Software para la Resolución Gráfica de Modelos de Programación Lineal

Juan Carlos Barrera Guevara, Diego Alejandro Machado Tovar
Facultad de Ciencias Básicas e Ingeniería, Universidad de los Llanos
Villavicencio, Colombia
jc.bguevara@unillanos.edu.co
damachado@unillanos.edu.co

I. INTRODUCCIÓN

La Investigación de Operaciones constituye una disciplina orientada al análisis y optimización de sistemas complejos mediante el uso de modelos matemáticos y técnicas cuantitativas. Dentro de este campo, la programación lineal se ha consolidado como una de las herramientas más relevantes, al permitir la formulación y resolución de problemas en los que se requiere maximizar o minimizar una función objetivo sujeta a un conjunto de restricciones lineales.

Si bien existen métodos algorítmicos de mayor alcance, el método gráfico conserva un papel fundamental en el ámbito académico, ya que facilita la comprensión de los principios básicos de la programación lineal. Su aplicación en modelos de dos variables permite ilustrar de manera clara los conceptos de factibilidad, región factible, frontera óptima y solución óptima, constituyendo así una base sólida para la posterior comprensión de técnicas más avanzadas.

En este contexto, el presente informe expone el diseño y desarrollo de un software orientado a la resolución de problemas de programación lineal mediante el método gráfico. La herramienta propuesta busca integrar los fundamentos teóricos con una implementación computacional que facilite la visualización de los modelos y brinde apoyo tanto en procesos de enseñanza como en escenarios de análisis aplicado.

II. REFERENTE TEÓRICO

A. Programación Lineal

La programación lineal es una técnica matemática empleada para optimizar la utilización de recursos limitados en un sistema. Su propósito consiste en determinar los valores de un conjunto de variables de decisión que maximizan o minimizan una función objetivo lineal, bajo un conjunto de restricciones también lineales. Estos modelos se utilizan ampliamente en problemas de asignación de recursos, planeación de la producción, transporte y gestión operativa, entre otros.

De manera formal, un modelo de programación lineal puede representarse como:

$$Maximizar\ o\ Minimizar\ Z = \sum_{j=1}^{n} c_j x_j$$

$$sujeto\ a:$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i\ ,\ i = 1, 2, \ldots, m$$

$$x_j \geq 0\ ,\ j = 1, 2, \ldots, n$$

En este esquema, xj corresponde a las variables de decisión, cj a los coeficientes de la función objetivo, aij a los coeficientes de las restricciones y bi la disponibilidad de recursos.

B. Fundamentos del Método Gráfico

Entre los procedimientos empleados para resolver modelos de programación lineal, el método gráfico se distingue por su simplicidad y su valor pedagógico. Aunque su aplicación está restringida a problemas con dos variables de decisión, este enfoque permite visualizar de manera clara los conceptos fundamentales de factibilidad, región factible y solución óptima. La técnica se basa en representar cada restricción como una recta en el plano cartesiano, delimitando así un polígono convexo conocido como región factible. Posteriormente, se identifican los vértices de dicha región, los cuales, de acuerdo con los principios de la programación lineal, son los únicos candidatos a contener la solución óptima del problema. Evaluando la función objetivo en cada uno de estos vértices, es posible determinar la alternativa que maximiza o minimiza el criterio de optimización planteado.

B. Limitaciones y Aplicaciones

El método gráfico presenta limitaciones inherentes a su naturaleza bidimensional, ya que no puede extenderse de forma práctica a problemas con tres o más variables. Sin embargo, su relevancia radica en su capacidad para introducir al analista en los fundamentos de la programación lineal, facilitando la comprensión de conceptos esenciales que serán generalizados en métodos más avanzados, como el algoritmo Simplex. En el ámbito académico, el método gráfico constituye una herramienta clave en la enseñanza de la optimización, mientras que en escenarios profesionales puede emplearse en problemas simples donde la representación visual favorezca la interpretación y comunicación de resultados.

III. PROCEDIMIENTO

El desarrollo del software siguió una metodología orientada a la construcción de un entorno interactivo que permitiera resolver problemas de programación lineal mediante el método gráfico. Para ello se implementó una aplicación de escritorio en Python que integra tanto una interfaz de usuario amigable como un motor de cálculo robusto, asegurando la correcta representación de la región factible y la identificación de la solución óptima.

La arquitectura del sistema se organizó en dos capas principales: una interfaz gráfica de usuario y un núcleo de procesamiento. En el front-end se utilizó el framework PySide6/Qt, donde componentes como QWidget, QToolBar, QStackedLayout y FigureCanvasQTAgg permitieron diseñar un entorno visual dinámico e interactivo. La visualización de la región factible y de los puntos de intersección se realizó mediante la integración de Matplotlib, embebido directamente en la aplicación, lo cual

posibilita manipular la gráfica de manera eficiente.

En el back-end se empleó Python 3 como lenguaje base, apoyado en librerías como NumPy para la gestión de operaciones numéricas y módulos estándar de la biblioteca nativa (os, re, math, typing y dataclasses) que facilitaron tanto el cálculo como el manejo de datos. Se desarrolló además un módulo propio encargado del parseo de restricciones, lo que permite interpretar expresiones introducidas por el usuario y transformarlas en estructuras adecuadas para el procesamiento matemático. El sistema implementa concurrencia mediante hilos gestionados con QThread, lo que garantiza que las operaciones intensivas de cálculo no interfieran con la fluidez de la interfaz.

Adicionalmente, se incorporaron funciones de reconocimiento óptico de caracteres (OCR) mediante la librería pytesseract, la cual requiere la instalación previa de Tesseract-OCR. Este componente, complementado con la librería Pillow para el preprocesamiento de imágenes, permite digitalizar restricciones capturadas desde documentos o imágenes, extendiendo la flexibilidad de las entradas admitidas. De manera complementaria, se integró un módulo de inteligencia artificial basado en el SDK de Groq, configurado para interactuar con el modelo Llama 3.3 70B Versatile a través de la API de Chat Completions. Esta integración se utiliza principalmente para la validación y soporte en la gestión de errores, con parámetros de configuración definidos en un archivo centralizado (config.py) que contempla temperatura, número máximo de tokens y tiempo de espera.

El flujo operativo del sistema se inicia con la entrada de datos, que puede realizarse de manera manual o mediante OCR. Posteriormente, las restricciones se procesan y almacenan en estructuras numéricas, tras lo cual se genera la representación gráfica de la región factible. El motor de cálculo identifica los puntos extremos de la misma, evalúa la función objetivo y determina la solución óptima, que es finalmente presentada al usuario en la interfaz. Para mantener la trazabilidad y modularidad, el sistema utiliza archivos de configuración gestionados con python-dotenv, lo que facilita su despliegue en distintos entornos sin requerir cambios en el código fuente.

La validación del software se llevó a cabo mediante la resolución de ejemplos estándar de programación lineal, verificando la correcta representación de restricciones, la delimitación precisa de la región factible y la obtención coherente de la solución óptima en comparación con resultados teóricos. Asimismo, la integración de librerías gráficas y de cómputo numérico permitió evidenciar la eficiencia del programa tanto en la ejecución como en la claridad de los resultados presentados al usuario.

IV. RESULTADOS

Con el fin de validar el correcto funcionamiento del software desarrollado, se realizaron pruebas sobre diversos problemas de programación lineal, aplicando el método gráfico como se planteó en la sección anterior. En esta sección se presentan los resultados obtenidos, los cuales incluyen la representación gráfica de la región factible, la localización de los vértices extremos y la determinación de la solución óptima para la función objetivo. Las evidencias gráficas y de salida del sistema permiten verificar que el comportamiento del programa es consistente con la teoría de programación lineal y demuestran su utilidad como herramienta de apoyo en la enseñanza y resolución de problemas de investigación de operaciones.

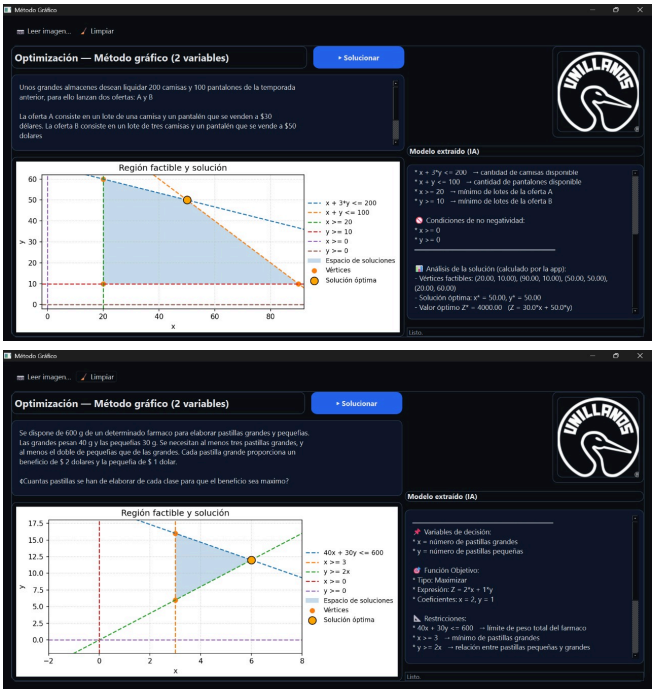

Figura 1. Representación gráfica de la región factible y solución óptima de dos problemas de programación lineal obtenida mediante el software desarrollado.

V. CONCLUSIONES

El desarrollo del software permitió materializar una herramienta computacional que integra los fundamentos teóricos de la programación lineal con las posibilidades tecnológicas actuales en el ámbito del cómputo interactivo. La implementación del método gráfico en un entorno de escritorio, apoyado en librerías como PySide6 y Matplotlib, demostró ser efectiva para la representación visual de restricciones y la identificación de soluciones óptimas, contribuyendo a la comprensión práctica de los modelos matemáticos estudiados en investigación de operaciones.

El diseño modular del sistema, que combina una interfaz de usuario intuitiva con un motor de cálculo robusto, aseguró un desempeño confiable en la resolución de problemas de dos variables. La incorporación de funcionalidades adicionales, tales como el reconocimiento óptico de caracteres y la integración con modelos de inteligencia artificial, amplió las capacidades del software, otorgándole un valor agregado en términos de flexibilidad y soporte automatizado al usuario.

Los resultados obtenidos evidencian que el programa cumple con los objetivos planteados, al ofrecer un medio eficaz para la visualización y resolución de problemas de programación lineal en entornos académicos y profesionales. Si bien el alcance del método gráfico se restringe a casos de dos variables, el sistema constituye una base sólida para futuras ampliaciones que integren métodos más avanzados de optimización aplicables a problemas de mayor complejidad.

VI. BIBLIOGRAFÍA

[1] H. A. Taha, Investigación de operaciones, 9.ª ed. México: Pearson Educación, 2017.

[2] F. S. Hillier y G. J. Lieberman, Introducción a la investigación de operaciones, 10.ª ed. México: McGraw-Hill, 2021.

[3] S. Boyd y L. Vandenberghe, Convex Optimization. Cambridge, Reino Unido: Cambridge University Press, 2004.

[4] W. McKinney, Python for Data Analysis, 3rd ed. Sebastopol, CA: O'Reilly Media, 2022.

[5] M. Summerfield, Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming. Upper Saddle River, NJ: Prentice Hall, 2007.

[6] J. D. Hunter, "Matplotlib: A 2D graphics environment," Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95, 2007.