Modelo Matemático de Programación Lineal, Método Simplex (2024)

Yeimy V. López Terreros - 160004713, Michael A. Rodriguez Estrada – 160004535

**Resumen - Este informe detalla el desarrollo de un programa en Python que permite construir y resolver modelos matemáticos de programación lineal. El programa implementa el método Simplex para encontrar la solución óptima de problemas de optimización, mediante la resolución iterativa de tablas que representan las restricciones y la función objetivo. El software fue diseñado para admitir modelos con múltiples restricciones y variables, proporcionando una herramienta eficiente para la toma de decisiones en problemas de optimización.**

**Índice de Términos – Programación lineal, función objetivo, restricciones, método Simplex, optimización.**

1. introducción

La programación lineal es una herramienta clave en la investigación de operaciones y la optimización de recursos, ampliamente utilizada para maximizar o minimizar una función objetivo sujeta a restricciones lineales. El método Simplex es una técnica iterativa y algebraica utilizada para resolver estos problemas cuando hay más de dos variables.

El objetivo de este laboratorio es diseñar y desarrollar un programa en Python que permita a los usuarios introducir y resolver problemas de programación lineal utilizando el método Simplex, mostrando las iteraciones en tablas y destacando los pivotes, para finalmente encontrar la solución óptima de la función objetivo.

1. referente teórico.
2. *Programación Lineal.*

La programación lineal es una técnica matemática para optimizar una función objetivo sujeta a restricciones lineales. El modelo de programación lineal consta de tres elementos fundamentales:

1. **Función Objetivo**: Una función lineal que representa el objetivo a optimizar (maximizar o minimizar).

Z = c1X1 + c2X2 + . . . + cnXn

Donde X1, X2, …, Xn son variables de decisión y C1, C2, …, Cn son coeficientes de la función objetivo.

1. **Restricciones**: Un conjunto de desigualdades o ecuaciones lineales que definen las limitaciones. Cada restricción tiene la forma:

a1X1 + a2X2 + … +anXn <= b

Donde a1, a2, …, an, son coeficientes y b es una constante

**Variables de Decisión**: Las incógnitas X1,X2,…,Xn ​ que deben ser determinadas para maximizar o minimizar la función objetivo.

1. *Método Simplex*

El método Simplex es una técnica algebraica para resolver problemas de programación lineal con varias variables. A diferencia del método gráfico, que está limitado a dos variables, el Simplex puede manejar cualquier número de variables y restricciones.

**Pasos principales del método Simplex:**

1. **Convertir el problema a forma estándar**: Esto implica transformar todas las restricciones en ecuaciones igualadas a una constante, agregando variables de holgura si es necesario.
2. **Construcción de la tabla inicial**: El problema se representa en una tabla con las variables básicas y no básicas, la función objetivo y las restricciones.
3. **Identificar la variable entrante y saliente**: Se selecciona la variable entrante (aquella que mejor mejora el valor de la función objetivo) y la variable saliente (la que limita más el aumento de la variable entrante).
4. **Operaciones de pivote**: El objetivo es hacer que la columna de la variable entrante sea una variable básica (es decir, que tenga un 1 en su fila y ceros en las demás). Esto se logra mediante operaciones algebraicas en la tabla.
5. **Repetir el proceso**: El algoritmo continúa iterando sobre las tablas hasta que no se pueda mejorar más la función objetivo o se alcance una condición de parada.
6. Análisis del Programa
7. **Requisitos Funcionales:**

**Ingreso de datos**:

* Función objetivo con coeficientes modificables para las variables de decisión.
* Restricciones lineales con opciones para mayor que, menor que o igual a, agregando automáticamente variables de holgura.

**Cálculo**:

* Resolver el problema de optimización lineal usando el método Simplex.
* Generar tablas iterativas mostrando las soluciones parciales en cada iteración.
* Destacar la columna y fila pivote en cada tabla.

**Visualización**:

* Mostrar todas las iteraciones del proceso Simplex en tablas.
* Resaltar las soluciones intermedias y el resultado final.

**Interfaz de Usuario**:

* Ingreso de datos mediante campos de texto y menús desplegables.
* Mostrar las iteraciones en una interfaz gráfica clara con las tablas de cada iteración.

1. **Requisitos No Funcionales:**

**Usabilidad**:

* La interfaz debe ser intuitiva y fácil de usar, con una disposición clara de las tablas.
* Los mensajes de error deben ser precisos y útiles en caso de entrada incorrecta.

**Estética**:

* El diseño visual de la interfaz debe ser amigable y atractivo.

**Eficiencia**:

* La solución del problema debe ser rápida y eficiente, incluso con múltiples restricciones y variables.

1. Diseño del software

### **Diseño de clases**

* **Clase VentanaPrincipal:** Controla la ventana principal y maneja la interfaz gráfica general.
* **Clase MetodoSimplex:** Implementa toda la lógica relacionada con el método Simplex.
* **Clase InterfazSimplex:** Controla los elementos específicos de la interfaz para el método Simplex (entradas, restricciones, etc.).
* **Clase TablaSimplex:** Controla la visualización de la tabla y los pasos iterativos del algoritmo Simplex.

**Estructura del código en módulos**

Puedes dividir el código en varios archivos para separar las responsabilidades.

* main.py: Punto de entrada de la aplicación.
* ventana\_principal.py: Contiene la clase VentanaPrincipal.
* metodo\_simplex.py: Contiene la lógica de MetodoSimplex.
* interfaz\_simplex.py: Maneja la interfaz para la entrada de datos del método Simplex.
* tabla\_simplex.py: Controla la generación y visualización de la tabla Simplex.

**Diagrama de Flujo:**

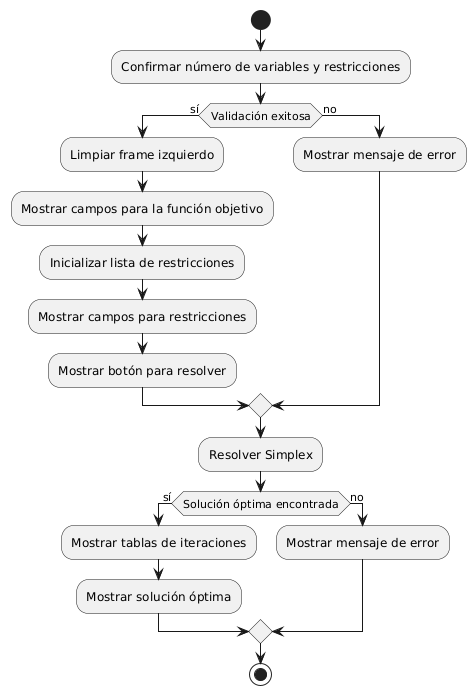
El diagrama de flujo muestra el proceso de resolución del problema de optimización lineal, desde la entrada de datos hasta la visualización de resultados.

Fig. 1 Diagrama de Flujo Método Simplex.

**Diagrama de clases:**

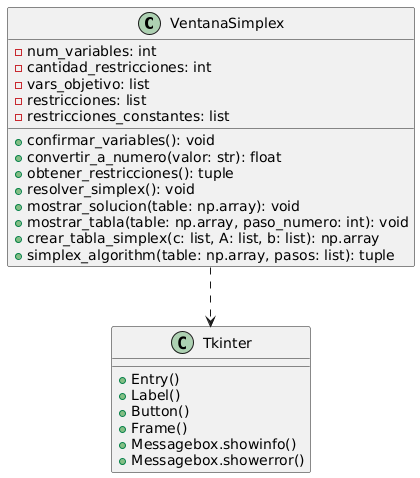
El diagrama de clases describe las clases del sistema, sus atributos, métodos, y las relaciones entre ellas****

Fig. 2 Diagrama de clases - Método Simplex.

**Diagrama de Secuencias:**

El diagrama de secuencias muestra cómo interactúan los objetos a lo largo del tiempo para llevar a cabo un proceso específico.

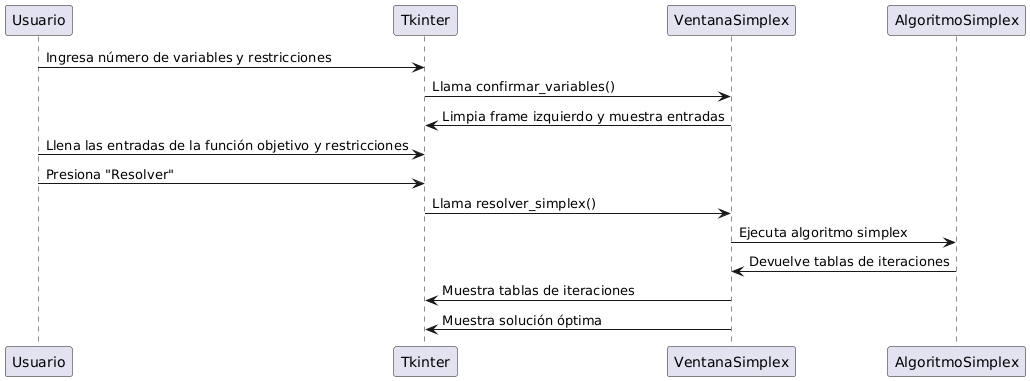


Fig. 3 Diagrama de Secuencia - Método Simplex.

**Diagrama de bloques:**

El diagrama de bloques muestra la arquitectura del sistema y cómo se dividen las diferentes partes del programa en bloques funcionales.

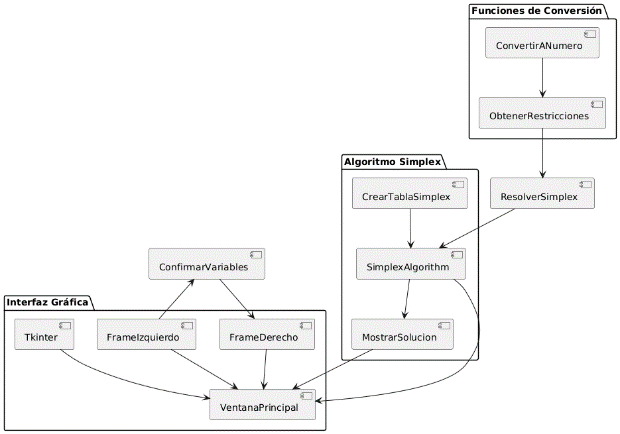
****

Fig. 4 Diagrama de bloques - Método simplex.

1. procedimiento

El desarrollo del programa se llevó a cabo utilizando el lenguaje de programación Python, aprovechando diversas bibliotecas. Se describen los pasos clave del procedimiento:

1. *Importación de Bibliotecas y Componentes.*

Se utilizan librerías como tkinter la cual sirve para tener una interfaz gráfica, también se usa messagebox para mostrar mensajes de información al usuario. Se utilizo la librería numpy la cual proporciona soporte para arrays multidimensionales, así como funciones avanzadas de matemáticas. Y la librería fracciones la cual permite trabajar con fracciones exactas en lugar de números flotantes.

import tkinter as tk

from tkinter import messagebox

import numpy as np

from fractions import Fraction

Fig. 5 Código importación de bibliotecas y componentes.

1. *Configuración Inicial de la Interfaz*

En la configuración inicial de la ventana principal, se establece el entorno gráfico mediante **Tkinter**, creando dos marcos principales:

* **frame\_izquierdo**: Para la entrada de datos.
* **frame\_doerecho**: Para la visualización de las tablas y la solución óptima.

ventana = tk.Tk()

ventana.title("Método Simplex")

ventana.geometry("1200x800")

ventana.configure(bg='#f8d7da')

# Crear frames para organizar la interfaz

frame\_izquierdo = tk.Frame(ventana)

frame\_izquierdo.pack(side=tk.LEFT, padx=10, pady=10)

frame\_derecho = tk.Frame(ventana)

frame\_derecho.pack(side=tk.RIGHT, padx=10, pady=10)

Fig. 6 Código para la configuración de la ventana.

1. *Entrada de Datos: Función Objetivo y Restricciones*

Una vez confirmada la cantidad de variables y restricciones, el código genera automáticamente los campos de entrada en la interfaz para que el usuario pueda ingresar los coeficientes de la función objetivo y las restricciones.

def confirmar\_variables():

global num\_variables, cantidad\_restricciones

num\_variables = int(numVar.get())

cantidad\_restricciones = int(resCan.get())

# Mostrar campos para la función objetivo

tk.Label(frame\_izquierdo, bg='#f8d7da', text="Función Objetivo (Z):").grid(row=0, column=0, padx=10, pady=10, columnspan=2)

Fig. 7 Código para la entrada de datos.

En esta sección se crean dinámicamente las entradas correspondientes a las variables de la función objetivo y restricciones, permitiendo la flexibilidad de definir diferentes problemas de optimización.

1. *Resolución del Método Simplex.*

Una vez ingresados los datos, se llama a la función **resolver\_simplex** para realizar el proceso de optimización. Esta función extrae los coeficientes ingresados por el usuario y construye la tabla inicial del método Simplex.

def resolver\_simplex():

# Obtener los coeficientes de la función objetivo

c = [convertir\_a\_numero(var.get()) for var in vars\_objetivo]

# Obtener las restricciones

A, b = obtener\_restricciones()

# Crear la tabla inicial del Simplex

table = crear\_tabla\_simplex(c, A, b)

pasos = [table.copy()]

# Ejecutar el algoritmo Simplex

optimal, pasos = simplex\_algorithm(table, pasos)

Fig. 8 Código para la solución del método Simplex.

Dos aspectos a resaltar son:

* **Obtención de la función objetivo y las restricciones** mediante la función obtener\_restricciones, que transforma las entradas en matrices adecuadas para el cálculo.
* **Construcción de la tabla Simplex inicial** en la función crear\_tabla\_simplex, que organiza los coeficientes en una tabla que seguirá el formato del método Simplex.

1. *Visualización de Resultados*

El código incluye funciones para visualizar cada iteración de la tabla Simplex en el marco derecho de la interfaz. Además, se resalta la **variable de entrada** y la **variable de salida** en cada iteración.

def mostrar\_tabla(table, paso\_numero):

row\_start = (paso\_numero - 1) \* (len(table) + 4)

tk.Label(frame\_derecho, bg='#f8d7da', text=f"{paso\_numero}er tablero Simplex").grid(row=row\_start, column=0, columnspan=len(table[0]) + 1, pady=10)

# Mostrar las celdas de la tabla con colores para las variables de entrada y salida

for i in range(len(table)):

for j in range(len(table[i])):

valor = round(table[i, j], 3) if isinstance(table[i, j], float) else table[i, j]

if j == col\_pivote and i == row\_pivote:

cell\_label = tk.Label(frame\_derecho, text=valor, borderwidth=1, relief="solid", width=10, bg="lightpink")

Fig. 9 Código para la Visualización de los resultados.

# **Resultados.**

El código desarrollado permite resolver problemas de optimización lineal mediante el método simplex. Los resultados incluyen una visualización de las tablas hechas en cada iteración, junto con una explicación textual detallada del valor de optimización.

**Datos de entrada:**

* Un espacio para ingresar la cantidad de variables que se quiere optimizar.
* Un espacio para definir el numero de restricciones que tendrá el problema.
* Un botón para confirmar estos valores y darnos paso a ingresar todos los datos del problema.

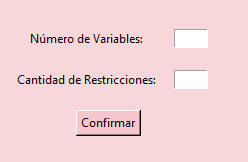


Imagen 1 Datos de entrada.

En esta parte se ingresa la cantidad de variables que tendrá nuestro problema a optimizar, así mismo se ingresa el numero de restricciones que este tendrá.

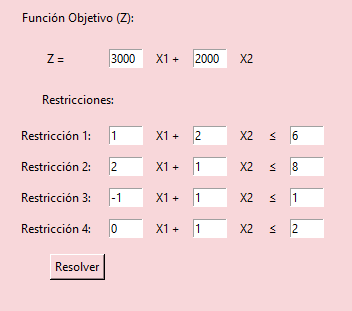


Imagen 2 Ingreso de los valores del problema.

En esta interfaz se ingresan los datos de nuestro problema de optimización, la función Z a optimizar y sus restricciones.

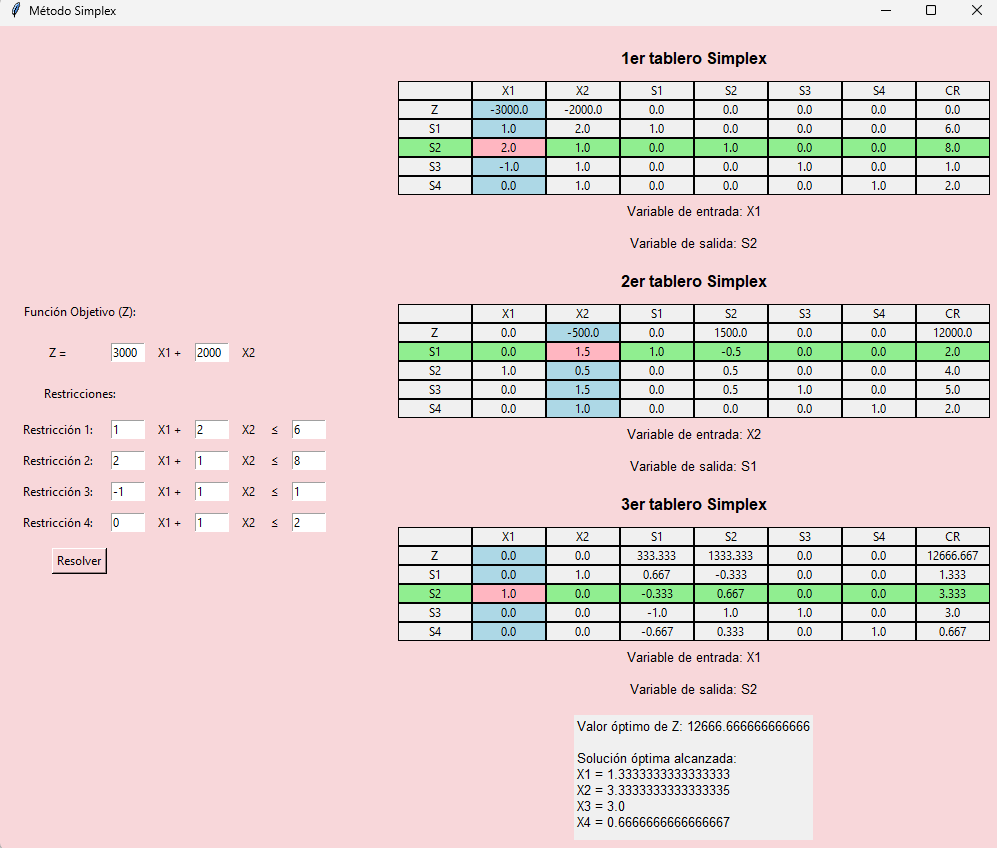


Imagen 3 Visualización de las tablas del método simplex.

Aquí se muestran las tablas iteradas junto con las filas, columnas y punto pivote.

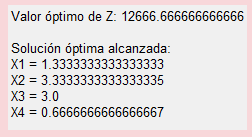


Imagen 4 Texto explicativo de la solución.

1. Conclusión.

El código implementado proporciona una solución interactiva para resolver problemas de optimización lineal utilizando el método Simplex. La interfaz gráfica facilita el ingreso de datos, mientras que el algoritmo Simplex se ejecuta de forma iterativa, presentando cada paso al usuario de manera visual. Las funciones principales están diseñadas para mantener el flujo del proceso Simplex claro, resaltando los elementos clave como las variables de entrada y salida en cada iteración.

Este enfoque permite que el usuario no solo obtenga la solución final, sino que también comprenda el proceso que llevó a dicha solución mediante la visualización de las tablas Simplex en cada iteración.

Referencias

1. Hillier, F. S., & Lieberman, G. J. (2014). Introduction to Operations Research (10th ed.). McGraw-Hill Education.
2. Winston, W. L. (2004). Operations Research: Applications and Algorithms (4th ed.). Cengage Learning.
3. Taha, H. A. (2011). Operations Research: An Introduction (9th ed.). Pearson.
4. Bazaraa, M. S., Jarvis, J. J., & Sherali, H. D. (2010). Linear Programming and Network Flows (4th ed.). Wiley.