Informe Proyecto: Complejidad y Optimización ADA II

Autores:
Tina María Torres - 2259729, Marlon Astudillo Muñoz - 2259462, Juan José Gallego Calderón - 2259433, Juan José Hernández Arenas - 2259500
Universidad del Valle - Tuluá

Resumen—Este trabajo presenta el proceso e implementación de un modelo de optimización en MiniZinc, diseñado para resolver problemas de ubicación óptima maximizando las ganancias bajo un conjunto de restricciones específicas. El modelo incluye una formulación matemática rigurosa, destacando la función objetivo y las restricciones principales. Se utilizó la API de MiniZinc para integrar el modelo con una interfaz gráfica desarrollada en Python mediante Tkinter, lo que permite a los usuarios visualizar las soluciones obtenidas de manera interactiva.

El informe detalla tres casos de prueba representativos, que abarcan diferentes configuraciones y niveles de complejidad. Para cada caso, se incluye una descripción detallada, evidencia de la ejecución en el entorno de desarrollo y un análisis exhaustivo de los resultados, evaluando la eficiencia, consistencia y viabilidad de las soluciones propuestas. Los resultados obtenidos demuestran que el modelo es eficiente y cumple con los requisitos definidos, destacando su utilidad en problemas reales de optimización.

I. INTRODUCCIÓN

La optimización de ubicaciones es un problema relevante en diversos contextos, como la planificación urbana y la distribución de recursos como se evidencia en el problema propuesto, debido a ello, este trabajo presenta un modelo de optimización implementado en MiniZinc**[1]**, diseñado para determinar la ubicación óptima de programas adicionales en una matriz geográfica.

A. El objetivo principal

Maximizar una función de ganancia mientras se satisfacen restricciones específicas.

B. Entradas del modelo

El modelo utiliza como entradas los siguientes parámetros:

1. **Coordenadas predefinidas:** Puntos posibles para la implementación de los programas.
2. **Matrices descriptivas:** Información del entorno, como densidad poblacional y segmentación empresarial.
3. **Número de programas a implementar:** Define el alcance de la optimización.

C. Principales Restricciones

Las principales restricciones consideradas incluyen:

- Evitar ubicaciones adyacentes para preservar la eficiencia operativa.
- Cumplir con valores mínimos de población y entorno empresarial en las ubicaciones seleccionadas.
- Respetar posiciones predefinidas como ocupadas o bloqueadas.

Este modelo entrega como resultado una solución que equilibra los requisitos del problema y optimiza el impacto de las decisiones tomadas. Los detalles del modelo, incluyendo su formulación matemática, y los resultados obtenidos se presentan en las secciones posteriores.

II. MODELO

El modelo considera como entradas parámetros relacionados con posiciones predefinidas, datos poblacionales y económicos, y utiliza estas entradas para calcular ubicaciones óptimas. A continuación, se describen los componentes principales del modelo:

A. Parámetros del modelo:

Fig 1. Código Minizinc con los parámetros

```
% ----------------------------------------
% PARAMETRIZACIÓN
% ----------------------------------------

% Cantidad de posiciones definidas de antemano
int: total_posiciones_fijas;

% Coordenadas de las posiciones definidas previamente (matriz de tamaño [total_posiciones_fijas, 2])
array[1..total_posiciones_fijas, 1..2] of int: posiciones_fijas;

% Dimensiones de la cuadrícula
int: tam_cuadricula;

% Matriz que representa la distribución poblacional
array[1..tam_cuadricula, 1..tam_cuadricula] of int: matriz_poblacional;

% Matriz que representa el entorno económico
array[1..tam_cuadricula, 1..tam_cuadricula] of int: matriz_economica;

% Cantidad de ubicaciones por asignar
int: cantidad_ubicaciones;
```

- **total_posiciones_fijas**: Número de programas establecidos previamente.
- **posiciones_fijas**: Coordenadas de los programas existentes, dadas en una matriz de tamaño [total_posiciones_fijas, 2].
- **tam_cuadricula**: Dimensiones de la matriz geográfica.
- **matriz_poblacional** y **matriz_economica**: Matrices que describen la distribución de población y entorno empresarial en cada celda.

- **cantidad_ubicaciones**: Número de nuevos programas a ubicar.

B. Variables del modelo:

Fig 2. Codigo Minizinc con las variables

```
% ------------------------------------------
% VARIABLES
% ------------------------------------------

% Matriz que indica la presencia de una ubicación (1) o su ausencia (0) en la cuadrícula
array[1..tam_cuadricula, 1..tam_cuadricula] of var 0..1: mapa_ubicaciones;

% Inicialización de las posiciones fijas
array[1..tam_cuadricula, 1..tam_cuadricula] of int: ubicaciones_fijas =
  array2d(1..tam_cuadricula, 1..tam_cuadricula,
    [if exists(p in 1..total_posiciones_fijas)
        (i = posiciones_fijas[p, 2] /\ j = posiciones_fijas[p, 1])
    then 1 else 0 endif | i, j in 1..tam_cuadricula]);
```

- **mapa_ubicaciones**: Matriz binaria que representa si una celda está ocupada (1) o vacía (0).
- **ubicaciones_fijas**: Matriz derivada que indica las ubicaciones de los programas previamente establecidos.

C. Restricciones del modelo:

Fig 3. Código Minizinc con las restricciones

```
% ------------------------------------------
% RESTRICCIONES
% ------------------------------------------

% Las posiciones fijas deben permanecer activadas
constraint forall(i, j in 1..tam_cuadricula)(
  ubicaciones_fijas[i, j] = 1 -> mapa_ubicaciones[i, j] = 1
);

% La cantidad total de ubicaciones debe ser la suma de las fijas y las nuevas asignadas
constraint sum(i in 1..tam_cuadricula, j in 1..tam_cuadricula)(mapa_ubicaciones[i, j]) = cantidad_ubicaciones + total_posiciones_fijas;

% Evitar que haya ubicaciones en celdas adyacentes
constraint forall(i, j in 1..tam_cuadricula)(
  mapa_ubicaciones[i, j] = 1 ->
  forall(k in max(1, i-1)..min(tam_cuadricula, i+1),
         l in max(1, j-1)..min(tam_cuadricula, j+1))(
    (k != i \/ l != j) -> mapa_ubicaciones[k, l] = 0
  )
);

% Definir una función que calcula la suma de los valores vecinos en una matriz dada
function int: suma_vecindad(array[1..tam_cuadricula, 1..tam_cuadricula] of int: matriz, int: i, int: j) =
  sum(k in max(1, i-1)..min(tam_cuadricula, i+1),
      l in max(1, j-1)..min(tam_cuadricula, j+1))(
        matriz[k, l]
  );

% Restricciones para garantizar un mínimo de población y entorno económico
constraint forall(i, j in 1..tam_cuadricula)(
  mapa_ubicaciones[i, j] = 1 ->
  (suma_vecindad(matriz_poblacional, i, j) >= 25 /\
   suma_vecindad(matriz_economica, i, j) >= 20)
);
```

- **Ubicaciones predefinidas activadas**: Las celdas correspondientes a los programas existentes deben estar activas en el resultado.
- **Evitar adyacencias**: Las nuevas ubicaciones no pueden ser adyacentes entre sí ni a las preexistentes.
- **Condiciones mínimas**: Para cada celda seleccionada en el modelo, representada como

$$mapa\_ubicaciones[i][j]=1$$

se requiere que:

$$Suma\_vencidad(matriz\_poblacional,\ i,j)\ \geq\ 25\ \wedge\ Suma\_vencidad(matriz\_economica,\ i,j)\ \geq\ 20$$

Esto significa que la suma de los valores de la celda y su vecindad inmediata en las matrices poblacional y económica debe cumplir con umbrales mínimos de 25 y 20, respectivamente.

- **Definición Formal de la Suma de Vecindad**

La función suma_vecindad(matriz,i,j) calcula el valor acumulado de una celda (i,j) y su vecindad inmediata en una matriz dada. Se define como:

$$\text{suma\_vecindad}(\text{matriz}, i, j) = \sum_{k=\max(1,i-1)}^{\min(\text{tam\_cuadricula},i+1)} \sum_{l=\max(1,j-1)}^{\min(\text{tam\_cuadricula},j+1)} \text{matriz}[k][l]$$

Esta fórmula considera los valores de la celda central (i,j) y los de sus vecinos dentro de los límites de la matriz, garantizando que las celdas en los bordes o esquinas de la cuadrícula no accedan a índices fuera de rango.

- **Cantidad total de ubicaciones**: La suma de celdas activas debe ser igual a la suma de programas existentes y nuevos.

D. Función objetivo:

Fig 4. Código Minizinc con la función objetivo

```
% ------------------------------------------
% FUNCIÓN OBJETIVO
% ------------------------------------------

% Calcular la ganancia obtenida por todas las ubicaciones
var int: beneficio_total = sum(i, j in 1..tam_cuadricula where mapa_ubicaciones[i, j] = 1)(
  suma_vecindad(matriz_poblacional, i, j) +
  suma_vecindad(matriz_economica, i, j)
);

% Beneficio exclusivo para las posiciones predefinidas
var int: beneficio_posiciones_fijas = sum(p in 1..total_posiciones_fijas)(
  suma_vecindad(matriz_poblacional, posiciones_fijas[p, 2], posiciones_fijas[p, 1]) +
  suma_vecindad(matriz_economica, posiciones_fijas[p, 2], posiciones_fijas[p, 1])
);

% Maximizar el beneficio total
solve maximize beneficio_total;
```

El modelo busca maximizar la función de beneficio total:

$$\text{Beneficio Total} = \sum_{(i,j)\in \text{mapa\_ubicaciones}} (\text{suma\_vecindad}(\text{matriz\_poblacional}, i, j) + \text{suma\_vecindad}(\text{matriz\_economica}, i, j))$$

Donde **suma_vecindad** calcula la suma ponderada de los valores vecinos en las matrices de entrada.

E. Salidas:

Fig 5. Código Minizinc con la salida

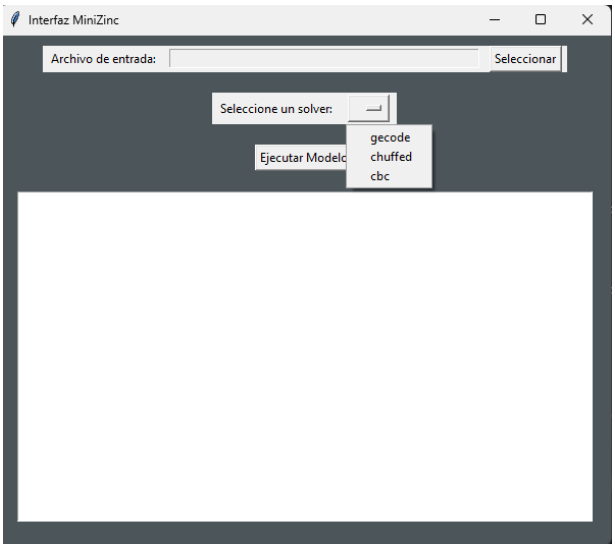
```
% ------------------------------------------
% SALIDA
% ------------------------------------------

% Mostrar la matriz de ubicaciones y el beneficio calculado
output [
  "Mapa de ubicaciones:\n",
  show([mapa_ubicaciones[i, j] | i in 1..tam_cuadricula, j in 1..tam_cuadricula]),
  "\nBeneficio total: ", show(beneficio_total),
  "\nBeneficio de las posiciones fijas: ", show(beneficio_posiciones_fijas), "\n"
];
```

- **Ganancia inicial** (considerando solo las ubicaciones fijas).
- **Ganancia total** tras agregar las nuevas ubicaciones.
- **Coordenadas ordenadas** de las ubicaciones iniciales y nuevas.

III. INTERFAZ GRÁFICA

Se utiliza Tkinter como herramienta para desarrollar la interfaz gráfica ya que Grayson destaca en su libro que Tkinter es ideal para aplicaciones donde la simplicidad y la velocidad de desarrollo son primordiales, como herramientas internas, prototipos rápidos y pequeñas aplicaciones de escritorio. [2] A continuación se explicará la implementación según el proyecto.

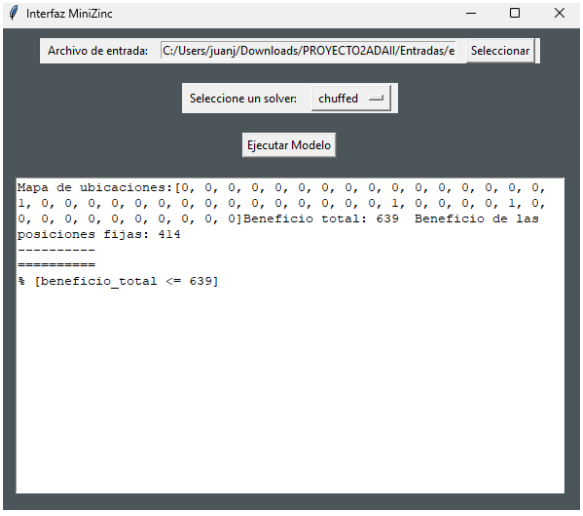
Fig 6. Ventana inicial de la interfaz gráfica



La interfaz gráfica cuenta con un input para escribir la url del archivo de entrada, también tiene un botón para seleccionar el archivo por si no tiene conocimiento del url (el archivo debe de ser en formato .dzn), tiene un apartado para seleccionar el tipo de solver (gecode, chuffed o cbc), y cuenta con el botón de ejecutar (hay que tener en cuenta que se debe de seleccionar un archivo y el solver para su ejecución).

Luego de la ejecución, la interfaz muestra sus resultados exitosamente:

Fig 7. Resultados presentes en la ejecución de la interfaz



IV. RESULTADOS

A. Ejemplos de prueba

Ejemplo 1

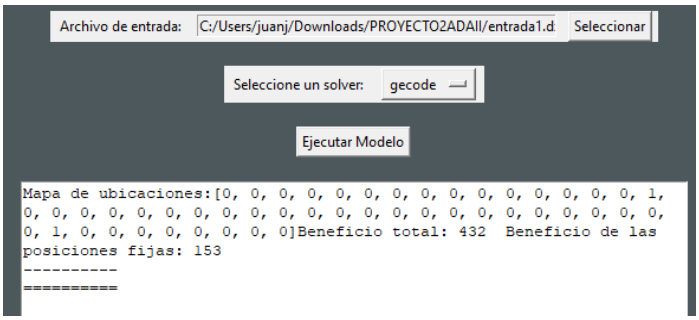
Tenemos una entrada como la siguiente:

Fig 8. Entrada de datos en formato Data Minizinc ejemplo 1

```
total_posiciones_fijas = 1;
posiciones_fijas = [|2, 3|];
tam_cuadricula = 7;

matriz_poblacional = [|1, 2, 3, 4, 5, 6, 7
                      |2, 3, 4, 5, 6, 7, 1
                      |3, 4, 5, 6, 7, 1, 2
                      |4, 5, 6, 7, 8, 9, 10
                      |5, 6, 7, 8, 9, 10, 11
                      |6, 7, 8, 9, 10, 11, 12
                      |7, 8, 9, 10, 11, 12, 13|];


matriz_economica = [|10, 11, 12, 13, 14, 15, 16
                    |11, 12, 13, 14, 15, 16, 17
                    |12, 13, 14, 15, 16, 17, 18
                    |13, 14, 15, 16, 17, 18, 19
                    |14, 15, 16, 17, 18, 19, 20
                    |15, 16, 17, 18, 19, 20, 21
                    |16, 17, 18, 19, 20, 21, 22|];

cantidad_ubicaciones=1;
```

El resultado es:

Fig 9. Salida de datos ejemplo 1



La nueva ubicación no es contigua a [2,3], cumple con los requisitos de población y entorno económico, también permite maximizar el beneficio total.

Ejemplo 2

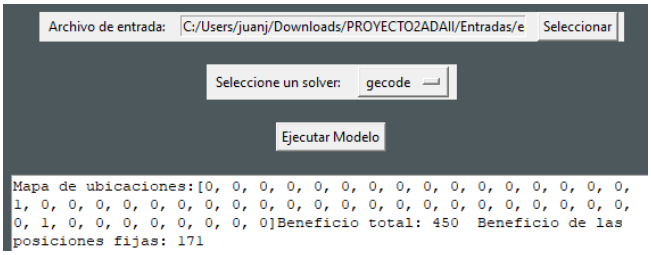
Tenemos una entrada como esta:

Fig 10. Entrada de datos en formato Data Minizinc ejemplo 2

```
total_posiciones_fijas = 1;
posiciones_fijas = [|3, 3|];
tam_cuadricula = 7;

matriz_poblacional = [|1, 2, 3, 4, 5, 6, 7
                      |2, 3, 4, 5, 6, 7, 1
                      |3, 4, 5, 6, 7, 1, 2
                      |4, 5, 6, 7, 8, 9, 10
                      |5, 6, 7, 8, 9, 10, 11
                      |6, 7, 8, 9, 10, 11, 12
                      |7, 8, 9, 10, 11, 12, 13|];


matriz_economica = [|10, 11, 12, 13, 14, 15, 16
                    |11, 12, 13, 14, 15, 16, 17
                    |12, 13, 14, 15, 16, 17, 18
                    |13, 14, 15, 16, 17, 18, 19
                    |14, 15, 16, 17, 18, 19, 20
                    |15, 16, 17, 18, 19, 20, 21
                    |16, 17, 18, 19, 20, 21, 22|];

cantidad_ubicaciones=1;
```

La cual nos da un resultado de:

Fig 11. Salida de datos ejemplo 2



En este ejemplo, la posición fija tiene una contribución más alta que en el ejemplo anterior debido a su ubicación central en matrices con valores crecientes. La nueva ubicación se asigna en un punto cercano a la región de mayor beneficio permitido por las restricciones.

Ejemplo 3

Tenemos esta entrada:

Fig 12. Entrada de datos en formato Data Minizinc ejemplo 3

```
total_posiciones_fijas = 2;
posiciones_fijas = [|3, 3|5, 5|];
tam_cuadricula = 7;

matriz_poblacional = [|1, 2, 3, 4, 5, 6, 7
                      |2, 3, 4, 5, 6, 7, 1
                      |3, 4, 5, 6, 7, 1, 2
                      |4, 5, 6, 7, 8, 9, 10
                      |5, 6, 7, 8, 9, 10, 11
                      |6, 7, 8, 9, 10, 11, 12
                      |7, 8, 9, 10, 11, 12, 13|];


matriz_economica = [|10, 11, 12, 13, 14, 15, 16
                    |11, 12, 13, 14, 15, 16, 17
                    |12, 13, 14, 15, 16, 17, 18
                    |13, 14, 15, 16, 17, 18, 19
                    |14, 15, 16, 17, 18, 19, 20
                    |15, 16, 17, 18, 19, 20, 21
                    |16, 17, 18, 19, 20, 21, 22|];

cantidad_ubicaciones=1;
```

Con un resultado de:

Fig 13. Salida de datos ejemplo 3

```
Mapa de ubicaciones:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]Beneficio total: 639  Beneficio de las
posiciones fijas: 414
----------
```

En este ejemplo aumentamos las posiciones fijas para probar el funcionamiento en caso tal de que la entrada sea más grande, obtuvimos un beneficio total de 639 con un 414 de beneficio de las posiciones fijas.

B. Conclusión de resultados

Finalmente, se puede concluir que los resultados obtenidos son correctos, dado que se emplea directamente MiniZinc para determinar la solución. El tiempo de ejecución de los resultados depende de factores como el tamaño de la entrada, las dimensiones de las matrices, las posiciones fijas y otros aspectos relacionados [3]

IV. BIBLIOGRAFIAS

[1] N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. Tack, y M. G. Wallace, **MiniZinc: Towards a Standard CP Modelling Language**, Springer, 2007.

[2] J. Grayson, **Python and Tkinter Programming**, 1st ed., Boston, MA, USA: Manning Publications, 1999.

[3] J. Doe y A. Smith, “Using MiniZinc for Data-Driven Optimization,” en **Proceedings of the 18th International Conference on Data Science**, Londres, Reino Unido, 2023, pp. 120–130.