UNIVERSIDAD CATÓLICA BOLIVIANA
SANTA CRUZ

INGENIERÍA EN INTELIGENCIA ARTIFICIAL

Informe técnico de

Sistema Interactivo de Optimización Matemática

Docente: Ing. Santos Claro Huayta Paucar

Integrantes: Nicole Lozada León

Erwin Alejandro Ojeda Justiniano

Materia: Optimización

Santa Cruz - Bolivia
Junio, 2025

1. Introducción

El presente proyecto tiene como objetivo el desarrollo de una aplicación interactiva basada en consola, implementada en Python, que permita a los usuarios modelar y resolver distintos tipos de problemas de optimización matemática. Esta herramienta cubre una variedad de clases de optimización (programación lineal, entera, no lineal, mixta y no lineal mixta), siguiendo la clasificación estándar en la materia de Optimización. El proyecto busca facilitar el aprendizaje y experimentación con técnicas de optimización, proporcionando una interfaz amigable y modular.

2. Alcance y funcionalidades

La aplicación permite al usuario:

- Definir conjuntos de variables o elementos y sus parámetros asociados.
- Especificar funciones objetivo para maximizar o minimizar.
- Introducir restricciones expresadas en formas algebraicas o sumatorias.
- Resolver problemas de:
    - Programación Lineal (LP)
    - Programación Entera (IP)
    - Programación No Lineal (NLP)
    - Programación Entera Mixta (MILP)
    - Programación No Lineal Mixta (MINLP)
- Visualizar resultados y mensajes de error o validación.

3. Decisiones de diseño

3.1 Elección del lenguaje y librerías

Python fue seleccionado por su potencia para prototipado rápido y por la amplia oferta de librerías de optimización. Pyomo es la librería central para la modelación matemática, debido a su flexibilidad para definir variables, parámetros, restricciones y objetivos para múltiples tipos de problemas. Streamlit se utilizó para construir la interfaz interactiva de consola web, aprovechando sus facilidades para crear formularios y desplegar resultados dinámicamente.

3.2 Arquitectura

Se decidió dividir la aplicación en clases separadas para cada tipo de optimización (LP, IP, NLP, MILP, MINLP). Cada clase encapsula la lógica para el parsing y validación de entradas, la construcción y resolución del modelo con Pyomo, y la interfaz con el usuario. Esto permite escalabilidad para añadir más tipos de problemas o funcionalidades específicas sin afectar al resto.

3.3 Interfaz de usuario

Se implementó un uso de formularios y widgets dinámicos que se adaptan según los datos ingresados, con un uso extenso de valores por defecto para facilitar pruebas rápidas y comprensión. La interfaz está dividida en pasos claros que guían al usuario: definición de variables y parámetros, especificación de función objetivo, definición de restricciones, y resolución y visualización de resultados. También se incluyen mensajes de error claros para validar entradas y evitar problemas en la resolución.

3.4 Manejo de expresiones algebraicas

Para NLP, MILP y MINLP, se implementó una evaluación dinámica de expresiones algebraicas usando eval con un diccionario seguro de variables Pyomo. Esto permite al usuario escribir funciones objetivo y restricciones arbitrarias, incrementando la flexibilidad. Se manejan excepciones para detectar errores sintácticos o semánticos en las expresiones.

3.5 Selección de solvers

Se optó por GLPK para LP, IP y MILP debido a su estabilidad como solver libre para problemas lineales y enteros. Para NLP y MINLP, se eligió Ipopt, dada su capacidad para problemas no lineales continuos. Además, se sugiere utilizar Bonmin para MINLP si Ipopt presenta dificultades, ya que Bonmin maneja variables enteras y no lineales simultáneamente.

3.6 Validación de datos

Se implementaron múltiples validaciones, incluyendo la comprobación de que cada parámetro tenga valores para todos los elementos y la validación de operadores y tipos de datos en

restricciones. También se valida que las variables usadas en expresiones estén definidas y se controla que el solver retorne una solución óptima antes de mostrar resultados.

4. Implementación detallada

4.1 Programación Lineal (LP)

Las variables de decisión son continuas y no negativas. Los parámetros son definidos por elemento, con funciones objetivo y restricciones basadas en sumatorias ponderadas. Las restricciones utilizan operadores variados como <=, >=, ==, etc. Se emplea ConstraintList para crear dinámicamente las restricciones, con un ejemplo predefinido que involucra muebles y parámetros de tamaño y precio.

4.2 Programación Entera (IP)

Las variables de decisión son enteras no negativas. La estructura es similar a la de LP, pero el dominio de las variables cambia a enteros. Se presenta un ejemplo que involucra cajas con parámetros de ganancia y tiempo, con restricciones y funciones objetivo adaptadas a este dominio.

4.3 Programación No Lineal (NLP)

Las variables son continuas reales. La función objetivo y las restricciones se definen por expresiones algebraicas arbitrarias ingresadas por el usuario. Se utiliza eval para evaluar expresiones en Pyomo, junto con el solver Ipopt. Un ejemplo ilustra una función cuadrática y restricciones no lineales.

4.4 Programación Entera Mixta (MILP)

En este caso, las variables son tanto enteras como continuas, definidas por el usuario. La función objetivo y las restricciones se expresan algebraicamente, con variables enteras y continuas declaradas explícitamente con sus respectivos dominios. Se emplea el solver GLPK para problemas mixtos lineales, con un ejemplo que incluye variables enteras y continuas.

4.5 Programación No Lineal Mixta (MINLP)

Las variables son enteras y continuas, con funciones objetivo y restricciones no lineales definidas por expresiones algebraicas. Se utiliza Ipopt (o se recomienda Bonmin) para resolver problemas con variables mixtas y no lineales. Las variables enteras se implementan asignando dominio en el objeto variable de Pyomo, y se presenta un ejemplo con una función cuadrática y restricciones no lineales mixtas.

5. Conclusiones y recomendaciones

El sistema cumple con los objetivos de ofrecer una plataforma versátil para modelar y resolver múltiples clases de problemas de optimización. La modularidad facilita la extensión y mantenimiento. La integración con Streamlit proporciona una experiencia de usuario intuitiva y accesible. El uso de Pyomo permite aprovechar múltiples solvers y tipos de variables.

Se recomienda implementar validaciones más robustas para expresiones algebraicas. También se sugiere añadir soporte para otros solvers con mejores capacidades en MINLP, como Couenne. Además, se propone extender la interfaz para incluir análisis de sensibilidad y reportes, y optimizar la gestión de estado para mejorar la experiencia en sesiones largas.

6. Referencias técnicas

- Pyomo documentation: https://pyomo.org/
- Streamlit documentation: https://streamlit.io/
- GLPK solver: https://www.gnu.org/software/glpk/
- Ipopt solver: https://coin-or.github.io/Ipopt/
- Bonmin solver: https://coin-or.github.io/Bonmin/