Informe de Laboratorio Método Simplex

José M. Benavides, Emmanuel Martinez

Ingeniería de Sistemas, Universidad de Los Llanos

Villavicencio, Colombia

jmbenavides@unillanos.edu.co

emartinez.sierra@unillanos.edu.co

Resumen – Este trabajo presenta una implementación del método Simplex utilizando Python con el apoyo de NumPy para cálculos matriciales y Tkinter para el desarrollo de una interfaz gráfica de usuario (GUI). La aplicación permite a los usuarios ingresar la función objetivo y restricciones, resolviendo el problema de forma iterativa mientras muestra cada paso del procedimiento.

I. INTRODUCCIÓN

El método Simplex es una de las técnicas más utilizadas en programación lineal para encontrar soluciones óptimas en problemas de optimización con restricciones lineales. Su aplicación es fundamental en áreas como la economía, la logística, la ingeniería y la investigación operativa.

El código desarrollado aplica la formulación estándar del método Simplex, identificando columnas y filas pivote, normalizando los coeficientes y realizando operaciones de fila hasta encontrar la solución óptima. A través de una interfaz interactiva, los usuarios pueden observar la evolución de la matriz de restricciones y los cambios en la función objetivo durante cada iteración.

II. REFERENTE TEÓRICO

La programación lineal es una técnica matemática utilizada para la optimización de recursos en problemas donde se busca maximizar o minimizar una función objetivo, sujeta a un conjunto de restricciones lineales. Se aplica en diversas áreas como economía, ingeniería, logística y administración de empresas. El método simplex se basa en la representación de restricciones como inecuaciones en un plano cartesiano. La intersección de estas restricciones forma una región factible, donde se encuentra la solución óptima. Para determinar la mejor solución, se evalúa la función objetivo en los vértices de la región factible.

Un problema de programación lineal se compone de:

- **Función objetivo**: Expresión matemática que se desea optimizar (maximizar o minimizar).

$$Z = ax + by$$

- **Restricciones**: Condiciones que limitan las posibles soluciones, expresadas como inecuaciones.

$$A_1x + B_1y \le C_1$$
$$A_2x + B_2y \le C_2$$

Condiciones de no negatividad:

$$x \ge 0 , y \ge 0$$

Estas aseguran que las soluciones sean válidas dentro del contexto del problema.

El método Simplex, desarrollado por George Dantzig en 1947, es un algoritmo iterativo que resuelve problemas de programación lineal transformando el sistema en una tabla matricial y aplicando operaciones de fila para mejorar progresivamente la solución hasta alcanzar un óptimo.

El procedimiento consta de los siguientes pasos:

III. PROCEDIMIENTO

- Formulación del problema: Se expresa el problema en su forma estándar, agregando variables de holgura si es necesario.
- Construcción de la tabla Simplex: Se organiza la información en una matriz inicial donde la última fila representa la función objetivo.
- Selección de la columna pivote: Se elige la variable de entrada al encontrar el coeficiente más negativo en la fila de la función objetivo.
- Determinación de la fila pivote: Se selecciona la variable de salida mediante la razón mínima entre los valores del lado derecho y los coeficientes de la columna pivote.
- Actualización de la tabla: Se normaliza la fila pivote y se aplican operaciones de fila para convertir el coeficiente de la columna pivote en cero en las demás filas.
- Verificación de optimalidad: Si no quedan coeficientes negativos en la fila de la función objetivo, la solución es óptima; de lo contrario, el proceso se repite desde el paso 3.

```python
# Encuentra la columna pivote en la última fila (Z)
def encontrar_columpiv(matriz, num_filas, num_colum):
    num_pivoteZ = 0
    colum_pivote = -1
    for j in range(num_colum - 1):
        if matriz[num_filas - 1][j] < num_pivoteZ:
            num_pivoteZ = matriz[num_filas - 1][j]
            colum_pivote = j
    return colum_pivote
```

Figura 1. Código para calcular la columna pivote.

```python
# Encuentra el elemento pivote dividiendo el lado derecho entre la columna pivote
def encontrar_elemento_pivote(matriz, colum_pivote, num_filas, num_colum):
    fila_pivot = -1
    num_menor = float('inf')
    for i in range(num_filas - 1):
        if matriz[i][colum_pivote] > 0:
            razon = matriz[i][num_colum - 1] / matriz[i][colum_pivote]
            if razon < num_menor:
                num_menor = razon
                fila_pivot = i
    return fila_pivot, matriz[fila_pivot][colum_pivote]
```

Figura 2. Código para calcular el elemento pivote.

```python
# Ejecuta el método simplex paso a paso mostrando cada iteración
def simplex():
    global matriz_1, matriz_2, num_filas, num_colum
    salidaaux = 1
    while salidaaux == 1:
        imprimir_matriz(matriz_1)
        colum_pivote = encontrar_columpiv(matriz_1, num_filas, num_colum)
        fila_pivot, elemento_pivote = encontrar_elemento_pivote(matriz_1, colum_pivote, num_filas, num_colum)

        for j in range(num_colum):
            matriz_2[fila_pivot][j] = matriz_1[fila_pivot][j] / elemento_pivote

        for i in range(num_filas):
            if i != fila_pivot:
                factor = matriz_1[i][colum_pivote]
                for j in range(num_colum):
                    matriz_2[i][j] = matriz_1[i][j] - factor * matriz_2[fila_pivot][j]

        matriz_1 = [row[:] for row in matriz_2]
        salidaaux = any(matriz_1[num_filas - 1][j] < 0 for j in range(num_colum - 1))

    resultado_final = f"Solución óptima encontrada: Z = {matriz_1[num_filas - 1][-1]:.2f}"
    steps_text.set(steps_text.get() + "\n" + resultado_final + "\n")
```

Figura 3. Código para aplicar método simplex, con ayuda de las funciones anteriores y otras implementaciones como normalizar la columna pivote y realizar operaciones matriciales.

```python
# Obtiene los datos de la interfaz y resuelve el método simplex
def resolver_simplex():
    global matriz_1, matriz_2, num_filas, num_colum
    try:
        numero_varZ = int(entry_vars.get())
        numero_inec = int(entry_ineqs.get())
        num_filas = numero_inec + 1
        num_colum = numero_inec + numero_varZ + 2

        matriz_1 = crear_matriz(num_filas, num_colum)
        matriz_2 = crear_matriz(num_filas, num_colum)

        coef_z = list(map(int, entry_z.get().split()))
        for j in range(numero_varZ):
            matriz_1[num_filas - 1][j + 1] = -coef_z[j]

        for i in range(numero_inec):
            coef_ineq = list(map(int, restricciones_entries[i].get().split()))
            for j in range(numero_varZ):
                matriz_1[i][j + 1] = coef_ineq[j]
            matriz_1[i][num_colum - 1] = coef_ineq[-1]
            matriz_1[i][numero_varZ + 1 + i] = 1

        simplex()
        imprimir_matriz(matriz_1)
    except Exception as e:
        steps_text.set(f"Error en los datos: {e}")
```

Figura 4. Código para obtener la información de la interfaz gráfica.

Todo ese proceso se desarrolló en un programa con el lenguaje de programación Python, el cual recibe la función objetivo y nos pide seleccionar nuestro objetivo, luego nos pide que ingresemos las restricciones que tiene nuestro problema como se observa en la figura 5, a continuación se generan los espacios donde ingresaremos las restricciones, una vez ingresadas las restricciones se da al botón calcular y nos arroja las vértices, la solución óptima y el grafico de la función con la región factible sombreada como se observa en la figura 6.

IV. RESULTADOS

Al realizar una ejecución del programa ingresando unos datos de pruebas como se observa en la figura 5, se obtuvieron los siguientes resultados observables en la figura 6.

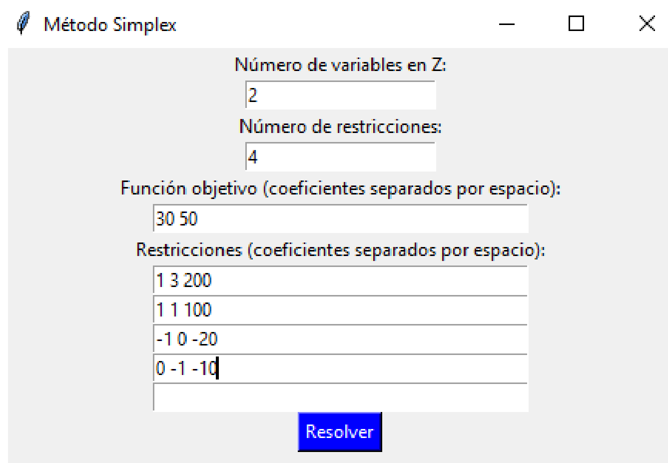

Figura 5. Ventana ingreso de datos y coeficientes.

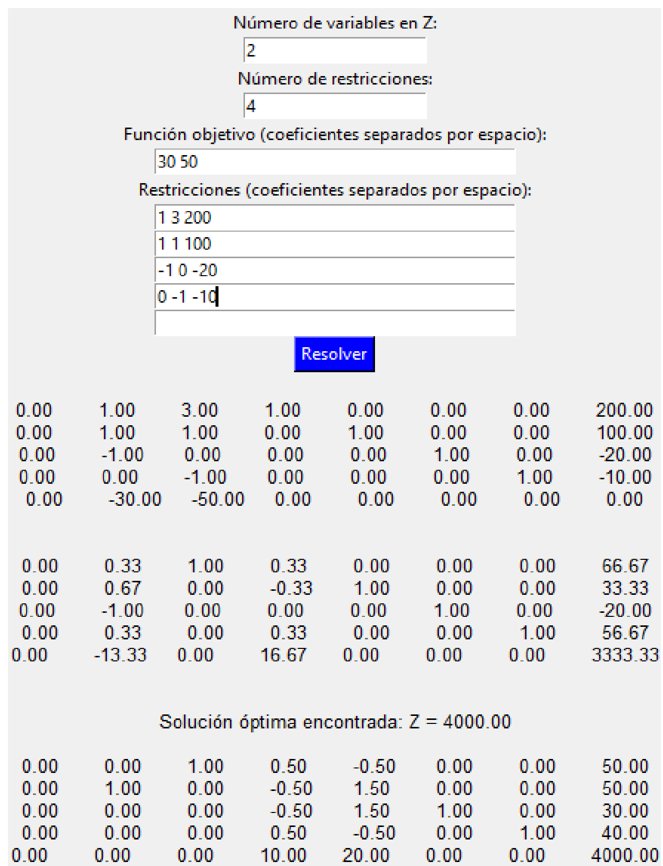

Figura 6. Resultados de método simplex.

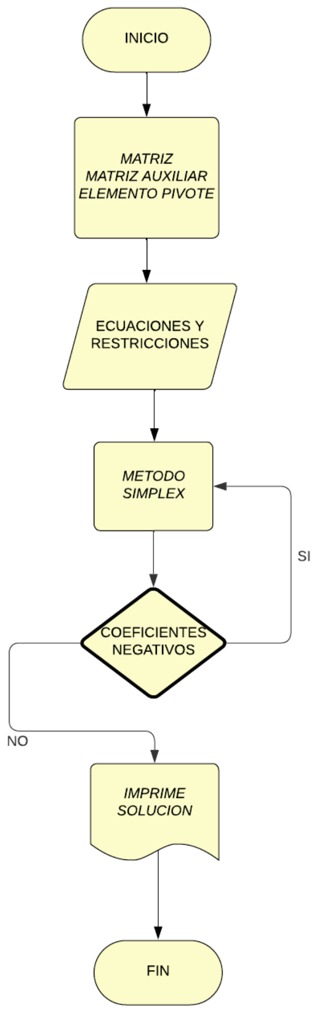

Figura 7. Diagrama de flujo del código simplificado.

V. CONCLUSIÓN

El método simplex permite solucionar de manera clara problemas de programación lineal con dos variables. En este caso, se encontró que la mejor solución es en el punto (50.0, 50.0), con un valor óptimo de 4000.0.
Se presentaron dificultades en la implementación de la lógica que conlleva el método simplex.

VI. REFERENCIAS

[1] Winston, W. L. (2004). Operations Research: Applications and Algorithms. Cengage Learning.

[2] Hillier, F. S., & Lieberman, G. J. (2010). Introduction to Operations Research. McGraw-Hill Education.

[3] Dantzig, G. B. (1951). *Maximization of a Linear Function of Variables Subject to Linear Inequalities*. In T. C. Koopmans (Ed.), *Activity Analysis of Production and Allocation* (pp. 339–347). Wiley.

[4] Bazaraa, M. S., Jarvis, J. J., & Sherali, H. D. (2010). *Linear Programming and Network Flows* (4th ed.). Wiley.