Programas de Programación Lineal

Francisco Cruz Eder, Silva Varela Gerardo Alejandro

Diciembre 2021

1 Método Simplex

1.1 Método Simplex en C

Este programa se auxilia de una interfaz gráfica y un archivo .txt de donde provienen los datos
Primero se lee el tipo de programa (Maximizar o Minimizar) y el número de variables y restricciones.

```
archivo = fopen("Entrada.txt","r");
salida = fopen("Resultado.txt","w");

fscanf(archivo,"%d",&opc); // 0 SI ES MAXIMIZAR 1 SI ES MINIMIZAR
fscanf(archivo,"%d",&m);
fscanf(archivo,"%d",&n);
```

De forma similar se leen los coeficientes de la función objetivo

```
for(j=0;j<n;j++){ //lee coeficientes de costos
    fscanf(archivo,"%lg",&costosAux[j].a);
    costosAux[j].b = 0;
}
```

Es importante mencionar que se utilizó una estructura para declarar un tipo de dato que almacena dos número reales, esto para representar la gran M en caso de que sea necesaria

```
typedef struct _M_{
    double a;
    double b;
}M;
```

También se leyeron los coeficientes en las restricciones, el tipo de restricción y el valor de la cota

```
for(i=0;i<m;i++){
    for(j=0;j<n;j++)
        fscanf(archivo,"%lg",&matrizAux[i][j]);
    fscanf(archivo,"%lg",&restricciones[i]);
    fscanf(archivo,"%lg",&matrizAux[i][n]);
}
```

Si el problema es de minimización se multiplica la función objetivo por -1 para obtener un problema de maximización

```
if(opc== 0){ // SI SE QUIERE MAXIMIZAR SE LLE
    for(j=0;j<n;j++){ //POR LO QUE LOS COEFIC
        costosAux[j].a = costosAux[j].a*-1;
    }
}
```

Se cuentan las variables de igualdad, de menor o igual y de mayor o igual, esto para saber que tipo de variables añadir al problema, si de holgura o artificiales

```
for(i=0;i<m;i++){
    if(restricciones[i] == 3)   //Se
        contador++;
    else if(restricciones[i]==2) //S
        contador1++;
    else if( restricciones[i] == 1)
        contador3++;
}
```

Se expande la matriz original por columnas llenando estas de ceros y después se identifican las entradas donde corresponden los coeficientes de las variables de holgura y artificiales

```
for(i=0;i<m;i++){
    if(restricciones[i] == 3){ //Se añad
        matriz[i][n + contador2 ] = 1;
        indices[i] = n+contador2;
        contador2++; }
    else if(restricciones[i]==2){ //Se a
        matriz[i][n + contador2 ] = -1;
        matriz[i][n + contador2 +1] = 1;
        if(opc== 0)
            costos[n + contador2 +1].b =
        else
            costos[n + contador2 +1].b =
        //Eliminar la variable artificia
        indices[i] = n+contador2+1;
        contador2+=2;
    }else if( restricciones[i] == 1){
        matriz[i][n+contador2 ] = -1;
        if(opc==0)
            costos[n+contador2].b = 1;
        else
            costos[n+contador2].b =- 1;
        //Eliminar la variable artificia
        indices[i] = n+contador2;
        contador2++;
    }
}
```

Por último se despeja la función objetivo y se eliminan las variables artificiales que se hayan agregado.

```
for(i=0;i<m;i++){
    if(restricciones[i] == 3)
        contador2++;
    else if(restricciones[i]==2){ //Se añaden variables de holg
        for(j=0;j<n+1 +contador + 2*contador1 +contador3;j++){
            if(opc== 0)
                costos[j].b =costos[j].b - matriz[i][j];
            else
                costos[j].b = costos[j].b + matriz[i][j];
        }
        contador2+=2;
    }else{//Eliminar la variable artificial de la función origi
        for(j=0;j<n+1 +contador + 2*contador1 +contador3;j++){
            if(opc== 0)
                costos[j].b =costos[j].b - matriz[i][j];
            else
                costos[j].b = costos[j].b + matriz[i][j];
        }
        contador2++;
    }
}
```

Así, tenemos el problema en forma estándar listo para que se aplique el método simplex.

Primero se verifican que los coeficientes de la función objetivo sean no negativos, en caso contrario se selecciona la entrada más negativa, está corresponderá a la variable que entra, en caso de que no haya ninguna se habrá llegado a una solución óptima

```
for(j=0;j<n;j++){
    if(compara(costos[j],cero)){  //Si se viola el crite
        if(masNega!=-1){
            //if(matriz[1][j] < matriz[1][masNega])
            if( compara(costos[j] , costos[masNega])  )
                masNega=j;
        }
        else masNega=j; //Si es la primer negativa que e
        contador++;//Cuenta los coeficientes no negativc
    }
}
if(contador==0){ //Usar un contador para esto
        printf("La tabla óptima es: \n");
        break;
}
```

En caso de que la solución no sea óptima se verificará el criterio de factibilidad, se para encontrar la variable de salida, está será la correspondiente al renglón con la relación menos positiva, en caso de que todas las relaciones sean negativas la solución no estará acotada por lo tanto el programa finalizará

```
for(i=0;i<m;i++){
    if(relaciones[i] >= 0){  //Si es menor que cero se ignora
        if(menosPos !=-1){
            printf("\n Comparando %lg con %lg",relaciones[i], relaciones[menosPos]);
            if(relaciones[i] < relaciones[menosPos])
                menosPos=i;
        }
        else menosPos=i; //Si es la primer negativa que encuentra
        contador++; //Aumenta el contador de costos no negativas
    }
}
indices[menosPos] = masNega;
if( contador == 0 ){ //Usar un contador para esto
    fprintf(salida,"%d",0);
    fprintf(salida,"La solución es no acotada");
    printf("No hay solución factible");
    break;
}
```

En caso contrario, se tendrán identificadas la variable de entrada y la de salida, por lo tanto se pivoteará la entrada correspondiente a esta

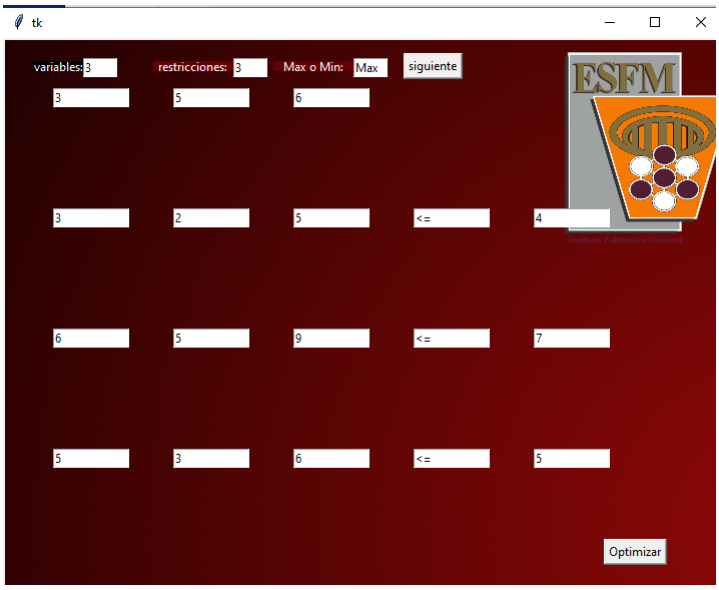
```
aux2 = costos[masNega].a;
aux3 = costos[masNega].b;
printf("La entrada a eliminar es: %lg + %lgM",costos[masNega].a,costos[masNega].b);
for(j=0;j<n+1;j++){
    costos[j].a = costos[j].a - aux2*matriz[menosPos][j];
    costos[j].b = costos[j].b - aux3*matriz[menosPos][j];
    imprime(costos[j]);
}
printf("\n");
for(i=0;i<m;i++){
    aux2 = matriz[i][masNega];
    if(i != menosPos){
        for(j=0;j<n+1;j++){
            matriz[i][j] = matriz[i][j] - matriz[menosPos][j]*aux2;
        }
    }
    printf("\n");
}
```

Se volverá a evaluar el criterio de óptimalidad y en caso de que aún haya variables negativas se realizará otra iteración

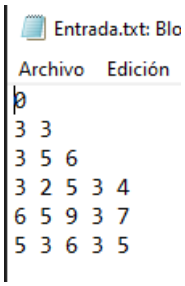
1.2 Interfaz Gráfica

El programa funciona mediante una interfaz gráfica elaborada en python con la biblioteca Tkinter, inicialmente se tienen que introducir el numero de variables y restricciones así como el tipo de problema (Maximización o Minimización), una vez introducidos el botón "siguiente" generará las casillas necesarias

Una vez generadas las casillas se ingresan los coeficientes de las variables en la función objetivo y en las restricciones, así como las capacidades y el tipo de restricción, por defecto estás son del tipo "<=" pero pueden ser cambiadas por "=" o ">="



Una vez introducidos los datos necesarios el botón de "Optimizar" ejecutará el programa elaborado en C para resolver el problema por el método simplex. Este generará un archivo .txt con todos los datos ingresados para su posterior lectura



2 Punto Medio

Se implementó un programa que simula el algoritmo de punto medio, para este es necesario ingresar el problema en forma estándar y por lo tanto primero se leen los datos necesarios como el número de variables y de restricciones, los coeficientes de la función objetivo, la matriz A y una solución inicial.
Una vez ingresados los datos, caculamos D, el cual está dado por

$$D = I_{nxn}x$$

Donde n es el número de variables, es decir, las entradas de la diagonal corresponden a los coeficientes de la solución inicial

```
for i in range(n):
  D[i][i] = float(X[i])
```

También se requieren calcular

$$\overline{A} = AD$$

$$\overline{c} = Dc$$

$$P = I - \overline{A}^t(\overline{A}\overline{A}^t)^{-1}\overline{A}$$

$$C_p = P\overline{c}$$

Calculando $\overline{A}$ y $\overline{c}$

```
Av = np.matmul(A,D)
Cv = np.matmul(D,C)
```

P se obtuvo de la siguiente fórmula

```
Avt = np.transpose(Av)
Wa = np.matmul(Av,Avt)
W = np.linalg.inv(Wa)
Va = np.matmul(Avt,W)
V = np.matmul(Va,Av)
I = np.arange(ma*na).reshape(ma, na)
for i in range(na):
  for j in range(ma):
    I[i][j] = 0
  I[i][i] = 1

 for i in range(na):
   for j in range(ma):
     V[i][j] = -V[i][j]
P = I+V
```

Se calculo Cp

```
Cp = np.matmul(P,Cv)
```

Se tomo el máximo de los valores absolutos de las componentes negativas de Cp

```
max = 0
for element in Cp:
  for el in element:
    if  el < 0 and max < abs(el):
       max = abs(el)
if max <= 0:
  print("No hay entradas negativas en Cp")
  break
```

Se obtuvo

$$\overline{x} = \vec{1} + \frac{\alpha}{M} C_p$$

De la siguiente manera

```
alpM = alpha/max
Cp = alpM*Cp
Xv = Uno + Cp
```

Por último se evalúo el criterio, tomando un $\varepsilon = 0.0005$ se calculo la norma de la diferencia de x y x_0 de modo que si

$$||x - x_0|| < \varepsilon$$

entonces hemos acabado y x $= x_0$, en caso contrario se realizará otra iteración

```
norm = 0
Xu = np.matmul(D,Xv)
Xuu = X - Xu
for element in Xuu:
  for el in element:
    norm += el*el
norm = math.sqrt(norm)
if norm < 0.0005 :
  na, ma = Xuu.shape
  print("La solución se encontró en la iteración "+str(cont)+" \nLa solución es:")
  for i in range(na):
    for j in range(ma):
      print("x"+str(i)+"=",Xu[i][j])
  break
```