Operacional foi a denominação dada ao conjunto de processos e métodos de análise de dados desenvolvidos por grupos acadêmicos que assessoraram as forças militares durante a 2ª Guerra Mundial. Tais grupos foram, inicialmente, criados na Inglaterra, com o objetivo de especular sobre problemas considerados novos que escapavam às rotinas bélicas, tanto no plano estratégico como tático. O primeiro desses grupos foi constituído por uma equipe integrada por três fisiologistas, dois físico-matemáticos, um astrofísico, um militar, um agrimensor, um físico e dois matemáticos. Dentre os problemas estudados pelos diversos grupos incluem-se: o emprego eficiente do radar, uso de canhões anti-aéreos, táticas de bombardeio a submarinos, escolta navais, etc. A surpreendente eficácia desses grupos deveu-se mais a engenhosidade usada para coletar dados e informações para subseqüentes análise do que, propriamente, à geração de técnicas específicas. Esta fórmula de abordar problemas complexos estimulou a disseminação desses grupos e, ao final da guerra, determinou sua manutenção, mas redirecionados para desafios ligados à gerência civil. O marco definitivo na afirmação da Pesquisa Operacional foi a publicação por G. Dantzig, em 1947, do método simplex para a programação linear. Assim, a programação linear se tornou a primeira técnica explícita e permanece hoje como a mais básica e útil de todas as técnicas da Pesquisa Operacional (PUCCINI e PIZZOLATO[1]).

Em meio aos avanços da programação linear e da informática, o método simplex passou a ser implementado em diversas plataformas de análise de dados. Dentre essas plataformas de análise de dados, destacam-se diversos softwares de diversas áreas, porém, o software gratuito com fácil acesso e manipulação, R project, ainda não possuía tal método de programação linear. Nestes termos, o presente trabalho traz uma implementação do Método Simplex na plataforma R project e mostra a eficiência do mesmo para a achar, algebricamente, a solução ótima de um modelo de programação linear.

2. MÉTODO SIMPLEX

2.1. DEFINIÇÃO

È o processo utilizado para achar, algebricamente, a solução ótima para um modelo de programação linear desde que essa solução ótima exista.

2.2. FORMA PADRÃO DA PROGRAMAÇÃO LINEAR

Dado um problema de programação linear com m restrições e n variáveis, a forma padrão do problema é dado por

$$\text{Max. (ou Min.) } Z = c_1x_1 + c_2x_2 + \ldots c_nx_n$$

$$\text{Sujeito a:}\quad \begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \text{K} & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \text{K} & + & a_{2n}x_n & = & b_2 \\ \text{M} & & \text{M} & & & & \text{M} & = & \text{M} \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \text{K} & + & a_{mn}x_n & = & b_m \end{array} \qquad (1.1)$$
$$x_1 \geq 0,\ x_2 \geq 0, \text{K}\ ,\ x_n \geq 0$$

Assim, para j = 1, 2, ... ,n e i = 1, 2, ... , m temos que:

Z é a função objetivo e seu valor deve ser otimizado (maximizado ou minimizado);
x_j são as variáveis do problema;
c_j são os lucros ou os custos unitários;
b_i são os as quantidades disponíveis dos recursos ($b_i \geq 0$);
a_{ij} é a quantidade do recurso i utilizado na produção da variável j.

A forma padrão pode ser representada na notação matricial dada por:

$$\text{Max. (ou Min.) } Z = cx$$

$$\text{Sujeito a: } \begin{array}{l} Ax = b \\ x > 0 \end{array}$$

em que:

Z = função objetivo;
A = matriz (m x n) dos coeficientes tecnológicos;
x = vetor coluna (n x 1) das variáveis de decisão;
b = vetor coluna (m x 1) do lado direito das restrições, $b \geq 0$, ou vetor dos recursos disponíveis;
c = vetor linha (1 x n) dos lucros (ou custos).

A utilização do método simplex requer que o problema esteja na forma padrão. Esta exigência não constitui uma limitação, pois todo sistema linear pode ser colocado na forma padrão, mediante operações simples.

2.3. PROCEDIMENTO DO MÉTODO SIMPLEX

Para ser iniciado o método simplex necessita-se conhecer uma solução factível do sistema (chamada solução inicial) da equação (1.1). Após isso, ele verifica se a presente solução é ótima, ou seja, esta solução produz o valor máximo para função objetivo. Caso ele esteja na solução ótima, está resolvido o problema, caso contrário, ele irá procurar outros pontos viáveis que forneça valores maiores para a função objetivo. O algoritmo termina quanto ele encontra um ponto viável do sistema e todos os demais pontos viáveis possíveis forneçam valores menores para a função objetivo do que este.

Sendo assim, o método simplex compreenderá os seguintes passos:

I) Achar uma solução viável inicial

II) Verificar se esta solução é ótima. Se for, o problema está resolvido. Caso contrário, vai para o passo III.

III) Determinar um outro ponto viável do sistema e volta para o passo II.

3. APLICAÇÃO DO MÉTODO SIMPLEX

Foram utilizados diversos exemplos da aplicação do algoritmo do método simplex desenvolvido na plataforma R project. Cada exemplo utilizado esta listado abaixo. Todos os exemplos foram colocados na forma padrão e deles foram extraídos as informações necessárias para a utilização do algoritmo criado. O código fonte do algoritmo está presente no anexo assim como um arquivo de ajuda explicando a utilização e os argumentos solicitados para utilização do algoritmo.

1ª aplicação (Silva, E.M.; Silva, E.M - cap. 4 pag46):
Maximizar a função $Z = 3x_1 + 5x_2$

$$\text{Sujeito a: } \begin{cases} 2x_1 + 4x_2 \le 10 \\ 6x_1 + x_2 \le 20 \\ x_1 - x_2 \le 30 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

Resultados: $x_1 = 3{,}18$; $x_2 = 0{,}91$; $Z = 14{,}09$

2ª aplicação (Silva, E.M.; Silva, E.M - cap. 4 pag56, exercício 1.1):
Maximizar a função $Z = 10x_1 + 12x_2$

$$\text{Sujeito a: } \begin{cases} x_1 + x_2 \le 100 \\ 2x_1 + 3x_2 \le 270 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

Resultados: $x_1 = 30$; $x_2 = 70$; $Z = 1140$

3ª aplicação (Silva, E.M.; Silva, E.M - cap. 4 pag56, exercício 1.2):
Maximizar a função $Z = 2x_1 + 3x_2 + 4x_3$

$$\text{Sujeito a: } \begin{cases} x_1 + x_2 + x_3 \le 100 \\ 2x_1 + x_2 \le 210 \\ x_1 \le 80 \\ x_1 \ge 0, x_2 \ge 0, x_3 \ge 0 \end{cases}$$

Resultados: $x_1 = 0$; $x_2 = 0$; $x_3 = 100$; $Z = 400$

4ª aplicação (Silva, E.M.; Silva, E.M - cap. 4 pag56, exercício 1.3):
Maximizar a função $Z = 0{,}2x_1 + 2x_2 + 4x_3$

$$\text{Sujeito a: } \begin{cases} x_1 + 2x_2 \le 20 \\ 3x_1 + x_3 \le 50 \\ x_1 + x_2 - x_3 \le 15 \\ x_1 \ge 0, x_2 \ge 0, x_3 \ge 0 \end{cases}$$

Resultados: $x_1 = 0$; $x_2 = 10$; $x_3 = 50$; $Z = 220$

5ª aplicação (Silva, E.M.; Silva, E.M - cap. 4 pag56, exercício 1.4):

Maximizar a função $Z = 5x_1 - 3x_2 + 4x_3 - x_4$

$$\text{Sujeito a: } \begin{cases} x_1 + x_2 + x_3 + x_4 \le 600 \\ 2x_1 + x_3 \le 280 \\ x_2 + 3x_4 \le 150 \\ x_1 \ge 0, x_2 \ge 0, x_3 \ge 0, x_4 \ge 0 \end{cases}$$

Resultados: $x_1 = 0$; $x_2 = 0$; $x_3 = 280$; $x_4 = 0$; $Z = 1120$

6ª aplicação (Silva, E.M.; Silva, E.M - cap. 4 pag56, exercício 1.5):
Maximizar a função $Z = 2x_1 + 4x_3$

$$\text{Sujeito a: } \begin{cases} x_1 + 2x_2 + x_3 \le 8000 \\ 2x_1 \le 6000 \\ x_2 + x_3 \le 620 \\ x_1 \ge 0, x_2 \ge 0, x_3 \ge 0 \end{cases}$$

Resultados: $x_1 = 3000$; $x_2 = 0$; $x_3 = 620$; $Z = 8480$

4. REFERÊNCIAS BIBLIOGRÁFICAS

[1] PUCCINI, A.L.; PIZZOLATO, N.D.. "*Programação Linear*". Livros técnicos e científicos editora S.A.

[2] SILVA, E.M; SILVA, E.M.; GONÇALVES, V.; MUROLO, A.C.. "*Pesquisa Operacional*". 2ª edição. Editora Atlas S.A – 1996.

5. ANEXOS

5.1. ALGORITMO DO SIMPLEX.

```
#########################################################################
#   UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO                            #
#   ALUNO: Alessandro Henrique da Silva Santos                          #
#                                                                       #
#   Simplex: Versão simplificada do algorítmo simplex e do dual         #
#            simplex usando abordagem por quadro.                       #
#                                                                       #
#   No seu modo padrão, esse comando resolve o seguinte problema:       #
#                                                                       #
#   MAX z = c' * x                                                      #
#                                                                       #
#   Sujeito a A * x <=b.                                                #
#                                                                       #
#   Como utilizar:                                                      #
#   Simplex1(c,A,b,p)                                                   #
#                                                                       #
#   Entrada:                                                            #
#   c: vetor com os coeficientes da função objetivo                     #
#   A: matriz com os coeficientes de restrição                          #
#   b: vetor com as capacidades                                         #
#   p: Parâmetro responsável pela identificação de qual algorítmo       #
#      deve ser usado: 0-Simplex ou 1-Dual Simplex                      #
#                                                                       #
#   Saída:                                                              #
#   B: tabela final (matriz)                                            #
#   k: Número de iterações                                              #
#   Algorítmo que foi utilizado                                         #
#                                                                       #
#   Observações:                                                        #
```

```
#                                                                      #
#   (a) nenhuma tentativa foi feita para tratar de situações           #
#       especiais como o caso da degeneração.                          #
#                                                                      #
#   (b) essa rotina foi desenvolvida apenas para propósito             #
#       didáticos.                                                     #
#                                                                      #
#   Referência:                                                        #
#       Código Original em MATLAB escrito por:                         #
#                                                                      #
#       Dr Renato J Cintra                                             #
#       Universidade Federal de Pernambuco                             #
#       20 de Março de 2005                                            #
#       rjdsc@de.ufpe.br                                               #
#                                                                      #
#       Portado para o R por:                                          #
#                                                                      #
#       Alessandro Henrique S. Santos                                  #
#       Universidade Federal Rural de Pernambuco                       #
#       30 de janeiro de 2007                                          #
#       alessandrohss@yahoo.com.br                                     #
#                                                                      #
########################################################################


Simplex1<- function(c, A, b, p){
if(p==0){
####Inicialização####################
####n é o número de variáveis de decisão####
####m é o número de restrinções#########
m <- dim(A)[1]
n <- dim(A)[2]

####I é uma matriz identidade mxm#######
A_I <- cbind(A,diag(m))

####tabela inicial##################
B <- cbind(A_I,b)
ul<- c(-c, rep(0,m),0)
B<-  rbind(B,ul)

####Iniciar a contagem de iterações######
k<- 0

####Início do loop(laço) - realiza o loop enquanto satisfeito#####
while(1){
if(k>factorial(m+n)/(factorial(m)*factorial(n))){
cat("Foi Utilizado o algorítmo do SIMPLEX","\n")
cat("O algorítimo apresentou ciclagem","\n")
break}
print(B)

####Obtenção da linha objetivo##########
obj_row <- B[dim(B)[1], 1:dim(B)[2]-1]

####Obtenção da linha de capacidade######
b <- B[1:dim(B)[1]-1, dim(B)[2]]

################################
#TEST 1 : Verificando se a solução é encontrada
if(!(sum(obj_row<0)>0)){

        cat("Foi Utilizado o algorítmo do SIMPLEX","\n")
        cat("Simplex: Solução otima encontrada em",k,"iteraçãos","\n")
        break
}

####Encontrando a coluna pivô#########
pivot_col_index <- which.min(obj_row)

####TESTE DE SANIDADE: um mínimo######
pivot_col_index <- pivot_col_index[1]

################################
#TEST 2 : Checando se a solução existe
aux_pivot_col = B[1:dim(B)[1]-1,pivot_col_index]
if(sum(aux_pivot_col<=0) == m){
```

```
        cat("Foi Utilizado o algorítmo do SIMPLEX","\n")
        cat("Simplex: Nenhuma solução finita foi encontrada em",k,"iterações", "\n")
        break}

####Tratando números nulos#####
aux_pivot_col[which(aux_pivot_col==0)] <- 10^-100
ratio<-c()

####Calcular as razões#########
for(i in 1:m){
if(aux_pivot_col[i]<0){
        next}
ratio[i] <- b[i]/aux_pivot_col[i]}

####Assumindo valor infinito para valores negativos
#### calculo da função minimo seguida
ratio[which(ratio<0)] <- Inf

####Encontrar linha pivo##########
pivot_row_index <- which.min(ratio)

####Teste de sanidade: pegando um valor mínimo####
pivot_row_index <- pivot_row_index[1]

####Dividindo a coluna pivô pelo pivô####
pivot_row <- B[pivot_row_index,1:dim(B)[2]] / B[pivot_row_index,pivot_col_index]

####procedendo com a eliminação##########
for(j in 1:dim(B)[1]){
        if( j == pivot_row_index){
                next}
        B[j,1:dim(B)[2]] <- (pivot_row * (-B[j,pivot_col_index])) + B[j,1:dim(B)[2]]

}

#### Assign pivot row into B###############
B[pivot_row_index,1:dim(B)[2]] <- pivot_row

#### Icrementando a Interação#############
k <- k+1
}
}

if(p==1){
####Inicialização####################
####n é o número de variáveis de decisão####
####m é o número de restrinções#########
m <- dim(A)[1]
n <- dim(A)[2]

####I é uma matriz identidade mxm#######
A_I <- cbind(A,diag(m))

####tabela inicial#################
B <- cbind(A_I,b)
ul<- c(-c, rep(0,m),0)
B<-  rbind(B,ul)

####Iniciar a contagem de iterações######
k<- 0

####Início do loop(laço) - realiza o loop enquanto satisfeito#####
while(1){
if(k>factorial(m+n)/(factorial(m)*factorial(n))){
cat("Foi Utilizado o algorítmo do DUAL-SIMPLEX","\n")
cat("O algorítimo apresentou ciclagem","\n")
break}
print(B)

####Obtenção da linha objetivo##########
obj_row <- B[dim(B)[1], 1:dim(B)[2]-1]

####Obtenção da linha de capacidade######
b <- B[1:dim(B)[1]-1, dim(B)[2]]

################################
#TEST 1 : Verificando se a solução é encontrada
```

```
if(!(sum(b<0)>0)){

        cat("Foi Utilizado o algorítmo do DUAL-SIMPLEX","\n")
        cat("Simplex: Solução otima encontrada em",k,"iteraçãos","\n")
        break
}

####Encontrando a coluna pivô#########
pivot_col_index <- which.min(b)

####TESTE DE SANIDADE: um mínimo######
pivot_col_index <- pivot_col_index[1]

################################
#TEST 2 : Checando se a solução existe
aux_pivot_col = B[pivot_col_index,1:dim(B)[2]-1]

if(sum(aux_pivot_col>=0) == n+m){
        cat("Foi Utilizado o algorítmo do DUAL-SIMPLEX","\n")
        cat("Simplex: Nenhuma solução finita foi encontrada em",k,"iterações", "\n")
        break}

####Tratando números nulos#####
aux_pivot_col[which(aux_pivot_col==0)] <- 10^-100
ratio<-c()
ratio<-obj_row/aux_pivot_col

####Assumindo valor infinito para valores negativos
#### calculo da função minimo seguida
ratio[which(ratio>=0)] <- -Inf

####Encontrar linha pivo##########
pivot_row_index <- which.max(ratio)

####Teste de sanidade: pegando um valor mínimo####
pivot_row_index <- pivot_row_index[1]

####Dividindo a coluna pivô pelo pivô####
pivot_row <- B[pivot_col_index,1:dim(B)[2]] / B[pivot_col_index,pivot_row_index]

####procedendo com a eliminação##########
for(j in 1:dim(B)[1]){
        if( j == pivot_col_index){
                next}
        B[j,1:dim(B)[2]] <- (pivot_row * (-B[j,pivot_row_index])) + B[j,1:dim(B)[2]]
}

#### Assign pivot row into B###############
B[pivot_col_index,1:dim(B)[2]] <- pivot_row

#### Icrementando a Interação#############
k <- k+1
}
}
}
```