INSTITUTO TECNOLÓGICO DE OAXACA

DESARROLLO DE SOFTWARE PARA LA TOMA DE DECISIONES

ACTIVIDAD 4. IMPLEMENTACIÓN DE UN MODELO DE DECISIÓN

Modelo MBI de programación por metas

D O C E N T E :

MARTINEZ NIETO ADELINA

EQUIPO 3

I N T E G R A N T E S :

CARRANZA MARTÍNEZ MAGALI 22161018

CUEVAS ESTÉVEZ MARCOS GABRIEL 21160618

DÍAZ CRISTÓBAL ZAID 22161047

GARCÍA LEYVA JESUS 20161206

GARCIA SANTIAGO EMILIO GABRIEL 21160643

GÓMEZ LÓPEZ SARAÍ 21160651

GRUPO: 8SB

OAXACA DE JUÁREZ 25 DE FEBRERO, 2026

ÍNDICE

Descripción del modelo MBI de programación por metas

El modelo MBI de Programación por Metas es una técnica de optimización utilizada para resolver problemas que involucran múltiples objetivos que pueden entrar en conflicto. A diferencia de la programación lineal tradicional, que busca optimizar una sola función objetivo, este modelo permite establecer varias metas simultáneamente, asignando prioridades o pesos a cada una según su importancia.

El modelo funciona convirtiendo cada meta en una ecuación que incluye variables de desviación (OVER y UNDER), las cuales representan cuánto se sobrepasa o cuánto falta para alcanzar el objetivo establecido. Posteriormente, el sistema minimiza una combinación ponderada de estas desviaciones, buscando la solución que mejor equilibre todas las metas según su relevancia.

Las variables de desviación

OVER: Cuánto sobrepasa la meta

UNDER: Cuánto le falta para llegar a la meta

n productos del usuario

X1,X2,...,Xn

Metas

- Ganancia
- Presupuesto
- Horas

Cada meta se convierte en

$$\sum(\text{coeficientes} \times \text{variables}) - \text{OVER}i + \text{UNDER}i = \text{Meta}i$$

Ejemplos

Meta de ganancia:

$$\sum(\text{Utilidad}j \times Xj) - \text{OVER}1 + \text{UNDER}1 = \text{MetaGanancia}$$

Meta de horas:

$$\sum(\text{Horas}j \times Xj) - \text{OVER}2 + \text{UNDER}2 = \text{MetaHoras}$$

Meta de presupuesto:

$$\sum(\text{Costo}j \times Xj) - \text{OVER}3 + \text{UNDER}3 = \text{MetaPresupuesto}$$

Función objetivo del modelo

Minimizar $Z=\sum(Pesoi \times Desviacionesi)$

Z es la función objetivo que minimiza la suma ponderada de las desviaciones respecto a las metas establecidas. Significa encontrar los valores de las variables (X_1, X_2, etc.) que hagan que Z sea lo más pequeño posible, es decir. Minimizar el incumplimiento de metas.

Problemas que resuelve

Este modelo puede resolver problemas de toma de decisiones donde se deben considerar varios criterios al mismo tiempo,como ya se mencionó, por ejemplo:

- Un ejemplo de aplicación es la determinación de la cantidad óptima de producción de distintos productos, buscando maximizar la utilidad sin exceder restricciones de presupuesto y horas de trabajo, como se presenta en el capítulo 4.10 de la obra de Efraim Turban.

- Asignar recursos limitados entre distintas áreas cumpliendo metas financieras y operativas.

- Diseñar planes de inversión equilibrando rentabilidad y riesgo.

- Distribuir personal cumpliendo turnos mínimos y reduciendo horas extra.

En general es útil cuando no se busca lo máximo o lo mínimo de una sola variable, sino un equilibrio óptimo entre varias metas importantes.

Implementación del modelo en software

Lenguaje y enfoque
El software fue implementado en el lenguaje de programación **Python**, debido a su facilidad para el modelado matemático, su sintaxis clara y la disponibilidad de librerías especializadas en optimización.

Se utilizó un enfoque modular, separando la lógica del modelo matemático, la captura de datos, la resolución del modelo y la presentación de resultados, esto permite que sea más organizado y mantenible.

El proyecto se organizó en carpetas para separar responsabilidades:

- **main.py** Punto de entrada del sistema.

- **model/** Contiene el modelo matemático de programación por metas.

- **services/** Encargado de ejecutar el solver.

- **utils/** Manejo de entrada y salida de datos.

- **config.py** Configuración general.

- **requirements.txt** Librerías necesarias.

Así cada módulo cumple una función específica.

Modelo matemático
El modelo MBI implementado corresponde a una Programación por Metas, donde el objetivo es minimizar las desviaciones respecto a metas establecidas.

Se definieron:

- Variables de decisión:

- X1: cantidad del producto 1

- X2: cantidad del producto 2

- Variables de desviación:

- OVER y UNDER para cada meta (ganancia, horas y presupuesto)

Las ecuaciones se formularon de la siguiente manera:

- Meta de **ganancia**

- Meta de **horas** disponibles

- Meta de **presupuesto**

El modelo fue implementado usando la librería pulp, que permite formular problemas de programación lineal en Python.

La función objetivo minimiza la suma ponderada de desviaciones no deseadas, siguiendo el enfoque MBI (Minimización de Brechas Indeseadas) de la obra de Efraim Turban.

Dentro del archivo goal_model.py

- Se crea el modelo como problema de minimización.

- Se definen variables de decisión y desviación.

- Se agregan las ecuaciones de metas utilizando los parámetros ingresados por el usuario.

- Se define la función objetivo.

El sistema fue diseñado de manera genérica, permitiendo que cualquier usuario ingrese:

- ***Ganancia por producto***
- ***Horas requeridas por producto***
- ***Presupuesto por producto***
- ***Metas deseadas***

Esto hace que el DSS sea adaptable a distintos escenarios empresariales.

Manual de uso

¿Qué hace el programa?

- Permite definir productos con sus utilidades, horas requeridas y costos.
- Establece metas de ganancia, horas de trabajo y presupuesto.
- Calcula cuántas unidades producir de cada producto para minimizar las desviaciones respecto a las metas.

Modos de uso

Modo	Descripción
1 - GUI	Interfaz gráfica con formularios visuales
2 - CLI	Línea de comandos con preguntas interactivas

Cómo iniciar el programa

Abrir una terminal en la carpeta del proyecto y ejecutar:

python main.py

Para abrir directamente la interfaz gráfica:

python main.py --gui

Nota

Al iniciar sin argumentos, el sistema preguntará qué modalidad desea usar (GUI o CLI).

Datos que debe ingresar el usuario

El sistema solicitará la siguiente información para construir y resolver el modelo de programación por metas:

Número de productos (n)

> **¿Qué es?**
> La cantidad de tipos de productos que se incluirán en el modelo.

Campo	Rango válido	Ejemplo
Número de productos	Entero positivo (≥ 1)	2 (CC-7 y CC-8)

Datos por producto

Para cada producto, el sistema pedirá tres valores:

Dato	Descripción	Rango válido	Ejemplo (libro)
Utilidad unitaria	Ganancia por unidad producida en $	Número positivo	8,000 (CC-7)
Horas requeridas	Horas de trabajo por unidad producida	Número positivo	300 (CC-7)
Costo unitario	Costo de producción por unidad en $	Número positivo	10,000 (CC-7)
Mínimo a producir	Unidades mínimas obligatorias (mercado)	Entero ≥ 0	100 (CC-7)

Restricciones de mercado

El campo 'mínimo a producir' corresponde a los requerimientos de mercado (ej: $producto_1 \geq 100$, $producto_2 \geq 200$). Son obligatorios y siempre los respetará, a diferencia de las metas que pueden desviarse con penalización.

Metas del modelo

Las metas son los que el sistema intentará alcanzar para cada restricción:

Meta	Descripción	Ejemplo
Meta de ganancia	Nivel de ganancia total deseado en $	5,000,000

Meta de horas	Total de horas de trabajo disponibles	200,000
Meta de presupuesto	Presupuesto total disponible en $	8,000,000

Pesos o Ponderaciones

Los pesos son el componente más importante del modelo. Determinan qué tan grave es desviarse de cada meta y en qué dirección. Las operaciones minimizan la suma ponderada de las desviaciones.

¿Qué significa cada peso?

Peso	¿Qué penaliza?	Efecto si peso es alto
Peso de ganancia	No alcanzar la meta de ganancia	Se prioriza alcanzar la ganancia
Peso de horas	Exceder las horas de trabajo	Se evita usar más horas de las planeadas
Peso de presupuesto	Exceder el presupuesto	Se evita gastar más del presupuesto

Valores recomendados y su interpretación

Los pesos deben ser números positivos. No hay un límite técnico, pero los valores deben ser coherentes entre sí porque se comparan relativamente:

Escenario	Pesos sugeridos	Resultado esperado
Priorizar ganancia sobre todo	peso_ganancia = 5,000 peso_horas = 10 peso_presupuesto = 10	Se sacrificará horas y presupuesto para alcanzar la ganancia
Controlar costos ante todo	peso_ganancia = 100 peso_horas = 50 peso_presupuesto = 2,000	Se respetará el presupuesto aunque se sacrifique algo de ganancia
Solo importa la ganancia	peso_ganancia = 10,000 peso_horas = 1 peso_presupuesto = 1	Maximizará ganancia ignorando casi todas las demás metas

Regla práctica

Si el peso de A es 10 veces mayor que el de B, el solver considerará desviarse de A como 10 veces más costoso que desviarse de B. Los pesos son comparativos, no absolutos. Se puede usar cualquier escala siempre que las proporciones reflejen sus prioridades.

Interpretación de resultados

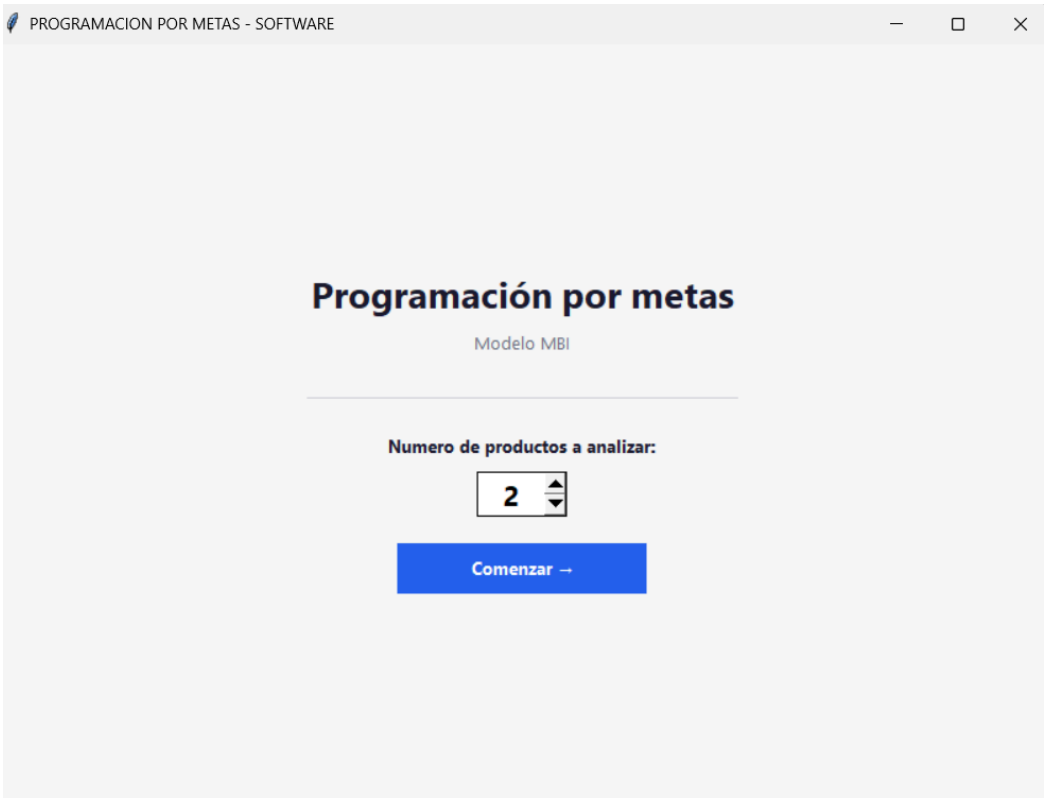
Variables de decisión

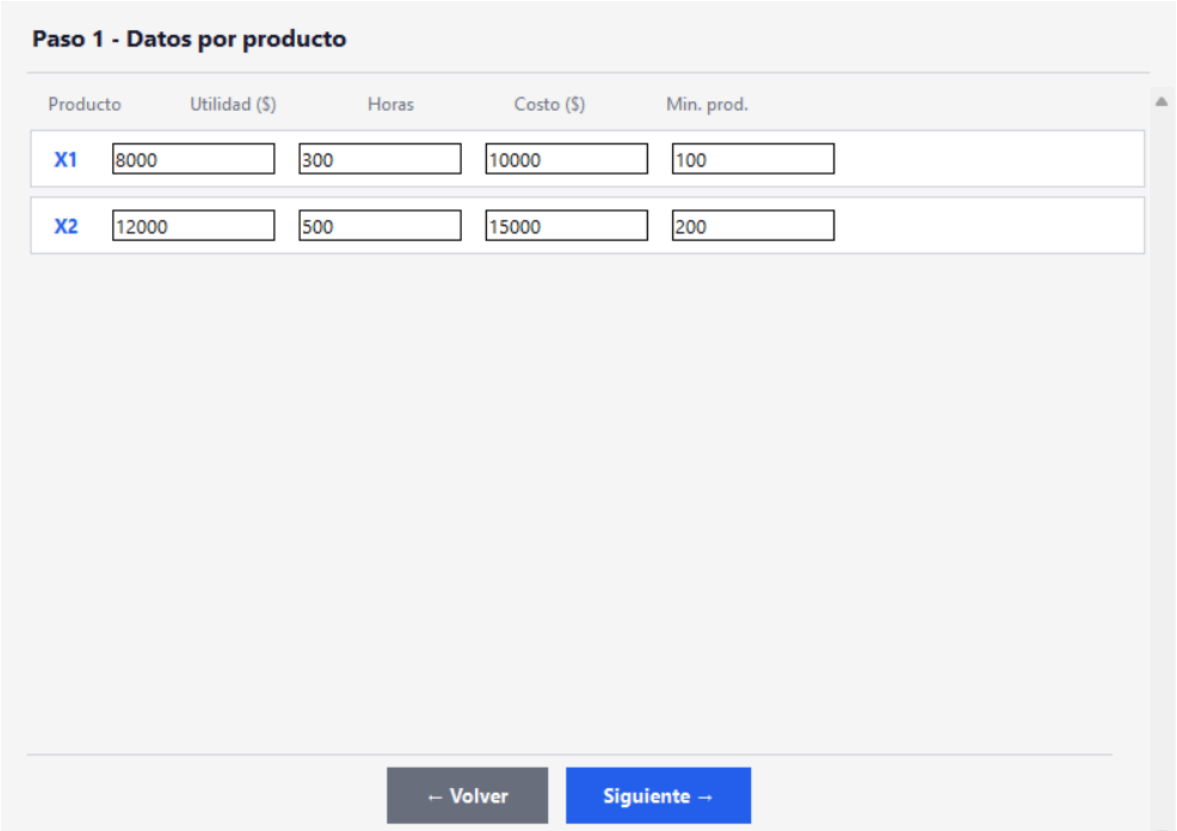
Son las cantidades a producir de cada producto. Ejemplo: Producto 1 = 500 significa producir 500 unidades del Producto 1.

> **Resultado óptimo**
> Una solución ideal tendría todas las variables de desviación en 0, lo que significaría que todas las metas se cumplieron exactamente. En la práctica, se busca la combinación que minimiza las penalizaciones según los pesos asignados.

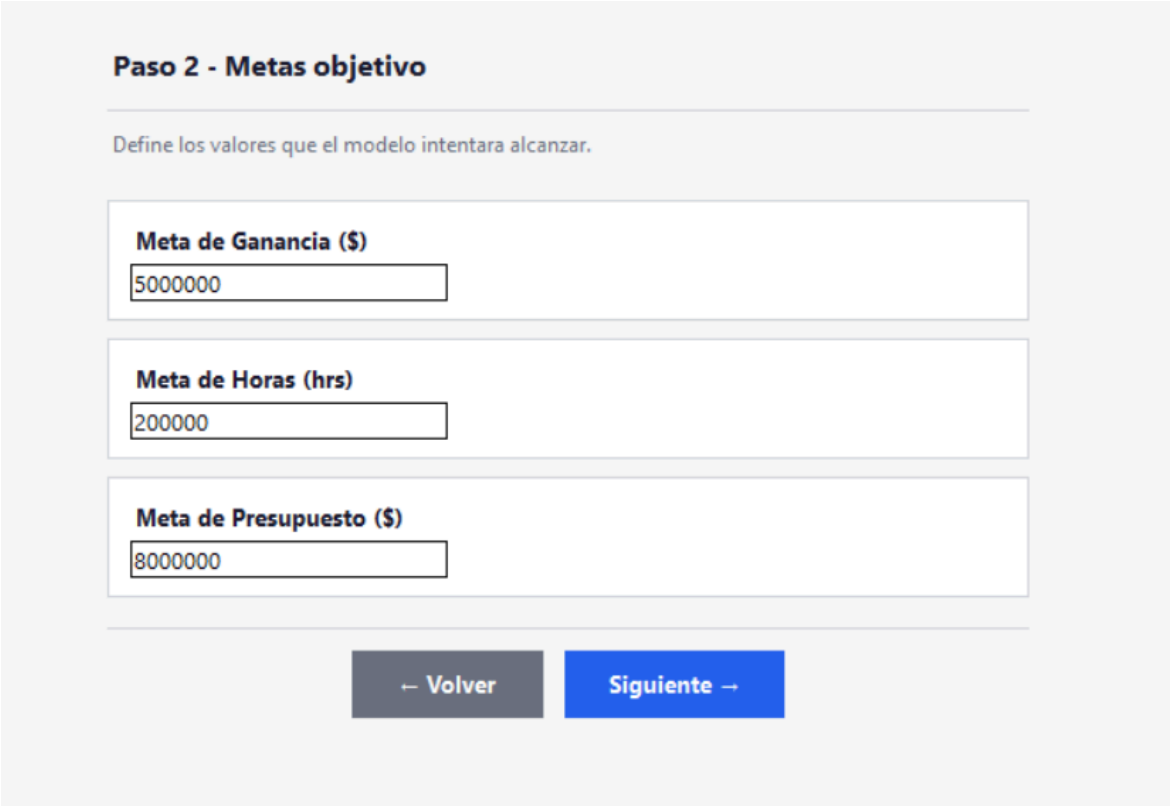
Ejemplo

Utilizando para la prueba los valores de la obra de Efraim Turban.





Y las metas y prioridades correspondientes.



Paso 3 - Pesos de penalizacion

Peso 0 = esa desviacion no penaliza. Mayor peso = mas importante.

Ganancia

OVER1 - superar meta: 0

UNDER1 - no alcanzar: 1000

Horas de trabajo

OVER2 - exceder horas: 50

UNDER2 - quedar corto: 10

Presupuesto

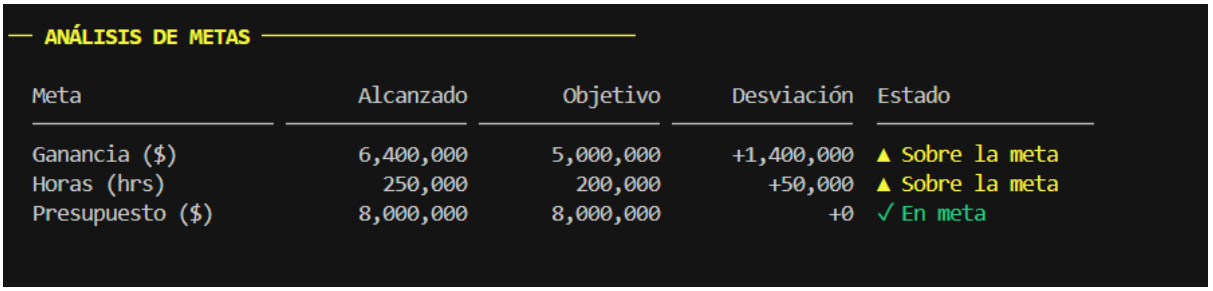
OVER3 - exceder presup.: 100

UNDER3 - quedar corto: 20

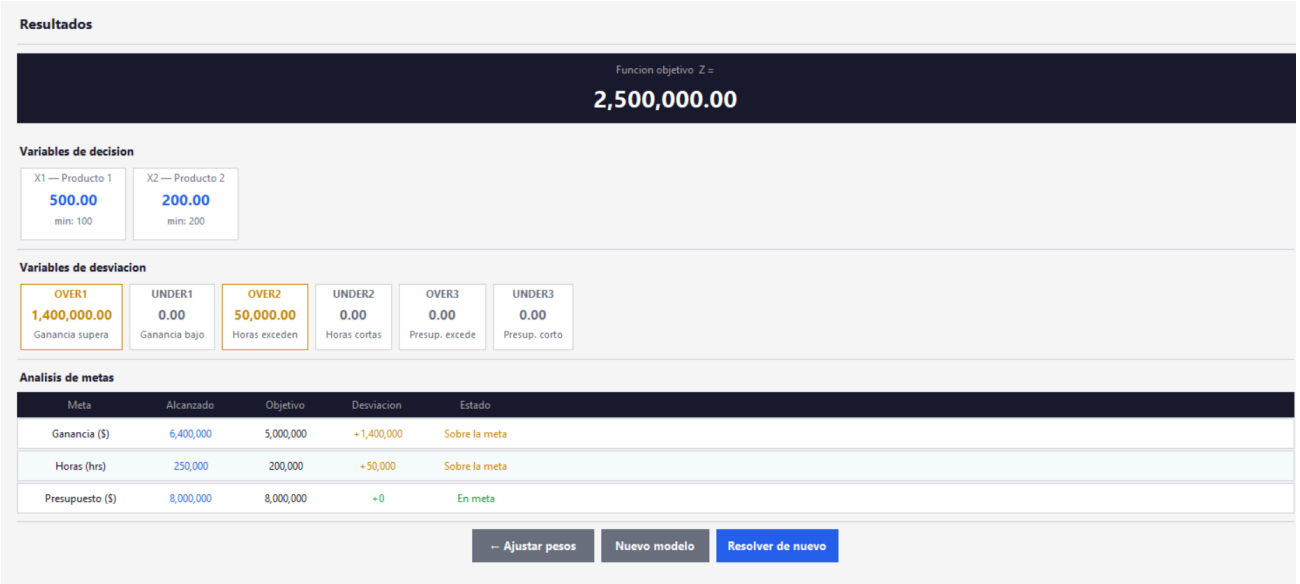
← Volver | Resolver

Resultados

En línea de comandos, los resultados se obtienen de esta manera:



En la interfaz de usuario los resultados se obtienen:



Se analizaron 2 productos (X1 y X2) con el modelo buscando minimizar las desviaciones respecto a 3 metas simultáneas.

Conclusiones

- La solución indica producir 500 unidades de X1 y 200 unidades de X2, ambos por encima de sus mínimos requeridos (100 y 200 respectivamente).
- La ganancia alcanzó $6,400,000 superando la meta de $5,000,000 en $1,400,000. Como el peso de OVER1 es 0, esta desviación no penaliza, lo cual es positivo.
- Las horas usadas fueron 250,000 contra una meta de 200,000, excediendo en 50,000 horas. Esta es la única desviación que penaliza y genera todo el costo de Z.
- El presupuesto se utilizó exactamente en $8,000,000, quedando perfectamente en meta sin ninguna desviación.
- Todo el costo proviene exclusivamente del exceso de horas (50,000 hrs × peso 50 = 2,500,000). Las otras dos metas no contribuyen al costo.

Repositorio en github

https://github.com/DiCZDC/ImplementacionDSS.git