INFORME PROYECTO II - ADA II

Víctor Manuel Hernández Ortiz *2259520*
Nicolás Mauricio Rojas Mendoza *2259460*
Esteban Alexander Revelo Salazar *2067507*

I. INTRODUCCION

En el siguiente trabajo se quiere dar solución al problema de ingenieros al infinito, en este tendremos que plantear como añadir nuevas escuelas de ingeniería basándonos en unas restricciones sobre el segmento de la población y el entorno empresarial, esto lo podremos hacer mediante la implementación de un modelo minizinc el cual evaluara y maximizara las nuevos programas sacando la mayor cantidad de ganancia.

II. MODELO MINIZINC

El modelo que se nos plantea, debe recorrer dos matrices, una del segmento y otra del entorno, estas son de tamaño n y ademas pueden haber programas ya instalados en otras ubicaciones, ademas debemos especificar la cantidad de ubicaciones nuevas que queremos, ante todo esto se plantean las siguientes restricciones.

A. *Restriccion de segmento de poblacion.*

La suma de la posición y los valores adyacentes en el punto (x, y), en la matriz de segmento tiene que ser ≥ 25. Esto lo podemos ver implementado en el modelo minizinc de la siguiente manera:

```
constraint
forall(new_loc in 0..num_nuevos-1) (
    let {
        var set of int: x_rango = max(0,
             ubicaciones_nuevas[new_loc, 1]-1)..min(n-1,
             ubicaciones_nuevas[new_loc, 1]+1),
        var set of int: y_rango = max(0,
             ubicaciones_nuevas[new_loc, 0]-1)..min(n-1,
             ubicaciones_nuevas[new_loc, 0]+1)
    } in
        sum([segmento_poblacion[i, j] | i in x_rango, j
             in y_rango]) >= 25
);
```

Lo que podemos observar es que un Constrain representa las restricciones, y al decirle al modelo que es for all, lo que estamos haciendo es aplicar la restriccion se aplique a cada elemento de un rango o conjunto. Dentro del let se definen dos variables que son los rangos de coordenadas en las dimensiones **x** y **y** dentro de los limites que es la dimension de la matriz, max y min aseguran que los rangos no salgan fuera de los límites de la matriz: max(0, ...) asegura que no se pase del borde inferior (0). min(n-1, ...) asegura que no se pase del borde superior (n-1).

luego tenemos

```
sum([segmento_poblacion[i, j] | i in x_rango, j in
     y_rango])
```

La sumatoria evalúa la población en los segmentos adyacentes a la ubicación nueva,

`[...| i in x_rango, j in y_rango]`: Es una comprensión de listas que genera todos los valores `segmento_poblacion[i, j]` para cada combinación de índices `(i, j)` dentro de los rangos `x_rango` y `y_rango`. La sumatoria, por tanto, calcula el total de población en el vecindario de x-rango y y-rango alrededor de la ubicación nueva.

Y finalmente, la restricción se aplica a la sumatoria, asegurando que sea mayor o igual a 25.

```
sum([segmento_poblacion[i, j] | i in x_rango, j in y_rango
    ]) >= 25
```

B. *Restriccion de entorno empresarial.*

La suma de la posición y los valores adyacentes en el punto (x, y), en la matriz de entorno tiene que ser ≥ 20. Esto lo podemos ver implementado en el modelo minizinc de la siguiente manera:

```
constraint
forall(new_loc in 0..num_nuevos-1) (
    let {
        var set of int: x_rango = max(0,
             ubicaciones_nuevas[new_loc, 1]-1)..min(n-1,
             ubicaciones_nuevas[new_loc, 1]+1),
        var set of int: y_rango = max(0,
             ubicaciones_nuevas[new_loc, 0]-1)..min(n-1,
             ubicaciones_nuevas[new_loc, 0]+1)
    } in
        sum([entorno_empresarial[i, j] | i in x_rango, j
             in y_rango]) >= 20
);
```

En esta parte aplicamos la misma logica de la restriccion anterior, pero ahora con la matriz de entorno empresarial, y la restriccion es que la suma de la posicion y sus adyacentes en la matriz de entorno tiene que ser ≥ 20.

C. *Restriccion de ubicaciones.*

Otra restriccion que se nos plantea es la de que las ubicaionces nuevas no pueden ser contiguas a las ya existentes, esto lo podemos ver implementado en el modelo minizinc de la siguiente manera:

```
constraint
  forall(i in 0..num_nuevos-1, j in 0..num_nuevos-1 where
       i != j) (
      abs(ubicaciones_nuevas[i, 0] - ubicaciones_nuevas[j
          , 0]) > 1 \/
      abs(ubicaciones_nuevas[i, 1] - ubicaciones_nuevas[j
          , 1]) > 1 \/
      (abs(ubicaciones_nuevas[i, 0] - ubicaciones_nuevas[
          j, 0]) + abs(ubicaciones_nuevas[i, 1] -
          ubicaciones_nuevas[j, 1]) > 1)
  );
```

La restricción se aplica a cada par de nuevas ubicaciones i y j donde i es diferente de j y la condición

```
abs(ubicaciones_nuevas[i, 0] - ubicaciones_nuevas[j, 0])
    > 1
```

asegura que la diferencia en la coordenada x entre dos ubicaciones nuevas es mayor que 1. Las condiciones

```
(abs(ubicaciones_nuevas[i, 0] - ubicaciones_nuevas[j, 0])
     + abs(ubicaciones_nuevas[i, 1] - ubicaciones_nuevas[
    j, 1]) > 1)
```

asegura que la suma de las diferencias absolutas en las coordenadas x y y entre dos ubicaciones nuevas es mayor que 1, lo que previene que estén adyacentes en cualquier dirección.

D. Restriccion de ubicaciones ya existentes.

Ademas no se pueden instalar programas en las mismas ubicaciones que ya existen.

```
constraint
  forall(new_loc in 0..num_nuevos-1, existing_loc in 0..
       num_existentes-1) (
      ubicaciones_nuevas[new_loc, 0] !=
          ubicaciones_existentes[existing_loc, 0] \/
      ubicaciones_nuevas[new_loc, 1] !=
          ubicaciones_existentes[existing_loc, 1]
  );
```

El propósito de esta restricción es asegurarse de que ninguna de las nuevas ubicaciones coincida exactamente con cualquiera de las ubicaciones existentes. En otras palabras, cada nueva ubicación debe tener al menos una coordenada (x o y) diferente de todas las ubicaciones existentes.

E. Funcion Objetivo.

La función objetivo es maximizar la ganancia de los programas instalados en las nuevas ubicaciones, cumpliendo con las restricciones anteriores, esto lo podemos ver implementado en el modelo minizinc de la siguiente manera:

```
  int: ganancia_sin_nuevas = sum([let {
    set of int: x_rango = max(0, ubicaciones_existentes[i,
         1]-1)..min(n-1, ubicaciones_existentes[i, 1]+1),
    set of int: y_rango = max(0, ubicaciones_existentes[i,
         0]-1)..min(n-1, ubicaciones_existentes[i, 0]+1)
} in
    sum([segmento_poblacion[x, y] + entorno_empresarial[x,
         y] | x in x_rango, y in y_rango])| i in 0..
         num_existentes-1]);

var int: ganancia_total = sum([let {
    var set of int: x_rango = max(0, ubicaciones_nuevas[i,
         1]-1)..min(n-1, ubicaciones_nuevas[i, 1]+1),
    var set of int: y_rango = max(0, ubicaciones_nuevas[i,
         0]-1)..min(n-1, ubicaciones_nuevas[i, 0]+1)
} in
    sum([segmento_poblacion[x, y] + entorno_empresarial[x,
         y] | x in x_rango, y in y_rango])| i in 0..
         num_nuevos-1]) + sum([let {
    set of int: x_rango = max(0, ubicaciones_existentes[i,
         1]-1)..min(n-1, ubicaciones_existentes[i, 1]+1),
    set of int: y_rango = max(0, ubicaciones_existentes[i,
         0]-1)..min(n-1, ubicaciones_existentes[i, 0]+1)
} in
    sum([segmento_poblacion[x, y] + entorno_empresarial[x,
         y] | x in x_rango, y in y_rango])| i in 0..
         num_existentes-1]);

solve maximize ganancia_total;
```

- ganancia sin nuevas: Calcula la ganancia sin considerar las nuevas ubicaciones, solo utilizando las ubicaciones existentes, para cada ubicación existente, se calcula la suma de los valores del segmento de población y del entorno empresarial en el rango adyacente (inmediato) a esa ubicación.
  `x_rango` y `y_rango`. definen el rango de filas y columnas alrededor de cada ubicación existente, la suma total de estas ganancias se almacena en la variable `ganancia_sin_nuevas`.
- ganancia total: Calcula la ganancia total considerando tanto las nuevas ubicaciones como las ubicaciones existentes, para cada nueva ubicación, se calcula la suma de los valores del segmento de población y del entorno empresarial en el rango adyacente (inmediato) a esa ubicación, similarmente, se recalcula la ganancia de las ubicaciones existentes.

```
    var set of int: x_rango = max(0, ubicaciones_nuevas
         [i, 1]-1)..min(n-1, ubicaciones_nuevas[i,
         1]+1)
    var set of int: y_rango = max(0, ubicaciones_nuevas
         [i, 0]-1)..min(n-1, ubicaciones_nuevas[i,
         0]+1)
```

  Esto asegura que se consideren las celdas adyacentes a la nueva ubicación (i, j) dentro de los límites de la matriz n x n
- solve maximize ganancia total: Indica que el objetivo del modelo es maximizar la ganancia total, el solver intentará encontrar las ubicaciones nuevas que maximicen la suma de las ganancias calculadas, respetando las restricciones establecidas.

III. EJEMPLOS DE PRUEBA

Dentro de los ejemplos primero debemos mirar como es el tipo de entrada esperada para que el modelo funcione, y esta es de la siguiente manera: la primera linea sera la cantidad de programas ya existentes, las siguientes lineas seran dependientes de la primera, pues que si por ejemplo tenemos 2, las siguientes dos lineas seran las coordenadas (x,y) de los programas ya existentes, luego tendremos el tamaño de las matrices de segmento y entorno, luego dependiendo de este tamaño tendremos las matrices de segmento y entorno juntas, por ultimo tendremos la cantidad de ubicaciones nuevas que queremos, esto corresponde a la entrada del modelo.

También debemos entender el formato de la salida, o la solucion que se espera arroje el modelo, esta sera como la de entrada, pero en la primera lien tendremos la ganancia total sin las nuevas ubicaciones, y luego la ganancia total con las nuevas ubicaciones, despues las coordenadas de los programas ya existentes y por ultimo las coordenadas de las nuevas ubicaciones.

Ahora teniendo en cuenta esto, podemos ver un ejemplo de entrada y solucion del modelo minizinc.

A. Ejemplo 1

```
1
1 3
5
5 10 15 20 25
10 15 20 25 30
15 20 25 30 35
20 25 30 35 40
25 30 35 40 45
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
2
```

En este caso la ganancia sin las entradas nuevas seria el calculo de la suma de la posicion (1,3) en la matriz de segmento y entorno.

- Suma de la posicion (1,3) y sus adyacentes en la matriz de segmento es

```
summa_segmento = 15 + 20 + 25 + 20 + 25 + 30 + 25 +
      30 + 35 = 225
```

  tenemos que se cumple la restriccion de que la suma de la posicion y sus adyacentes en la matriz de segmento tiene que ser ≥ 25.
- Suma de la posicion (1,3) y sus adyacentes en la matriz de entorno es

```
summa_entorno = 3 + 4 + 5 + 4+ 5 + 6 + 5 + 6 + 7 =
     45
```

  tenemos que se cumple la restriccion de que la suma de la posicion y sus adyacentes en la matriz de entorno tiene que ser ≥ 20.

Tenemos que la ganancia total sin las nuevas ubicaciones es 225 + 45 = 270, ahora al ejecutar el modelo nos genera la siguiente salida.

```
270
918
1 3
3 3
3 1
```

los nuevos programas estan en las posiciones (3,3) y (3,1) ahora calculemos si se cumplen las restricciones de las nuevas ubicaciones y la ganancia total con las nuevas ubicaciones.

- **Posicion (3,3) en la matriz.**
  - Suma de la posicion (3,3) y sus adyacentes en la matriz de segmento es

```
summa_segmento = 25 + 30 + 35 + 30 + 35 + 40
      + 35 + 40 + 45 = 315
se cumple la restriccion.
```

  - Suma de la posicion (3,3) y sus adyacentes en la matriz de entorno es

```
summa_entorno = 5 + 6 + 7 + 6 + 7 + 8 + 7 +
     8 + 9 = 63
se cumple la restriccion.
```

  La ganancia total de la ubicacion (3,3) es: 315 + 63 = 378.
- **Posicion (3,1) en la matriz.**
  - Suma de la posicion (3,1) y sus adyacentes en la matriz de segmento es

```
summa_segmento = 15 + 20 + 25 + 20 + 25 + 30
      + 25 + 30 + 35 = 225
se cumple la restriccion.
```

  - Suma de la posicion (3,1) y sus adyacentes en la matriz de entorno es

```
summa_entorno = 3 + 4 + 5 + 4 + 5 + 6 + 5 +
     6 + 7 = 45
se cumple la restriccion.
```

  La ganancia total de la ubicacion (3,1) es: 225 + 45 = 270.
- La ganancia total con las nuevas ubicaciones es 378 + 270 = 648, pero a esta le tenemos que sumar la ganacias que ya teniamos, entonces la ganancia total con las nuevas ubicaciones es 648 + 270 = 918.

Efectivamente el modelo nos arroja una solucion correcta, cumpliendo con las restricciones y maximizando la ganancia.

B. Ejemplo 2

```
1
3 3
7
5 8 3 12 7 14 2
9 4 11 6 15 1 10
8 13 2 7 12 5 3
6 1 9 14 3 8 7
11 7 4 2 10 9 6
2 14 5 13 8 12 4
10 3 6 11 7 15 9
12 1 14 8 2 10 5
4 9 7 3 11 6 12
13 2 8 5 10 4 7
5 7 3 6 12 8 1
11 6 9 4 14 3 10
2 10 13 5 7 11 8
4 7 3 9 5 10 15
3
```

- Suma de la posicion (3,3) y sus adyacentes en la matriz de segmento es:

```
summa_segmento = 2 + 7 + 12 + 9 + 14 + 3 + 4 + 2 +
     10 = 63
Se cumple la restriccion.
```

- Suma de la posicion (3,3) y sus adyacentes en la matriz de entorno es:

```
summa_entorno = 8 + 5 + 10 + 3 + 6 + 12 + 9 + 4 +
     14 = 71
Se cumple la restriccion.
```

  La ganancia total sin las nuevas ubicaciones es 63 + 71 = 134, ahora al ejecutar el modelo nos genera la siguiente salida.

```
134
576
3 3
3 1
5 1
5 5
```

Las ubicaciones nuevas son (3,1), (5,1) y (5,5), ahora calculemos si se cumplen las restricciones de las nuevas ubicaciones y la ganancia total con las nuevas ubicaciones.

- **Posicion (3,1) en la matriz.**
  - Suma de la posicion (3,1) y sus adyacentes en la matriz de segmento es

```
summa_segmento = 3 + 12 + 7 + 11 + 6 + 15 +
     2 + 7 + 12 = 75
se cumple la restriccion.
```

– Suma de la posicion (3,1) y sus adyacentes en la matriz de entorno es

```
summa_entorno = 14 + 8 + 2 + 7 + 3 + 11 + 8
     + 5 + 10 = 68
se cumple la restriccion.
```

La ganancia total de la ubicacion (3,1) es: 75 + 68 = 143.

- **Posicion (5,1) en la matriz.**
  – Suma de la posicion (5,1) y sus adyacentes en la matriz de segmento es

```
summa_segmento = 7 + 14 + 2 + 15 + 1 + 10 +
     12 + 5 + 3 = 69
se cumple la restriccion.
```

  – Suma de la posicion (5,1) y sus adyacentes en la matriz de entorno es

```
summa_entorno = 2 + 10 + 5 + 11 + 6 + 12 +
     10 + 4 + 7 = 67
se cumple la restriccion.
```

  La ganancia total de la ubicacion (5,1) es: 69 + 67 = 136.
- **Posicion (5,5) en la matriz.**
  – Suma de la posicion (5,5) y sus adyacentes en la matriz de segmento es

```
summa_segmento = 10 + 9 + 6 + 8 + 12 + 4 + 7
      + 15 + 9 = 80
se cumple la restriccion.
```

  – Suma de la posicion (5,5) y sus adyacentes en la matriz de entorno es

```
summa_entorno = 14 + 3 + 10 + 7 + 11 + 8 + 5
      + 10 + 15 = 83
se cumple la restriccion.
```

  La ganancia total de la ubicacion (5,5) es: 68 + 73 = 141.
- La ganancia total con las nuevas ubicaciones es 143 + 136 + 141 = 420, pero a esta le tenemos que sumar la ganacias que ya teniamos, entonces la ganancia total con las nuevas ubicaciones es 420 + 134 = 576.

Otra vez el modelo nos arroja una solucion correcta, cumpliendo con las restricciones y maximizando la ganancia.

C. Ejemplo 3

```
1
5 5
10
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0
2
```

En este ejemplo buscamos generar que no se cumplan las restricciones esto para ver si el modelo esta correctamente implementado, para ello usaremos una matriz de tamaño 10 que esta llena de 0's y 1's, ahora al ejecutar el modelo nos genera la siguiente salida.

```
Resolviendo el modelo...

Contenido completo del resultado:

None
```

El none representa de que no existe solución en esta entrada, ni siquiera el programa ya instalado en la matriz genera la ganancia porque no cumple con las restricciones.

IV. CONCLUSIONES

En este proyecto se desarrolló y llevó a cabo un modelo en MiniZinc para resolver el problema de incorporar nuevas escuelas de ingeniería, buscando maximizar las ganancias bajo limitaciones específicas de segmentación poblacional y contexto empresarial. A partir de este análisis, se obtienen las siguientes conclusiones:

- **Efectividad en la creación del modelo:**
  La implementación de restricciones específicas y funciones objetivo claramente definidas aseguraron que las localizaciones elegidas satisfacieran los requisitos mínimos establecidos, tales como la densidad de población y un entorno empresarial propicio.
- **Versatilidad y ampliabilidad del modelo:**
  La creación del modelo en MiniZinc resultó ser eficaz para gestionar diversas restricciones y variables, lo que permite su ajuste a otros escenarios o dimensiones, como matrices más extensas o distintos parámetros de elección.
- **Optimización asegurada:**
  Mediante la programación por restricciones, el modelo pudo determinar disposiciones ideales para la implementación de nuevas escuelas, incrementando la ganancia total y evitando choques con ubicaciones ya existentes.
- **Viabilidad de ejecución:**
  Los resultados generados por el modelo son pertinentes a situaciones reales de asignación de recursos, evidenciando la utilidad de las herramientas de optimización en la formulación de decisiones estratégicas.

En resumen, el desarrollo de este proyecto destaca la importancia de la optimización basada en restricciones como herramienta clave para resolver problemas complejos de ingeniería. Las mejoras futuras pueden incluir la incorporación de parámetros adicionales, como costos de instalación o impactos

ambientales, para fortalecer el análisis y acercarlo a escenarios del mundo real.

V. Bibliografia y Referencias

References

[1] "The MiniZinc Handbook 2.8.7", *More Complex Models — The MiniZinc Handbook 2.8.7*. Accedido el 27 de diciembre de 2024. [En línea]. Disponible: https://docs.minizinc.dev/en/stable/modelling2.html

[2] F. Rossi, P. Van Beek, and T. Walsh, Eds., *Handbook of Constraint Programming*. Elsevier, 2006.

[3] "The MiniZinc Handbook 2.8.7", *Basic Modelling in MiniZinc — The MiniZinc Handbook 2.8.7*. Accedido el 27 de diciembre de 2024. [En línea]. Disponible: https://docs.minizinc.dev/en/stable/modelling.html

[4] "MiniZinc Python 0.9.0 documentation", *Getting Started — MiniZinc Python 0.9.0 documentation*. Accedido el 27 de diciembre de 2024. [En línea]. Disponible: https://python.minizinc.dev/en/latest/getting_started.html

[5] "The MiniZinc Handbook 2.5.5", *Global constraints — The MiniZinc Handbook 2.5.5*. Accedido el 27 de diciembre de 2024. [En línea]. Disponible: https://docs.minizinc.dev/en/2.5.5/lib-globals.html