Proyecto Ingeniería de sistemas al ∞ (Diciembre 26 de 2024)

Joan Sebastián Saavedra Perafán 202313025, Juan Manuel Hoyos Contreras 202380796, Sebastián Cifuentes Florez 202380764.

Palabras clave: Funciones, modelo, valores, variable.

ABSTRACT

This project optimizes the location of new systems engineering programs, maximizing population and business impact while adhering to geographic and socioeconomic constraints. A MiniZinc model is implemented to select locations within a matrix, ensuring minimum population and business environment thresholds, and preventing overlaps. The model uses auxiliary functions for efficient calculations and decision variables for new locations and their neighbors. It integrates a graphical interface developed with JavaScript, allowing users to upload data, select solvers, and export solutions. The implementation demonstrates efficiency, flexibility, and a user-friendly interface that simplifies its use.

I. RESUMEN

El proyecto optimiza la ubicación de nuevos programas de ingeniería de sistemas, maximizando el impacto poblacional y empresarial mientras respeta restricciones geográficas y socioeconómicas. Se implementa un modelo en MiniZinc que selecciona localizaciones en una matriz, asegurando umbrales mínimos de población y atractivo económico, evitando solapamientos. El modelo usa funciones auxiliares para cálculos eficientes y variables de decisión para las nuevas localizaciones y sus vecinos. Integra una interfaz gráfica desarrollada con JavaScript, que permite cargar datos, seleccionar solvers y exportar soluciones. La implementación demuestra eficiencia, flexibilidad y una interfaz amigable que simplifica su uso.

II. INTRODUCCIÓN

El proyecto aborda la optimización de la ubicación de nuevos programas de ingeniería de sistemas, considerando restricciones geográficas y socioeconómicas en un plano cartesiano bidimensional. Los objetivos clave fueron maximizar el impacto en términos de población y entorno empresarial, garantizando simultáneamente la distancia adecuada respecto a las localizaciones existentes.

III. PROCEDIMIENTOS REALIZADOS

A. *Modelo en Minizinc*

La idea principal es seleccionar nuevas localizaciones en una matriz de nxn para maximizar una ganancia total, considerando las restricciones y posiciones establecidas previamente.
A continuación, les mostraremos los parámetros, las notaciones matemáticas y los datos:

$$\text{total\_gain\_new} = \sum_{i=1}^{k} \left(\sum_{\substack{j=1 \\ \text{is\_in\_bounds}(x_i'+\Delta x_j, y_i'+\Delta y_j)}}^{9} P(x_i'+\Delta x_j, y_i'+\Delta y_j) + \sum_{\substack{j=1 \\ \text{is\_in\_bounds}(x_i'+\Delta x_j, y_i'+\Delta y_j)}}^{9} E(x_i'+\Delta x_j, y_i'+\Delta y_j) \right)$$

Fig. 1. Notación Matemática función objetivo.

$$\forall i \in \{1,\dots,k\}, \forall j \in \{1,\dots,9\}, \forall e \in \{1,\dots,v\}, \forall m \in \{1,\dots,9\}:$$
$$(\text{neighbor\_x}_{i,j} \neq x_e \vee \text{neighbor\_y}_{i,j} \neq y_e) \wedge (\text{neighbor\_x}_{i,j} \neq x_e + \Delta x_m \vee \text{neighbor\_y}_{i,j} \neq y_e + \Delta y_m)$$

Fig. 2. Restricción para asegurar que no haya posiciones contiguas con las localizaciones ya establecidas.

$$\forall i, l \in \{1,\dots,k\}, \forall j, m \in \{1,\dots,9\}, i \neq l:$$
$$\text{neighbor\_x}_{i,j} \neq \text{neighbor\_x}_{l,m} \vee \text{neighbor\_y}_{i,j} \neq \text{neighbor\_y}_{l,m}$$

Fig. 3. Restricción para asegurar que no haya solapamiento entre posiciones contiguas.

$$\forall i \in \{1,\dots,k\}:$$
$$\sum_{\substack{j=1 \\ \text{is\_in\_bounds}(x_i'+\Delta x_j, y_i'+\Delta y_j)}}^{9} P(x_i'+\Delta x_j, y_i'+\Delta y_j) \geq 25$$

Fig. 4. Restricción donde el segmento de población no puede ser menor que 25 en las nuevas localizaciones.

J. Hoyos, S. Saavedra, and S. Cifuentes were with Universidad del Valle, Tulua, Colombia. They are now with the Department of Systems Engineering.

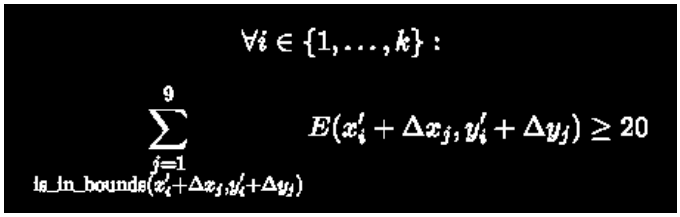

Fig. 5. Restricción donde el entorno empresarial no puede ser menor que 20 en las nuevas localizaciones.

1. **n**
Tamaño de la matriz (número de filas y columnas).
2. **k**
Número de nuevas localizaciones a seleccionar.
3. **v**
Número de localizaciones ya establecidas.

En nuestro código definimos dos matrices para representar datos de la población y el entorno empresarial:

1. **P**
Matriz de población que indica la cantidad de personas en cada celda.
2. **E**
Matriz del entorno empresarial que evalúa el atractivo económico de cada celda.

Para almacenar las coordenadas de las posiciones establecidas en términos de filas y columnas se tienen las siguientes coordenadas de localizaciones:

1. **x_coordinate**
2. **y_coordinate**

Estos dos vectores son los que contienen las posiciones de las localizaciones ya establecidas.

Se ajusta nuestro código con las variables de decisión en las que el modelo se ajusta para encontrar la mejor solución posible, en estas variables se encuentran las de la nueva localización:

1. **new_x**
2. **new_y**

Con estas dos variables se encuentran las coordenadas x y y de las nuevas localizaciones (k en total).

Tenemos otra variable de decisión que serán los Vecinos de las nuevas localizaciones:

1. **neighbor_x**
2. **neighbor_y**

Estas dos serán matrices que almacenan las coordenadas de las celdas vecinas (incluyendo diagonales) para cada nueva localización.

En este modelo, las funciones auxiliares desempeñan un papel fundamental para simplificar cálculos recurrentes y garantizar que las restricciones y objetivos se calculen de manera eficiente. A continuación, detallamos dos funciones auxiliares principales:

1. **Verificación de límites (is_in_bounds)**

Garantizar que las coordenadas estén dentro de los límites de la matriz de tamaño nxn. Esto es necesario porque muchas operaciones del modelo (como sumar vecinos) implican trabajar con desplazamientos.

```
function bool: is_in_bounds(int: x, int: y) =
  x >= 0 /\ x < n /\ y >= 0 /\ y < n;
```

x está entre 0 y $n-1$(límites de las filas).
y está entre 0 y $n-1$(límites de las columnas).
Devuelve true si las coordenadas están dentro de la matriz y false en caso contrario.

Variante para variables de decisión:

```
function var bool: is_in_bounds(var int: x, var int: y) =
  x >= 0 /\ x < n /\ y >= 0 /\ y < n;
```

Se evalúan límites en coordenadas de las nuevas localizaciones que aún no están determinadas.

2. **Suma con vecinos (sum_with_neighbors)**

Calcular la suma de los valores de una posición específica y todas sus celdas vecinas (incluyendo diagonales) en una matriz dada.

```
function int: sum_with_neighbors_old(int: x, int: y, array[int, int] of int: matrix) = (
  sum([matrix[x + dx[i], y + dy[i]] | i in 1..9 where is_in_bounds(x + dx[i], y + dy[i])])
);
```

Proceso:

Usa los desplazamientos para calcular las coordenadas de las celdas, comprueba si cada celda está dentro de los límites y suma los valores correspondientes de la matriz para las celdas válidas.

Variante para variables de decisión:

```
function var int: sum_with_neighbors(var int: x, var int: y, array[int, int] of int: matrix) = (
  sum([matrix[x + dx[i], y + dy[i]] | i in 1..9 where is_in_bounds(x + dx[i], y + dy[i])])
);
```

Usa variables de decisión para las coordenadas y permite calcular la suma para las nuevas localizaciones

En nuestro modelo tenemos las siguientes restricciones:

- Evitar solapamientos:
  - Se asegura de que ninguna nueva localización o sus vecinos coincidan con las posiciones ya establecidas o sus vecinos.
  - Se evita que las posiciones vecinas de distintas nuevas localizaciones se solapen.
- Umbrales mínimos:
  - Cada nueva localización debe tener una suma mínima de 25 en la matriz de población.
  - Cada nueva localización debe tener una suma mínima de 20 en la matriz del entorno empresarial.

La función objetivo contiene lo siguiente:

Maximizar la ganancia total:

Suma de los valores (P+E) en las nuevas localizaciones y sus vecinos.

Suma total combinada:

Combina las ganancias de las localizaciones establecidas y las nuevas.

Finalmente podemos decir que el modelo está diseñado para resolver este problema de optimización. A continuación, se detalla lo que el modelo genera en nuestra salida:

1. Ganancia total sin incluir la ganancia de nuevas localizaciones.
2. Ganancia total luego de agregar las nuevas localizaciones.
3. Coordenadas de las localizaciones ya establecidas
4. Coordenadas de las nuevas localizaciones ordenadas por el primer valor.

B. Integración con lenguaje de programación

El archivo contiene los siguientes elementos principales:

Modelo1.mzn: Archivo del modelo en MiniZinc.
Data1.dzn: Modelo de datos de prueba
index.html: Frontend para integrar MiniZinc con la aplicación.

1. Frontend (index.html):
   - Actúa como la interfaz para interactuar con MiniZinc.
   - Contiene el diseño básico con estilos en línea, un área de texto y botones para ejecutar acciones.
2. Backend (back-minizic):
   - Contiene varios archivos importantes:
     - index.js: Archivo principal de backend. Maneja la lógica de ejecución del modelo.
     - package.json y package-lock.json: Archivos relacionados con dependencias para un entorno Node.js.
     - temp.mzn: Un archivo temporal, se utilizó para generar o guardar modelos dinámicamente.

C. Interfaz gráfica

Fig. 1. Interfaz gráfica Minizinc.

Componentes de Entrada

1. **input type="file" (Carga de datos)**

Este componente permite al usuario cargar un archivo de texto desde su dispositivo. El archivo que se carga contiene los datos que serán procesados por el modelo MiniZinc.
A continuación, se muestra la lógica:

- Evento change: Cuando el usuario selecciona un archivo, se activa el evento change. El evento captura el archivo que el usuario ha elegido.
- Uso del FileReader: El archivo cargado es leído utilizando el objeto FileReader de JavaScript, que lee el contenido del archivo como texto.
- Habilitación del botón "Ejecutar": Una vez que el archivo ha sido cargado y leído correctamente, el contenido del archivo se almacena en la variable entradaUsuario, y el botón "Ejecutar" es habilitado (solveButton.disabled = false), lo que permite al usuario hacer clic en él para ejecutar el modelo.

2. **<select> (Selección de solver)**

Este componente permite al usuario seleccionar el solver que desea utilizar para resolver el modelo MiniZinc. El solver es el motor de cálculo que procesa los datos según el modelo proporcionado (en este caso, "Chuffed", "Gecode", o "Gurobi").
A continuación, se muestra la lógica:
Captura del valor seleccionado: Cuando el usuario selecciona una opción del desplegable, el valor seleccionado se captura mediante solverSelect.value. Este valor se utilizará más adelante para configurar el modelo y pasar el solver correcto cuando se ejecute.

Componentes de Ejecución

1. **Botón "Ejecutar" solveButton**

Este botón inicia la resolución del modelo MiniZinc cuando el usuario ya ha cargado el archivo con los datos y seleccionado un solver.

Componentes de Salida

2. **Botón "Exportar Soluciones" exportButton**

   Este botón permite al usuario exportar las soluciones obtenidas del modelo MiniZinc a un archivo de texto que puede ser descargado a su dispositivo.

Componentes Estáticos

1. <h1> (Título principal)
2. <h2> (Encabezado secundario)

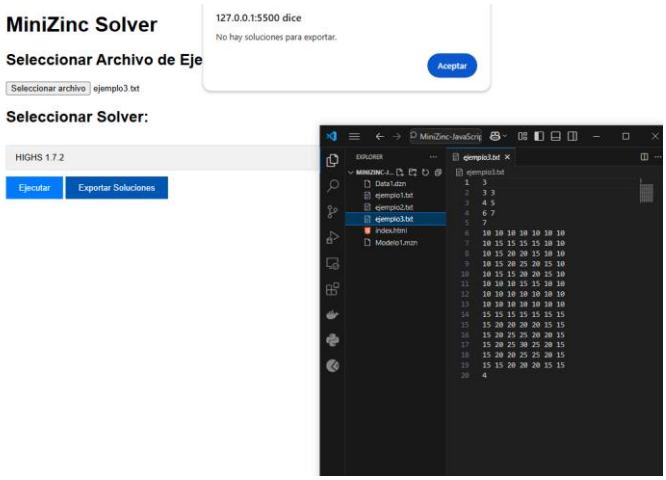

Fig. 9. Ejecución con el ejemplo3.txt y el solver Highs 1.7.2.

IV. DISCUSIÓN DE RESULTADOS

A. Sección de pruebas

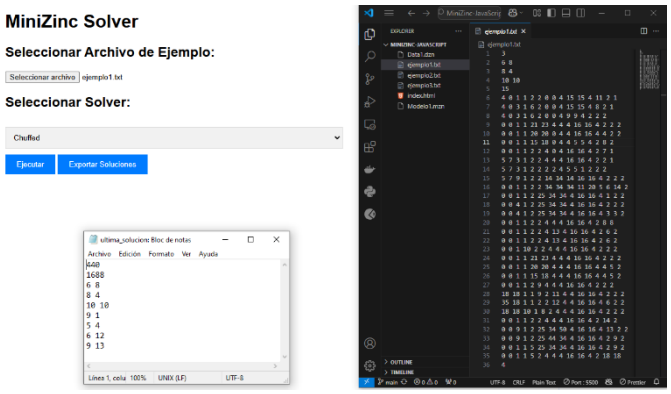

Fig. 7. Ejecución con el ejemplo1.txt y el solver shuffed.

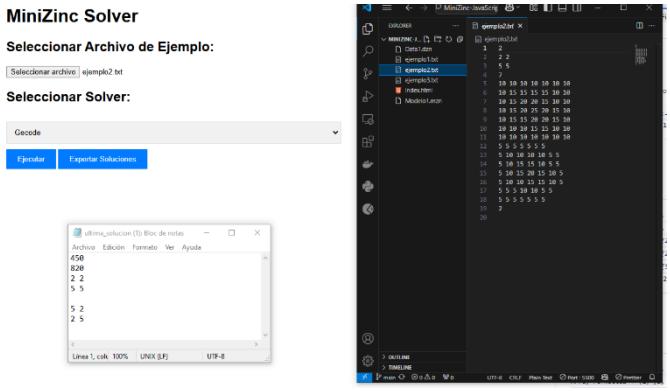

Fig. 8. Ejecución con el ejemplo2.txt y el solver Gecode.

V. CONCLUSIONES

El modelo MiniZinc, está diseñado para optimizar la selección de nuevas ubicaciones basándose en factores como la población y el entorno empresarial, demuestra una notable eficiencia y modularidad. La separación entre datos y modelo permite una flexibilidad significativa y facilita la reutilización con distintos conjuntos de datos. La integración técnica con JavaScript proporciona una interfaz amigable que simplifica la interacción del usuario, mientras que el backend automatiza la ejecución y procesamiento de los resultados.

VI. REFERENCES

[1] Gurobi Optimization, LLC, Gurobi Optimizer Reference Manual, 2024. [Online]. Available: https://www.gurobi.com/documentation/

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 4th ed. Cambridge, MA, USA: MIT Press, 2022.

[3] J. N. Hooker, Integrated Methods for Optimization, 2nd ed. Boston, MA, USA: Springer, 2007.

[4] "Complejidad y Optimización 2021-II," *YouTube Playlist*, YouTube, https://www.youtube.com/playlist?list=PLMk3Ka15Yzy4JRKqNPxmj30RSNe3Ife2I.

[5] S. Arora and B. Barak, *Computational Complexity: A Modern Approach* (Draft), Princeton University, Jan. 2007. [Online]. Available: https://theory.cs.princeton.edu/complexity/book.pdf.