Proyecto Final – Complejidad y Optimización

Carlos Alberto Camacho Castaño -2160331
Juan David Valencia Montalvo - 2160103

***Resumen*— Este proyecto implementa un modelo de optimización utilizando MiniZinc para maximizar la suma de valores adyacentes en dos matrices, sujeto a restricciones como la selección limitada de casillas y la no adyacencia entre las seleccionadas. El sistema incluye una interfaz gráfica en Python para cargar datos de entrada y procesar los resultados del modelo. La solución generada ofrece una herramienta eficiente para resolver problemas de ubicación óptima en matrices bidimensionales.**

I. Introducción

En el desarrollo de soluciones computacionales para problemas de optimización, es esencial combinar modelado matemático con herramientas de programación que permitan automatizar la ejecución y análisis de resultados. Este informe presenta un sistema diseñado para resolver un problema de selección óptima en matrices bidimensionales, donde el objetivo es maximizar la suma de valores adyacentes de dos matrices, sujeto a restricciones de capacidad y separación.
El sistema se basa en un modelo desarrollado en MiniZinc, este modelo utiliza estructuras como matrices y funciones para calcular sumas de valores adyacentes, además de implementar restricciones sobre la selección de casillas predefinidas y la no adyacencia.
Para facilitar la interacción, se desarrolló una interfaz gráfica en

Python. Esta interfaz permite a los usuarios cargar archivos de entrada, generar automáticamente los parámetros requeridos en formato .dzn, ejecutar el modelo de MiniZinc, y mostrar los resultados de manera estructurada.

II. Modelado

A continuación tenemos una explicación detallada acerca del modelado en minizinc que aborda las restricciones planteadas.

- *Parámetros y variables:*

n: Dimension de las matrices (nxn)

Max_selecciones: Máximo número de casillas seleccionables.

$M_1, M_2 \in \mathbb{Z}^{nxn}$: Dos matrices de valores enteros.

$P \in \{0,1\}^{nxn}$: Matriz de preseleción, donde $P\ [i,j]$ indica que la casilla (i,j) está seleccionada previamente.

- *Valores de decisión*

$S \in \{0,1\}^{nxn}$: Matriz binaria que indica si una casilla (i,j) es seleccionada $(s[i,j] = 1)$ o no $(s[i,j] = 0)$

- *Función Objetivo*

Maximizar la suma de los valores adyacentes en ambas matrices para las casillas seleccionadas:

$$\text{Maximizar: } \sum_{i=1}^{n}\sum_{j=1}^{n} S[i,j] \cdot \left(\sum_{(x,y)\in \text{ady}(i,j)} M_1[x,y] + \sum_{(x,y)\in \text{ady}(i,j)} M_2[x,y] \right),$$

Donde ady(i,j) representa casillas adyancentes a (i,j), es decir:

$$\text{ady}(i,j) = \{(x,y) \mid 1 \le x,y \le n, |x-i| \le 1, |y-j| \le 1, (x,y) \ne (i,j)\}.$$

- *Restricciones*

Las casillas preseleccionadas deben permanecer seleccionadas:

$$P[i,j] = 1 \implies S[i,j] = 1, \quad \forall i,j \in \{1,\dots,n\}.$$

La suma de las casillas seleccionadas (nuevas y preseleccionadas) no debe exceder el límite máximo:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} S[i,j] \le \quad \text{max\_selecciones}$$

No se permite seleccionar casillas adyacentes entre sí:

$$S[i,j] = 1 \implies S[x,y] = 0, \quad \forall (x,y) \in \text{ady}(i,j), \forall i,j \in \{1,\dots,n\}.$$

- *Aspectos relevantes*

Cálculo de la suma de valores adyacentes

Para una casilla seleccionada (i,j), la suma de valores adyacentes en una matriz M se define como:

$$\text{Suma\_adaycentes } (M,i,j) = \sum_{(x,y)\in \text{ady}(i,j)} M[x,y].$$

Relación entre preseleccionadas y nuevas selecciones

El número de nuevas selecciones se calcula como:

$$\text{nuevas\_selecciones} = \sum_{i=1}^{n}\sum_{j=1}^{n} S[i,j] \cdot (1 - P[i,j]).$$

Básicamente, el modelo desarrollado resuelve un problema de optimización combinatoria para seleccionar casillas en dos matrices M_1 y M_2, maximizando la suma de los valores en las casillas adyacentes seleccionadas. El modelo utiliza variables binarias para representar si una casilla está seleccionada o no, y su función objetivo calcula la suma ponderada de los valores adyacentes en ambas matrices para las casillas seleccionadas. Las restricciones incluyen mantener las casillas preseleccionadas, limitar el número máximo de selecciones y evitar la selección de casillas adyacentes.

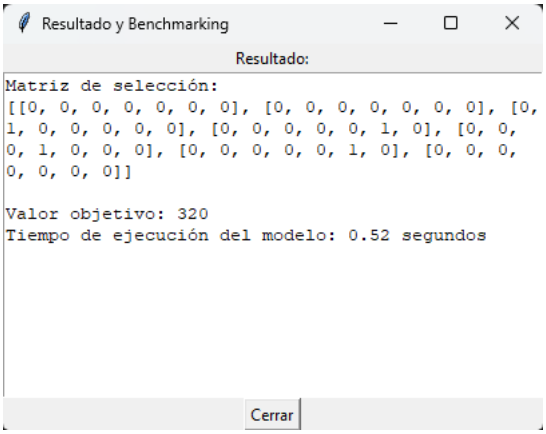
En términos de implementación, el archivo en MiniZinc define estas reglas usando estructuras como arrays para las matrices y expresiones como forall para aplicar las restricciones. Por ejemplo, la matriz de preselección asegura que las casillas predefinidas se mantengan seleccionadas, mientras que la lógica de no adyacencia prohíbe seleccionar casillas vecinas usando índices y condiciones. La suma de valores adyacentes se calcula mediante una función que recorre las casillas vecinas dentro de los límites de la matriz. En Python, se diseña una interfaz gráfica para cargar datos desde un archivo, generar los parámetros en formato .dzn y ejecutar el modelo. Finalmente, los resultados se visualizan de manera intuitiva, facilitando su interpretación. Este enfoque combina técnicas de modelado matemático con herramientas prácticas para resolver problemas reales de optimización.

Tanto la función principal, como el modelado en MiniZinc serán explicados en el video.

III. Pruebas

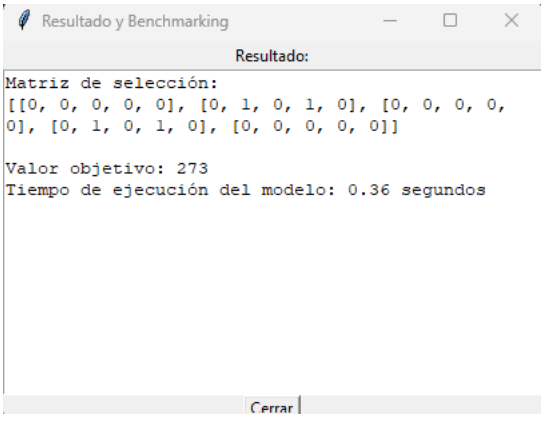
- *Prueba 1.*

```
2
5 5
2 1
7
2 3 8 1 6 5 0
9 4 7 3 2 1 8
5 0 1 9 4 7 3
8 3 2 5 7 0 4
1 6 9 4 3 8 2
4 7 0 8 1 6 9
3 2 5 0 9 4 7
6 2 9 3 0 7 5
1 5 8 2 4 6 3
4 9 1 0 7 3 6
2 7 6 8 9 1 4
3 4 0 5 8 9 2
9 1 2 4 6 0 8
5 3 7 9 2 4 1
2
```



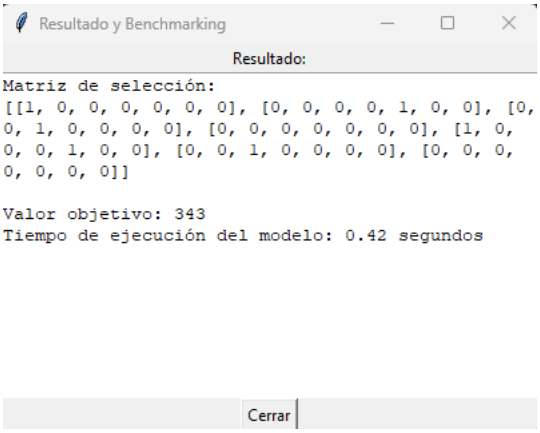
- *Prueba 2*

```
2
1 1
3 3
5
3 2 8 6 1
7 4 3 0 9
5 9 1 4 2
8 7 6 5 3
4 2 1 0 6
2 9 3 1 8
1 0 7 6 5
3 8 2 4 7
6 5 1 0 9
7 0 5 9 1
3
```



- *Prueba 3*

3
0 0
2 2
4 4
7
4 5 1 6 7 2 3
1 8 4 9 0 3 5
6 3 8 2 1 7 9
9 0 2 5 3 4 8
7 6 1 4 8 9 2
8 9 0 1 6 3 7
1 0 0 9 9 9 7
0 0 0 6 0 2 0
1 8 4 9 0 3 5
6 3 2 2 1 7 9
9 0 0 5 3 4 8
7 6 1 4 0 0 0
8 9 0 1 1 3 7
1 9 0 9 2 9 7
3



- *Discusión de resultados*

En general, el valor objetivo obtenido en cada prueba varía en función del tamaño de las matrices, la cantidad de programas a ubicar, y las posiciones preseleccionadas. Por ejemplo, la Prueba 1 alcanzó un valor objetivo más alto (320) que la Prueba 2 (273), probablemente debido al mayor tamaño de las matrices y al número de opciones disponibles para optimizar la solución. Sin embargo, en la Prueba 3, aunque las matrices son también grandes, el valor objetivo alcanzado (343) refleja un escenario más complejo por la mayor cantidad de posiciones preseleccionadas, lo que limita las opciones de colocación pero puede generar soluciones más específicas.
En términos de tiempo de ejecución, se observa que este depende tanto del tamaño de las matrices como de la complejidad del problema. La Prueba 2, siendo la más pequeña, fue la más rápida (0.36 segundos), mientras que la Prueba 1 y la Prueba 3, que tienen configuraciones más grandes o complejas, requirieron más tiempo de procesamiento (0.52 y 0.42 segundos, respectivamente). Esto indica que el modelo escala razonablemente bien, pero se enfrenta a un incremento gradual en el tiempo de ejecución a medida que aumentan las restricciones y las dimensiones de entrada.
En resumen, estas pruebas evidencian que el modelo puede manejar diferentes configuraciones de entrada de manera eficiente, manteniendo tiempos de ejecución bajos mientras maximiza el valor objetivo. Esto sugiere que el enfoque es adecuado para resolver problemas de esta naturaleza en un rango moderado de tamaños y complejidad.

IV. Conclusiones

En cuanto a la implementación, se logró integrar de manera efectiva técnicas de programación matemática mediante MiniZinc, junto con una interfaz gráfica interactiva en Python. Esto permitió procesar datos de entrada y resolver problemas de optimización de forma automatizada. Los resultados obtenidos mostraron cómo el modelo responde a diferentes configuraciones, validando su capacidad para seleccionar las casillas más beneficiosas según las restricciones definidas.

Este proyecto representa un esfuerzo significativo para modelar y optimizar un sistema basado en datos de población y empleo, utilizando conceptos de programación matemática y algoritmos eficientes. La implementación combina análisis lógico, notación matemática formal y un enfoque computacional robusto para resolver problemas complejos. A través de pruebas específicas, se logró evaluar el comportamiento del modelo en diferentes escenarios, mostrando cómo los parámetros iniciales y las configuraciones influyen en los resultados obtenidos. Sin embargo, el proyecto también pone de manifiesto la importancia de validar cuidadosamente los datos de entrada y garantizar que coincidan con las especificaciones del modelo para evitar errores en la ejecución. En términos generales, este trabajo demuestra cómo los principios de codificación y optimización pueden integrarse eficazmente para abordar desafíos prácticos en la toma de decisiones, en este caso, ayudar al profe con su problema y estar más cerca del trabajo soñado de los 300 USD.

V. Referencias

[1] https://www.youtube.com/watch?v=ObLwtx9c1so&t=673s&ab_channel=ProfeFernando
[2] https://www.youtube.com/watch?v=2YlW46uQn2o&t=81s
[3] https://www.youtube.com/watch?v=2e645ibBIZo
[4] https://www.ieee.org/conferences/publishing/
[5] https://scielo.conicyt.cl/pdf/rcp/v80n1/art10
[6] https://docs.minizinc.dev/en/stable/python.html