UNIVERSIDAD DE LOS LLANOS
Facultad de Ciencias básicas e ingenierías
Departamento de Ingeniería de sistemas

INFORME DE LABORATORIO OPTIMIZACIÓN

Métodos de Programación Lineal

Juan S. Romero Aguirre[1], Yarlinson T. Barranco[2]

2. Cod: 160004540, Ingeniería de sistemas,
3. Cod: 160004501, Ingeniería de sistemas.

Facultad de Ciencias Básicas e Ingenierías.
Ingeniería de Sistemas

I. Resumen

El objetivo de este proyecto es desarrollar una plataforma educativa en Django que permita a los usuarios aprender y simular diferentes métodos de programación lineal. La metodología a utilizar consiste en la identificación de los métodos de programación lineal incluyendo. Este software está diseñado para ser una herramienta educativa interactiva, que facilita la comprensión y la aplicación de métodos de optimización lineal a través de simulaciones, visualizaciones y explicaciones teóricas.

II. Introducción

La programación lineal es una técnica matemática utilizada para optimizar una función objetivo lineal, sujeta a un conjunto de restricciones lineales. Esta disciplina es fundamental en diversas áreas como la investigación de operaciones, la economía, la ingeniería y la gestión de recursos.[1]

Un problema de programación lineal consta de tres elementos principales:

- Función Objetivo: Es una ecuación lineal que debe ser maximizada o minimizada. Por ejemplo, se puede buscar maximizar los ingresos o minimizar los costos.
- Restricciones: Son ecuaciones o desigualdades lineales que limitan las variables de decisión. Estas restricciones representan las limitaciones o recursos disponibles, como el tiempo, el dinero o la capacidad de producción.
- Variables de Decisión: Son las incógnitas que representan las cantidades que debemos determinar para optimizar la función objetivo. Estas variables deben ajustarse de manera que se cumplan todas las restricciones y se maximice o minimice la función objetivo.

Modelo común de investigación de operaciones (forma canónica):

$$Opt\left\{z = \sum_{i=1}^{n} C_i X_i\right\}$$

s.a

$$\sum_{j=1}^{n} a_i; x_i \leq \geq b$$

Donde:
X_i = Variables de decisión
C_i = Coeficientes de la función objetivo
a_i = Parámetros de las restricciones
b = Coeficientes de restricción
Opt = Max o Min

Solución gráfica: El procedimiento de solución gráfica comprende dos pasos:

- Determinación del espacio de soluciones que define todas las soluciones factibles del modelo.
- Determinación de la solución óptima, entre todos los puntos factibles del espacio de soluciones.

III. Marco teórico

En la actualidad, existen diversas herramientas y software dedicados a la optimización lineal. Algunas de las plataformas más conocidas incluyen MATLAB, Gurobi, CPLEX y LINDO. Estas herramientas, aunque poderosas, a menudo están orientadas a usuarios avanzados y pueden no ser accesibles para principiantes.Además, muchas carecen de elementos

didácticos que faciliten el aprendizaje y la comprensión de los métodos subyacentes. El proyecto propuesto busca llenar este vacío proporcionando una plataforma educativa interactiva que no solo permita la resolución de problemas de programación lineal, sino que también ofrezca explicaciones teóricas y visualizaciones que hagan más accesibles estos conceptos.[2]

El desarrollo de esta plataforma debe considerar varias restricciones experimentales, entre ellas: Rendimiento del Sistema: La plataforma debe ser capaz de manejar múltiples simulaciones de manera eficiente sin comprometer el rendimiento. Exactitud de los resultados: Los algoritmos implementados deben proporcionar resultados precisos y confiables.

Breve reseña de la programación lineal[3]:

- Siglo XVIII - Métodos Iniciales: Se desarrollan conceptos relacionados con la optimización, como el método de los mínimos cuadrados de Gauss.
- Década de 1940 - Algoritmo del Simplex: George Dantzig crea el algoritmo del simplex, una técnica eficiente para resolver problemas de programación lineal.
- Década de 1950 - Aplicaciones en Economía: Kantorovich y Koopmans aplican la programación lineal en economía, lo que impulsa su uso.
- Década de 1960 - Software Comercial: Se desarrollan los primeros programas comerciales para la resolución de problemas de programación lineal.
- Década de 1970 - Ampliación de Aplicaciones: La programación lineal se aplica en logística, producción y otras áreas.
- Década de 1990 en Adelante - Código Abierto: Se crean bibliotecas de código abierto para resolver problemas de programación lineal, facilitando su uso generalizado.

IV. Sección experimental

Lo primero que se realizó fue un análisis detallado del software a realizar, una vez planteado se realizó mediante web con LucidChart los siguientes diagramas:

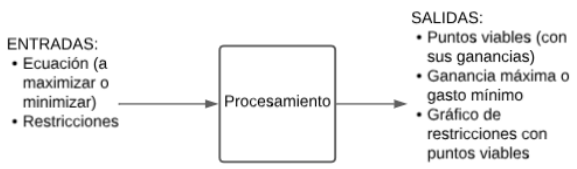

Figura 1. Diagrama de entrada-salida.

Entrada:
1. El usuario ingresa una función objetivo en una caja de entrada de texto etiquetada como "Ingrese la función objetivo".
2. El usuario ingresa las restricciones lineales, una por línea, en un área de entrada de texto etiquetada como "Ingrese las restricciones (una por línea)".
3. El usuario selecciona si desea maximizar o minimizar la función objetivo utilizando botones de opción etiquetados como "Maximizar" o "Minimizar" bajo la etiqueta "Seleccione la dirección de la optimización". 4. El usuario hace clic en el botón "Calcular" para iniciar el proceso de optimización.

Procesamiento:
1. Analiza las restricciones lineales, convirtiendo las desigualdades en igualdad y separando las variables y las expresiones correspondientes.
2. Agrega restricciones de no negatividad para las variables x e y.
3. Calcula los puntos de intersección entre las restricciones.
4. Determina si se debe maximizar o minimizar la función objetivo.
5. Calcula el valor óptimo (ganancia máxima o costo mínimo) utilizando las restricciones y la función objetivo.
6. Muestra los resultados en la GUI, incluyendo los puntos de intersección y el valor óptimo.
7. Genera un gráfico que visualiza las ecuaciones y los puntos de intersección en un plano XY.

Salida:
1. La GUI muestra los resultados sobre los puntos de intersección encontrados, los valores óptimos y la dirección de la optimización (maximizar o minimizar).

Después se realizó el diagrama de flujo el cual ayuda a definir los métodos, variables y procedimientos, lo cual ayuda a tener una idea más detallada el funcionamiento de programa:

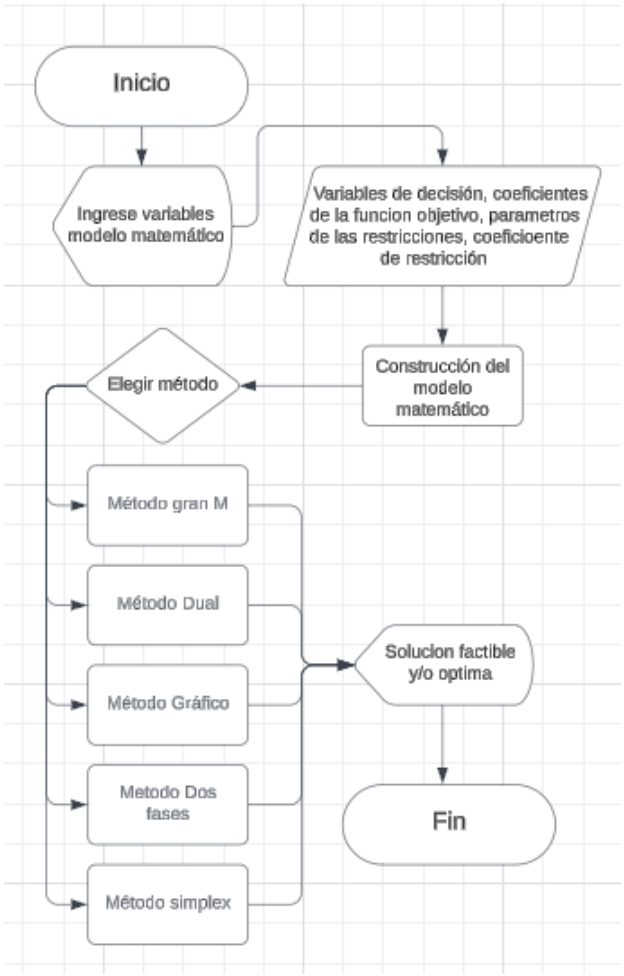

Figura 2. *Diagrama de flujo.*

El usuario deberá ingresar el modelo matemático en el software en el cual se muestra la solución gráfica y la solución óptima.

Ya identificadas las soluciones, mediante la herramienta Visual Studio Code, comenzó el desarrollo del software usando lenguaje de programación Python 3.11.5. A continuación se presenta las partes fundamentales del código fuente:

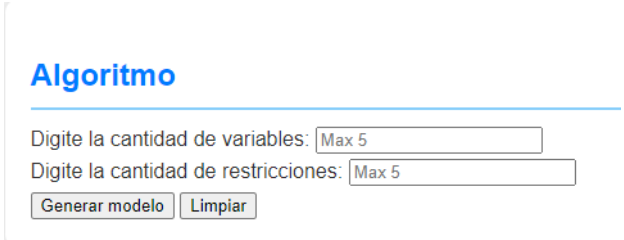

Figura 3. *Guardado de los datos.*

```python
def generar_restricciones():
    global restriccion_entries, restriccion_labels, calcular_btn
    global num_restricciones_entry

    num_restricciones = int(num_restricciones_entry.get())
    # print(num_restricciones)

    for widget in canvas.winfo_children():
        widget.destroy()

    restriccion_entries = []
    restriccion_labels = []
    if num_restricciones > 5:
        num_restricciones = 5

    for i in range(num_restricciones):
        restriccion_label = Label(root, text=f"Restriccion {i +1}: ")
        restriccion_label.pack()
        restriccion_entry = Entry(root)
        restriccion_entry.pack()
        restriccion_labels.append(restriccion_label)
        restriccion_entries.append(restriccion_entry)
```

Figura 4. *función restricciones.*

```python
def preparar_datos(self):
    self.funcion_objetivo = [-1 * float(val) for val in self.funcion_objetivo]
    self.restricciones = [list(map(float, restr)) for restr in self.restricciones]
    self.funcion_objetivo = list(map(float, self.funcion_objetivo))

    for i in range(len(self.restricciones)):
        if self.signos[i] == '>=':
            self.cantHolguras += 1
            self.restricciones[i] = [-1 * float(val) for val in self.restricciones[i]]
            for j in range(self.cantHolguras - 1):
                self.restricciones[i].insert(-1, 0)
            self.restricciones[i].insert(-1, -1)
            for j in range(self.totalHolguras - (self.cantHolguras - 1) - 1):
                self.restricciones[i].insert(-1, 0)
            self.signos[i] = '='
        elif self.signos[i] == '<=':
            self.cantHolguras += 1
            self.restricciones[i] = [1 * float(val) for val in self.restricciones[i]]
            for j in range(self.cantHolguras - 1):
                self.restricciones[i].insert(-1, 0)
            self.restricciones[i].insert(-1, 1)
            for j in range(self.totalHolguras - (self.cantHolguras - 1) - 1):
                self.restricciones[i].insert(-1, 0)
            self.signos[i] = '='
        self.funcion_objetivo.append(0)
    self.funcion_objetivo.append(0)
    self.crear_matriz_final()

def crear_matriz_final(self):
    matriz_final = [self.funcion_objetivo] + self.restricciones
    self.matriz_final_np = np.array(matriz_final, dtype=float)
    self.matrices_resultado.append(self.matriz_final_np.copy())  # Guardar la matriz inicial
```

Figura 5. *Realizar procedimiento*

```python
# Función para graficar la región acotada
def graficar_region_acotada(vertices):

    # organizar
    vertices_organizados = sorted(vertices, key=lambda x: (x[0], x[1]))
    x_vertices, y_vertices = zip(*vertices_organizados)
    plt.fill(x_vertices, y_vertices, alpha=0.2, color="blue", label="Región factible")
    # print(vertices)
    # print(vertices_organizados)


# Función para graficar las restricciones
def graficar_restricciones():

    for i in range(len(ecuaciones)):
        ec_label = restriccion_entries[i].get()
        A, B, C = ecuaciones[i]
        if B == 0:
            m = C / A
            plt.axvline(m, linestyle="-")

        elif A == 0:
            m = C / B
            plt.axhline(m, linestyle="-")

        else:
            x = np.linspace(0, 1000)
            y = (C - (A * x)) / B
            plt.plot(x, y, label=ec_label)
```

Figura 6. *Función encargada de graficar los datos.*

V. Resultados

En la siguiente imagen se muestra la interfaz resultante del software:

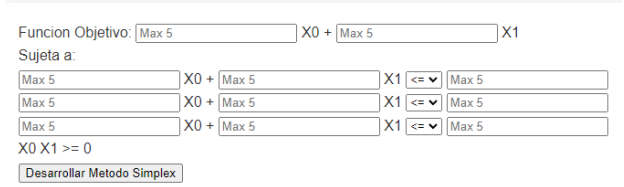

Figura 7. *GUI.*

La siguiente prueba de escritorio se realizó con el fin de comprobar el correcto funcionamiento del aplicativo donde se tienen los siguientes datos.

$$Z - 300X0 - 400X1 + 0S0 + 0S1 = 0$$

$$3\,X0 + 3\,X1 + S1 = 120$$

$$3\,X0 + 6\,X1 + S2 = 180$$

Resultados del método simplex

Tablas:

Iteración #0

X0	X1	S0	S1	K
3.0	3.0	1.0	0.0	120.0
3.0	6.0	0.0	1.0	180.0
-300.0	-400.0	0.0	0.0	0.0

Iteración #1

X0	X1	S0	S1	K
1.5	0.0	1.0	-0.5	30.0
0.5	1.0	0.0	0.16666666666666666	30.0
-100.0	0.0	0.0	66.66666666666666	12000.0

Iteración #2

X0	X1	S0	S1	K
1.0	0.0	0.6666666666666666	-0.3333333333333333	20.0
0.0	1.0	-0.3333333333333333	0.3333333333333333	20.0
0.0	0.0	66.66666666666666	33.33333333333333	14000.0

Resultados:

Tipo de resultados: {'X1': 20.0, 'X2': 20.0}

Z: 14000.0

Figura 7. *Prueba de escritorio*

VI. Conclusiones

El desarrollo de la plataforma educativa para la simulación de métodos de programación lineal en Django representa una contribución significativa al ámbito educativo. Al proporcionar una herramienta interactiva y accesible, se facilita el aprendizaje y la comprensión de conceptos complejos en programación lineal.

La arquitectura y diseño del software, utilizando Django como framework principal, proporcionan una base sólida para futuras expansiones y mejoras. La plataforma está diseñada para ser escalable, lo que permite la inclusión de nuevos métodos de optimización y características adicionales en el futuro.

VII. Referencias

- Programación lineal: Qué es, usos y pasos para realizarla. Investigación de mercado. Cristina Ortega. Fecha desconocida https://www.questionpro.com/blog/es/programacion-lineal/

- Santana Robles, F. (2019). Aplicaciones de la programación lineal. *Ingenio Y Conciencia Boletín Científico De La Escuela Superior Ciudad Sahagún*, *6*(12), 95-96. https://doi.org/10.29057/escs.v6i12.4113

- Historia de la programación lineal. Optimus. 7 de May de 2012. Autor desconocido. https://inveoperaciones.wordpress.com/2012/05/07/historia-de-la-programacion-lineal/