Informe Proyecto:
Complejidad y Optimización ADA II

Autores:
Arango Rodas, Manuel - 2259571
García Castañeda Alex -2259517
Gomez Agudelo, Sebastián - 2259474
Henao Aricapa, Stiven - 2259603
Universidad del Valle - Tuluá

Resumen

Este informe presenta el desarrollo e implementación de un modelo de optimización en **MiniZinc** para abordar problemas de ubicación óptima, maximizando ganancias bajo restricciones específicas. El modelo se integró con **PyQt5** y la API de MiniZinc, lo que permitió diseñar una interfaz gráfica intuitiva para visualizar las soluciones obtenidas. El programa fue sometido a evaluación en distintos escenarios, demostrando eficiencia, consistencia y cumplimiento de los requisitos establecidos. En este documento se exponen los resultados obtenidos a través de ejemplos prácticos y se analizan las conclusiones relacionadas con el desempeño del modelo ante diferentes configuraciones y niveles de complejidad.

I. Introducción

Este modelo de optimización implementado en MiniZinc busca determinar la ubicación óptima de programas adicionales en una matriz geográfica, considerando restricciones específicas y maximizando una función de ganancia. Utiliza como entrada parámetros como coordenadas predefinidas, matrices que describen características del entorno (segmento poblacional y entorno empresarial), y el número de programas a implementar. Las restricciones incluyen evitar ubicaciones adyacentes, cumplir con valores mínimos de población y entorno, y respetar las posiciones previamente definidas. El resultado final entrega una solución que equilibra las condiciones del problema y optimiza el impacto de las decisiones tomadas.

II. Resumen del Modelo

Parámetros del Modelo:

num_posiciones_existentes: Cantidad de posiciones predefinidas.

ciudades: Coordenadas de las posiciones existentes en una matriz.

tamano_matriz: Dimensiones de la matriz.

matriz_segmento_poblacion y matriz_entorno_empresarial: Matrices que representan características relevantes (segmento poblacional y entorno empresarial).

num_programas: Cantidad de programas adicionales (ubicaciones) a construir.

III. Variables del Modelo

ubicaciones: Matriz binaria (0 o 1) que determina la ubicación de las construcciones.

ubicaciones_predefinidas: Matriz que fija las ubicaciones predefinidas según las coordenadas en ciudades.

IV. Restricciones

Ubicaciones predefinidas: Las posiciones existentes deben conservar su valor como 1.

Cantidad de ubicaciones: La suma total de ubicaciones (1) debe ser igual a *num_programas* más las posiciones existentes.

Zonas adyacentes: No se pueden colocar ubicaciones en celdas adyacentes (horizontal, vertical o diagonal).

Criterios de viabilidad: La suma de valores en las matrices *matriz_segmento_poblacion* y *matriz_entorno_empresarial* alrededor de una celda seleccionada debe cumplir con:

Segmento poblacional ≥ 25

Entorno empresarial ≥ 20

V. Función Objetivo

Maximizar la ganancia total basada en las sumas de valores en las matrices *matriz_segmento_poblacion* y *matriz_entorno_empresarial* alrededor de las ubicaciones seleccionadas.

Se calcula también la ganancia específica para las coordenadas de las posiciones existentes (*ganancia_ciudades*).

VI. Salida del Modelo

Imprime las ubicaciones seleccionadas como una lista de valores binarios.

Muestra las ganancias totales (*ganancia*) y las ganancias en posiciones predefinidas (*ganancia_ciudades*).

VII. Interfaz Gráfica

La interfaz gráfica de usuario (GUI) de esta aplicación se desarrolló utilizando **PyQt5**, una biblioteca que permite crear interfaces gráficas en Python de manera eficiente y flexible. Para la integración con el motor de resolución de problemas, se empleó la API de **MiniZinc**, que facilita la interacción con modelos de optimización desde Python. Esta combinación permite una experiencia de usuario interactiva y eficiente en la resolución de problemas complejos.

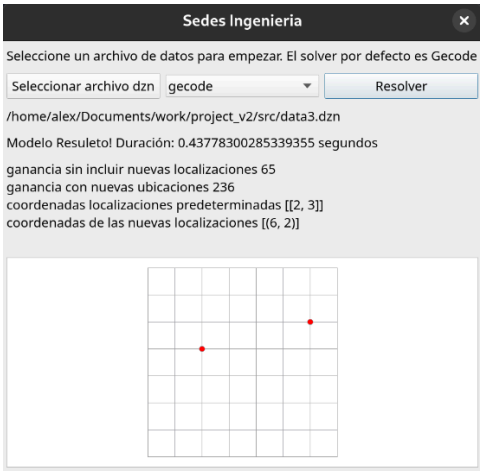
VIII. Discusión de resultados

Los resultados del modelo de optimización fueron evaluados utilizando cuatro casos de prueba representativos, diseñados para explorar distintas configuraciones de entrada y restricciones. A continuación, se describen los escenarios analizados y se discuten las implicaciones de los resultados obtenidos:

Entrada ejemplo 1

```
num_posiciones_existentes=1;

ciudades=[|2, 3|];

tamano_matriz=7;

matriz_segmento_poblacion=[|1, 1, 1, 4, 5, 6, 7
                          |1, 1, 1, 5, 6, 7, 1
                          |1, 1, 5, 6, 7, 1, 2
                          |4, 5, 6, 7, 1, 2, 3
                          |5, 6, 7, 1, 2, 3, 4
                          |6, 7, 1, 2, 3, 4, 5
                          |7, 1, 2, 3, 4, 5, 6|];

matriz_entorno_empresarial=[|7, 9, 5, 4, 3, 2, 100
                          |6, 5, 4, 3, 2, 1, 7
                          |9, 4, 3, 2, 1, 7, 6
                          |4, 3, 2, 1, 7, 6, 5
                          |3, 2, 1, 7, 6, 5, 4
                          |2, 1, 7, 6, 5, 4, 3
                          |9, 7, 6, 5, 4, 3, 9|];

num_programas=1;
```

Salida ejemplo 1



Ejemplo 1. Caso base, el algoritmo identifica y selecciona la solución óptima de manera directa, dado que resulta evidente incluso mediante una inspección visual de los datos.

Entrada ejemplo 2

```
num_posiciones_existentes=3;

ciudades=[|2,3
|2,6
|6,2|];

tamano_matriz=7;

matriz_segmento_poblacion=[|1, 1, 1, 4, 5, 6, 7
                          |1, 1, 1, 5, 6, 7, 1
                          |1, 1, 5, 6, 7, 1, 2
                          |4, 5, 6, 7, 1, 2, 3
                          |5, 6, 7, 1, 2, 3, 4
                          |6, 7, 1, 2, 3, 4, 5
                          |7, 1, 2, 3, 4, 5, 6|];

matriz_entorno_empresarial=[|7, 9, 5, 4, 3, 2, 1
                          |6, 5, 4, 3, 2, 1, 7
                          |9, 4, 3, 2, 1, 7, 6
                          |4, 3, 2, 1, 7, 6, 5
                          |3, 2, 1, 7, 6, 5, 4
                          |2, 1, 7, 6, 5, 4, 3
                          |9, 7, 6, 5, 4, 3, 9|];
```

Salida ejemplo 2

Ejemplo 2. En este escenario, se evalúa el desempeño del algoritmo incrementando ligeramente las restricciones necesarias para maximizar la ganancia de las nuevas ubicaciones requeridas.

Entrada ejemplo 3

```
num_posiciones_existentes=3;

ciudades=[|1,1
|5,5
|2,6|];

tamano_matriz=7;

matriz_segmento_poblacion=[|1, 1, 1, 4, 5, 6, 7
                           |1, 1, 1, 5, 6, 7, 1
                           |1, 1, 5, 6, 7, 1, 2
                           |4, 5, 6, 7, 1, 2, 3
                           |5, 6, 7, 1, 2, 3, 4
                           |6, 7, 1, 2, 3, 4, 5
                           |7, 1, 2, 3, 4, 5, 6|];

matriz_entorno_empresarial=[|7, 9, 5, 4, 3, 2, 1
                           |6, 5, 4, 3, 2, 1, 7
                           |9, 4, 3, 2, 1, 7, 6
                           |4, 3, 2, 1, 7, 6, 5
                           |3, 2, 1, 7, 6, 5, 4
                           |2, 1, 7, 6, 5, 4, 3
                           |9, 7, 6, 5, 4, 3, 9|];

num_programas=1;
```

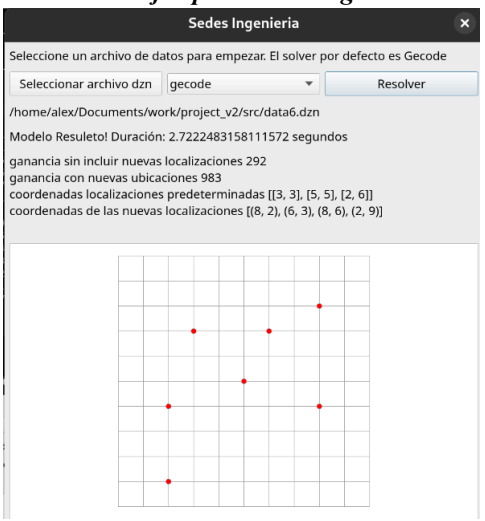
Salida ejemplo 3

Ejemplo 3. Como se puede observar, no existe una solución factible debido a que en la posición (1,1), especificada en la instancia, no se cumple la restricción de que el segmento poblacional debe ser mayor o igual a 25. Esto provoca que el modelo no pueda satisfacer las condiciones establecidas y, en consecuencia, no se genere una solución válida.

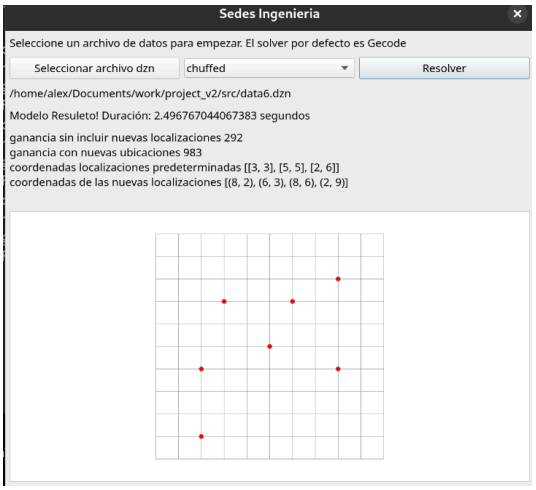
Entrada ejemplo 4

```
num_posiciones_existentes=3;

ciudades=[|3,3
|5,5
|2,6|];

tamano_matriz=10;

matriz_segmento_poblacion=[|26, 1, 1, 4, 5, 6, 7, 8, 9, 10
                           |1, 1, 1, 5, 6, 7, 8, 9, 10, 1
                           |1, 10, 5, 6, 7, 8, 9, 10, 1, 2
                           |4, 5, 6, 7, 8, 9, 10, 1, 2, 3
                           |5, 6, 7, 8, 9, 10, 1, 2, 30, 4
                           |6, 7, 8, 9, 10, 1, 2, 3, 4, 5
                           |7, 8, 9, 10, 1, 2, 30, 4, 5, 6
                           |8, 9, 10, 1, 2, 3, 4, 5, 6, 7
                           |9, 10, 1, 2, 3, 40, 5, 6, 7, 8
                           |10, 1, 2, 3, 4, 5, 6, 7, 8, 9|];

matriz_entorno_empresarial=[|9, 9, 5, 4, 3, 2, 1, 8, 9, 6
                           |6, 5, 4, 3, 2, 1, 70, 8, 9, 5
                           |9, 4, 3, 2, 1, 7, 6, 8, 5, 4
                           |4, 3, 2, 1, 7, 6, 5, 8, 4, 3
                           |3, 2, 1, 7, 6, 5, 4, 8, 3, 2
                           |2, 1, 7, 6, 5, 4, 3, 40, 2, 1
                           |9, 7, 6, 5, 4, 3, 2, 8, 1, 7
                           |8, 9, 7, 6, 5, 4, 3, 2, 1, 9
                           |9, 8, 6, 5, 4, 3, 2, 1, 7, 6
                           |6, 50, 4, 3, 20, 1, 9, 8, 7, 40|];

num_programas=4;
```

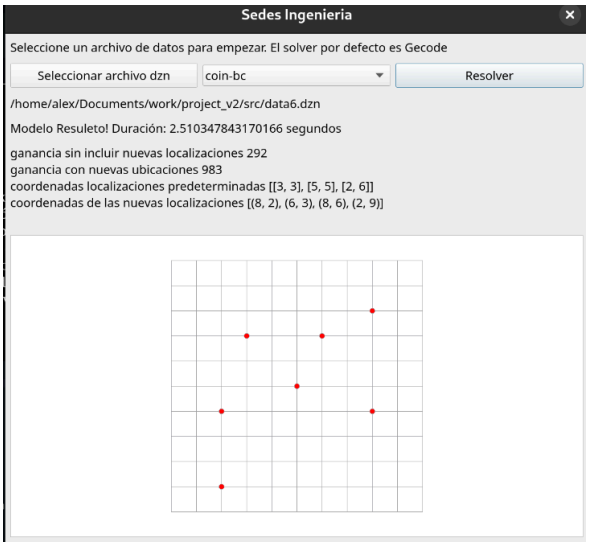
Salida ejemplo 4 - solver gecode



Salida ejemplo 4 - chuffed



Salida ejemplo 4 - coin-bc



Ejemplo 4. En este escenario, se evalúa el desempeño del algoritmo utilizando matrices poblacionales y empresariales de mayor dimensión (10x10). El objetivo es verificar si el algoritmo maneja eficientemente la complejidad adicional de los cálculos y mantiene su consistencia al ejecutarse en los tres solvers sin fallos.

IX. Conclusiones

El desarrollo de este programa se llevó a cabo en dos etapas principales:

1. Implementación del Modelo en MiniZinc: Se formuló el modelo de optimización en MiniZinc, permitiendo resolver el problema de maximización de ganancias y obtener resultados en forma de diccionarios con múltiples respuestas del output. Para alimentar el modelo, se utilizaron archivos **.dzn** que contienen las instancias del problema, facilitando la definición de parámetros específicos y casos de prueba.

2. Integración con PyQt5 y la API de MiniZinc: Para mejorar la visualización de los resultados, se desarrolló una interfaz gráfica utilizando PyQt5, complementada con la API de MiniZinc. Esta combinación facilitó la conexión entre el modelo de optimización y la interfaz de usuario, permitiendo una interacción eficiente y una visualización detallada de las soluciones.

La interfaz desarrollada permite visualizar detalladamente la solución en una representación matricial, mostrando tanto las ubicaciones predeterminadas como las nuevas, además de las ganancias asociadas a cada una. Además, se presentan las ganancias de las ubicaciones predeterminadas y las nuevas ubicaciones, proporcionando una visión clara del impacto de las decisiones tomadas.

El programa cumple con las restricciones establecidas y funciona correctamente, ofreciendo una herramienta robusta para la toma de decisiones en la ubicación óptima de programas adicionales.

X. Bibliografía

- *Modelamiento básico en MiniZinc:* 2.1. Basic Modelling in MiniZinc — The MiniZinc Handbook 2.8.7
- *Modelos más complejos:* https://docs.minizinc.dev/en/stable/modelling2.html
- *Python:* https://docs.minizinc.dev/en/stable/python.html
- *pyqt5:* https://pypi.org/project/PyQt5/
- *API de minizinc con python:* API — MiniZinc Python 0.9.0 documentation