INFORME PROYECTO 2 ANALISIS Y DISEÑO DE ALGORITMOS II - COMPLEJIDAD Y OPTIMIZACIÓN

Juan Miguel Posso Alvarado
2259610

Jhon Alejandro Martinez Murillo
2259565

I. RESUMEN

En este proyecto, desarrollamos un modelo en MiniZinc, aprovechando además su biblioteca disponible en Python, para proporcionar una solución al problema planteado. El objetivo fue encontrar la mejor solución dada dos matrices y unos puntos iniciales. Implementamos diversas restricciones, definiciones y funciones para garantizar que se cumplieran todas las condiciones necesarias. Se ha demostrado que el programa funciona correctamente, y ahora, nuestro profesor ya no tendrá que caminar preocupado por las sedes de la universidad, ¡sino que saltará de alegría!

II. INTRODUCCION

SOLUCION DEL PROBLEMA

La solución al problema planteado se desarrolló mediante la integración de diversos elementos, tales como la gestión de datos, el uso de restricciones lógicas y matemáticas, y la optimización del modelo a través de MiniZinc. En esta sección, se explica la lógica general que permite al programa resolver el problema de manera correcta y eficiente.

LÓGICA DE LA SOLUCIÓN

El programa está diseñado para resolver el problema de ubicación óptima de nuevos programas de ingeniería de sistemas bajo ciertas restricciones, asegurando una solución eficiente y válida.

A. Estructura General de la Solución

La solución se desarrolla en las siguientes etapas clave:

1) **Lectura y Procesamiento de Datos:** Los datos iniciales se leen desde un archivo de entrada, validando y estructurando la información en un formato adecuado para el modelo.
2) **Definición del Modelo:** El modelo en MiniZinc utiliza los datos procesados para definir las restricciones y la función objetivo.
3) **Ejecución del Modelo:** Se resuelve el modelo utilizando un solver adecuado, generando una solución que cumpla con las restricciones.
4) **Procesamiento de Resultados:** Los resultados obtenidos se interpretan y organizan para generar un archivo de salida legible.

B. Lógica de la Solución

El programa utiliza las siguientes lógicas principales para garantizar la solución correcta del problema:

- **Uso de Restricciones:** Las restricciones definidas aseguran que las nuevas ubicaciones cumplen con las condiciones necesarias, como evitar proximidad a ubicaciones existentes o garantizar un impacto poblacional y empresarial mínimos. Estas restricciones se traducen en lógica matemática en el modelo.
- **Maximización de la Ganancia:** La función objetivo busca maximizar la suma del impacto poblacional y empresarial, tanto para las ubicaciones existentes como para las nuevas, garantizando una solución óptima.
- **Separación de Datos y Modelo:** Al definir los datos en un archivo `.dzn`, se asegura que el modelo sea reutilizable y que los datos sean manejados de manera modular.

CONSIDERACIONES PARA LA SOLUCIÓN

Para llegar a la solución final, se tuvieron en cuenta los siguientes aspectos clave:

- **Modularidad:** Cada etapa del programa (lectura de datos, generación de datos en formato `.dzn`, resolución del modelo y procesamiento de resultados) está diseñada de manera independiente para facilitar el mantenimiento y la extensión del código.
- **Validación de Datos:** Se implementó un sistema de validación para garantizar que los datos de entrada cumplen con el formato esperado, evitando errores durante la ejecución del modelo.
- **Flexibilidad del Modelo:** El uso de MiniZinc permite que el modelo sea flexible y ajustable para diferentes configuraciones del problema, como el cambio en el número de ubicaciones existentes o el impacto requerido.

III. EXPLICACION DEL MODELO

Este informe explica en detalle el modelo implementado en MiniZinc para ubicar nuevos programas de ingeniería de sistemas. Se describen los parámetros, variables, restricciones y la función objetivo, utilizando notación matemática y conceptual. El objetivo principal es maximizar la ganancia total basada en datos poblacionales y empresariales.

PARÁMETROS Y VARIABLES

A. Parámetros

Los parámetros del modelo son:

- num_existing_locations: Número de ubicaciones existentes.
- existing_locations: Coordenadas (x, y) de las ubicaciones existentes.
- matrix_size: Dimensiones de las matrices de datos.
- population_matrix: Matriz que representa el impacto poblacional de cada ubicación.
- business_matrix: Matriz que representa el impacto empresarial de cada ubicación.
- num_new_programs: Número de nuevos programas a ubicar.

B. Variables

- new_locations$[i]$ $=$ (x, y): Coordenadas de las nuevas ubicaciones propuestas, donde $i \in \{1, \ldots, \text{num\_new\_programs}\}$.

FUNCIÓN OBJETIVO

La función objetivo busca maximizar la ganancia total, definida como:

$$\text{Ganancia Total} = \text{Ganancia Inicial} + \text{Ganancia Nueva} \quad (1)$$

C. Ganancia Inicial

La ganancia inicial se calcula sumando el impacto poblacional y empresarial de las ubicaciones existentes y sus celdas adyacentes:

$$\sum_{(x,y)\in \text{existing\_locations}} \sum_{i=x-1}^{x+1} \sum_{j=y-1}^{y+1} (\text{population\_matrix}[i,j] + \text{business\_matrix}[i,j]) \quad (2)$$

D. Ganancia de Nuevas Ubicaciones

De manera similar, la ganancia de las nuevas ubicaciones es:

$$\sum_{(x,y)\in \text{new\_locations}} \sum_{i=x-1}^{x+1} \sum_{j=y-1}^{y+1} (\text{population\_matrix}[i,j] + \text{business\_matrix}[i,j]) \quad (3)$$

RESTRICCIONES

E. Restricción 1: Proximidad a Ubicaciones Existentes

Los nuevos programas no pueden estar contiguos a las ubicaciones existentes:

$$\forall i, j \quad \text{distancia\_Manhattan}(\text{new\_locations}[i], \text{existing\_locations}[j]) > 1 \quad (4)$$

F. Restricción 2: Impacto Poblacional Mínimo

Cada nueva ubicación debe tener un impacto poblacional mínimo de 25:

$$\forall (x, y) \in \text{new\_locations}, \quad \sum_{i=x-1}^{x+1} \sum_{j=y-1}^{y+1} \text{population\_matrix}[i,j] \geq 25 \quad (5)$$

G. Restricción 3: Impacto Empresarial Mínimo

Cada nueva ubicación debe tener un impacto empresarial mínimo de 20:

$$\forall (x, y) \in \text{new\_locations}, \quad \sum_{i=x-1}^{x+1} \sum_{j=y-1}^{y+1} \text{business\_matrix}[i,j] \geq 20 \quad (6)$$

H. Restricción 4: Proximidad entre Nuevas Ubicaciones

Los nuevos programas no pueden estar contiguos entre sí:

$$\forall i \neq j, \quad \text{distancia\_Manhattan}(\text{new\_locations}[i], \text{new\_locations}[j]) > 1 \quad (7)$$

I. Restricción 5: Unicidad de Ubicaciones

Cada ubicación debe ser única:

$$\forall i \neq j, \quad \text{new\_locations}[i] \neq \text{new\_locations}[j] \quad (8)$$

EXPLICACION DEL PROGRAMA

En este informe se presenta el desarrollo e implementación de un sistema para automatizar la resolución de problemas utilizando MiniZinc. Se aborda el diseño de una interfaz gráfica, la gestión de archivos de entrada y salida, y la integración con el modelo MiniZinc para resolver el problema planteado.

INTERFAZ

La interfaz gráfica fue desarrollada en Python utilizando la librería `tkinter`. Su propósito principal es facilitar la interacción del usuario con el sistema. Las funciones clave de la interfaz incluyen:

J. Funcionamiento

La interfaz cuenta con:

- Un botón para cargar el archivo de entrada. Este archivo debe estar en formato `.txt` y contiene los datos necesarios para la ejecución del modelo.
- Un menú desplegable para seleccionar el solver a utilizar (`gecode`, `chuffed`, o `cp-sat`).
- Un botón para ejecutar el modelo. Este inicia el proceso de lectura de datos, generación del archivo `.dzn`, ejecución del modelo MiniZinc y generación del archivo de salida.

K. Guardado y carga de archivos

- El archivo de entrada se carga mediante un explorador de archivos integrado, verificando su formato y existencia antes de continuar.
- El archivo de salida se genera de manera interactiva, permitiendo al usuario elegir el nombre y la ubicación del archivo `.txt` resultante.

LECTURA DEL ARCHIVO DE ENTRADA

La función `leerArchivo` procesa el archivo de entrada para extraer y estructurar los datos. El archivo de entrada contiene:

- El número de ubicaciones de programas existentes.
- Las coordenadas de dichas ubicaciones.
- El tamaño de las matrices de población y entorno empresarial.
- Las matrices de datos de población y entorno empresarial.
- El número de nuevos programas a ubicar.

Esta función valida los datos y los organiza en un diccionario para facilitar su uso en las etapas posteriores.

CREACIÓN DEL ARCHIVO `.dzn`

La función `escribir_data_dzn` permite generar un archivo `.dzn`, que es el formato estándar para la definición de datos en MiniZinc. Esto ofrece las siguientes ventajas:

- Los datos se estructuran de manera que el modelo MiniZinc pueda acceder a ellos directamente.
- La separación entre datos y modelo mejora la modularidad y el rendimiento.

La función toma los datos procesados del archivo de entrada y los transforma en el formato `.dzn`, optimizando su compatibilidad con MiniZinc.

PROCESAMIENTO DE RESULTADOS

La función `procesar_resultado` maneja la salida generada por el modelo MiniZinc tras su ejecución. Los elementos clave procesados incluyen:

- **Ganancia inicial:** Calculada en base a las ubicaciones existentes.
- **Ubicaciones nuevas:** Coordenadas generadas por el modelo para los nuevos programas.
- **Ganancia total:** Suma de la ganancia inicial y la obtenida tras la adición de nuevas ubicaciones.

Esta función organiza los datos de salida para su uso en el archivo de salida.

GENERACIÓN DEL ARCHIVO DE SALIDA

La función `escribirArchivo` genera un archivo `salida.txt` que contiene los resultados del modelo en un formato legible. Los puntos clave incluyen:

- Escritura de la ganancia inicial y total.
- Listado de ubicaciones establecidas y nuevas, ordenadas por el primer valor de las coordenadas.
- Formato estructurado para facilitar su interpretación.

IV. PRUEBAS

En esta sección se presentan los resultados obtenidos al ejecutar el programa para los ejercicios de prueba. Se incluye la descripción de cada ejercicio, la evidencia de ejecución, y una discusión sobre los resultados obtenidos.

EJERCICIO DEL PDF

A. Descripción

Este ejercicio toma como entrada 3 ubicaciones iniciales, una matriz de población y entorno empresarial de tamaño 15x15, y busca ubicar 4 nuevos programas.

B. Evidencia de Ejecución

Entrada:

```
3
6 8
8 4
10 10
15
4 0 1 1 2 2 0 0 4 15 15 4 11 2 1
... (las matrices se encuentran en el pdf)
4
```

Salida:

```
440
2157
6 8
8 4
10 10
5 12
6 11
6 13
7 12
```

C. Discusión

La ganancia inicial de 440 refleja el impacto combinado de las ubicaciones existentes en la matriz. Con la inclusión de las nuevas ubicaciones, la ganancia total incrementó significativamente a 2157, demostrando la efectividad del modelo al seleccionar puntos óptimos que maximicen los beneficios poblacionales y empresariales.

EJEMPLO 1

D. Descripción

Este ejemplo considera 3 ubicaciones iniciales, una matriz de tamaño 6x6, y busca ubicar 3 nuevos programas.

E. Evidencia de Ejecución

Entrada:

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

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

Salida:

```
198
389
1 1
2 3
4 5
0 4
3 1
4 3
```

F. Discusión

La ganancia inicial de 198 aumentó a 389 al incluir las nuevas ubicaciones. Esto demuestra que el modelo logra identificar puntos estratégicos incluso en matrices más pequeñas, garantizando un uso eficiente del espacio disponible.

EJEMPLO 2

G. Descripción

En este caso, se tienen 2 ubicaciones iniciales, una matriz de 5x5, y se busca ubicar 2 nuevos programas.

H. Evidencia de Ejecución

Entrada:

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

Salida:

```
141
304
1 2
3 0
3 2
3 4
```

I. Discusión

La ganancia inicial de 141 aumentó a 304, demostrando que el modelo es capaz de maximizar el impacto empresarial y poblacional incluso con un menor número de ubicaciones iniciales y nuevas.

EJEMPLO 3

J. Descripción

Este ejemplo incluye 3 ubicaciones iniciales, una matriz de 8x8, y busca ubicar 3 nuevos programas.

K. Evidencia de Ejecución

Entrada:

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

Salida:

```
269
555
1 3
3 6
6 2
1 5
3 1
6 4
```

L. Discusión

La ganancia inicial de 269 aumentó a 555 tras la inclusión de las nuevas ubicaciones. Esto muestra que el modelo aprovecha matrices más grandes para generar beneficios significativos.

V. Conclusiones

Este proyecto integró de manera efectiva el uso de MiniZinc como herramienta de optimización con la implementación de un sistema en Python para automatizar el flujo de datos y la interpretación de resultados. La solución aborda un problema real de localización de nuevos programas de ingeniería, considerando restricciones específicas y maximizando beneficios poblacionales y empresariales.

A. Modelo

El modelo desarrollado en MiniZinc demuestra ser altamente modular y eficiente. Su estructura permite manejar restricciones complejas, como la proximidad entre ubicaciones y requisitos mínimos de impacto. La función objetivo garantiza la optimización global al buscar maximizar la ganancia total, integrando de manera equilibrada los datos poblacionales y empresariales.

B. Implementación

La implementación en Python incluyó:

- Lectura y validación de datos de entrada en un formato accesible y estructurado.
- Generación de archivos `.dzn` para facilitar la integración con MiniZinc.
- Procesamiento de resultados y generación de archivos de salida con los datos relevantes de manera ordenada.
- Una interfaz gráfica amigable para interactuar con el sistema, permitiendo configuraciones dinámicas y personalizables.

C. Resultados

Los resultados obtenidos muestran:

- Una ganancia inicial y total consistente con las restricciones y el modelo matemático.
- La identificación de ubicaciones öptimas para maximizar el impacto, incluso en escenarios con restricciones estrictas.
- Flexibilidad del sistema para adaptarse a diferentes configuraciones de entrada y escalabilidad en cuanto al tamaño de las matrices.

D. Conclusión Final

El proyecto cumple con los objetivos establecidos, proporcionando una solución robusta, eficiente y modular para problemas de optimización de ubicaciones. La combinación de MiniZinc con Python permitió integrar el modelado matemático con herramientas prácticas de manejo de datos, logrando un sistema completo y funcional. Este enfoque puede ser aplicado a problemas similares, demostrando su versatilidad y potencial para futuros desarrollos.

VI. Bibliografia

References

[1] ”MiniZinc Standard Library: Built-in Functions - array2d,” *MiniZinc Documentation*, [Online]. Available: https://docs.minizinc.dev/en/stable/lib-stdlib-builtins.html#array2d. [Accessed: Dec. 25, 2024].

[2] ”Modelling with MiniZinc,” *MiniZinc Documentation*, [Online]. Available: https://docs.minizinc.dev/en/stable/modelling2.html. [Accessed: Dec. 25, 2024].

[3] ”MiniZinc Python API: Solvers,” *MiniZinc*, [Online]. Available: https://python.minizinc.dev/en/latest/api.html#solvers. [Accessed: Dec. 25, 2024].

[4] ”Basic Usage of MiniZinc in Python: Finding All Optimal Solutions,” *MiniZinc*, [Online]. Available: https://python.minizinc.dev/en/latest/basic_usage.html#finding-all-optimal-solutions. [Accessed: Dec. 25, 2024].

[5] ”MiniZinc FlatZinc Specification,” *MiniZinc Documentation*, [Online]. Available: https://docs.minizinc.dev/en/stable/fzn-spec.html. [Accessed: Dec. 25, 2024].

[6] ”Getting Started with MiniZinc in Python,” *MiniZinc*, [Online]. Available: https://python.minizinc.dev/en/latest/getting_started.html#installation. [Accessed: Dec. 25, 2024].