**Programación Lineal**

Catedrático: Jesús María Alvarado Andrade

Catedrático auxiliar: José Fernando Orellana Wer

Anesveth A. Maatens (20190339)

David Corzo (20190432)

**PSEUDOCÓDIGO - ALGORITMO SIMPLEX**

START

//RECIBIR INPUT del usuario

Input Función Objetivo <- En forma de diccionario. Keys son las variables y values los coeficientes;

Cantidad\_Variables <- Cantidad de Keys de Función Objetivo

Input constraints <- En forma de diccionario de diccionarios. Un diccionario por restricción. En las restricciones Las primeras Keys son las variables y values los coeficientes. Key 'symbol' tiene de value el signo de inecuación (como <= o >=). Key 'c' tiene de value la constante de la inecuación (<='125')

Constraints <- [

{x1:1,x2:3,x3:4,c:<=,18},

{x1:1,x2:3,x3:4,c:<=,19},

]

Maximizar <- True or False

//CREAR MATRIZ IDENTIDAD Y AÑADIRLA A LAS INECUACIONES

Def agregar\_slack\_vars(constraints) {

FOR i in constraints {

//Se añade cada fila de matriz identidad en el orden en que se reciben las restricciones. Por cada fila que se añada, se corre el 1 a la siguiente variable slack

constraints[i].update([s1:1,s2:0,s3:0,... ,sn]);

}

}

//Función para arreglar los signos de la matriz identidad dentro de las ecuaciones

Def PonerTodoEnStandardForm(constraints){

# se encarga de poner todo en standard form.

FOR i in constraints:

IF (constraint[i][‘symbol’] == ‘>=’ ):

FOR key que contenga ‘s’ en el nombre:

-1\*constraints[i][s];

}

//TABLA DE SOLUCIÓN.

Simplex\_table = list()

Def assemble\_initial\_simplex\_table(simplex\_table) {

//DEFINIR LA SOLUCIÓN BÁSICA INICIAL

simplex\_table.initialsol= por cada elemento de constraints, el número al que igualaba la restricción es asignado a la s{i} que contenga el número 1.

simplex\_table.cj = objective\_function.coheficient();

simplex\_table.basic\_vars = objective\_function.all\_vars();

Simplex\_table = slack\_variables();

Simplex\_table.add(all\_coheficients\_of\_restrictions);

}

//PROCEDIMIENTO DE TABLA

//Realizar las iteraciones necesarias

Def do\_iteration(simplex\_table) {

While True {

get\_key\_element(simplex\_table);

get\_key\_row(simplex\_table);

get\_key\_column(simplex\_table);

Variable\_entrada <- get\_variable\_entrada(row cj-zj;)//elemento más positivo si es maximizar, elemento más negativo si es minimizar, último elemento si todos son iguales.

a <- row de variable\_entrada;

b <- row de solucion\_inicial;

divide\_key\_row\_by\_key\_element(b,a);// Returns variable salida

multiply\_all\_rows\_with \_variable\_salida(-(variable\_salida));

update\_table();

get\_cj-zj() <- returns cj-zj

If (Maximizar):

BREAK WHEN cj-zj <= 0

ELSE:

BREAK WHEN cj-zj >= 0

}

}

La respuesta estará en terminos de todas las variables de la función objetivo. A cada variable se le asigna el valor correspondiente de la columna de solución inicial.

Utilidad <- zj

Answer = ‘Utilidad = Si1x1 + Si2x2 +... Sixi’

END