COMPLEJIDAD Y OPTIMIZACIÓN (PROYECTO 2 ADA 2)

Victor Manuel Alzate, Juan Esteban Lopez, Diego Alejandro Tolosa, Daniel Melendez

Facultad de ingeniería, Universidad del valle

Tuluá, Colombia

alzate.victor@correounivalle.edu.co

juan.aranzazu@correounivalle.edu.co

diego.tolosa@correounivalle.edu.co

daniel.melendez@correounivalle.edu.co

Abstract- Este informe presenta el desarrollo de un modelo de optimización para la ubicación de nuevas sedes utilizando la integración de MiniZinc y Python. El modelo busca determinar la mejor ubicación para las nuevas sedes, considerando restricciones como la no adyacencia a sedes existentes y la adecuada separación entre nuevas ubicaciones. A través de un enfoque basado en programación matemática, el modelo maximiza el impacto de las nuevas sedes en términos de segmento de población y entorno empresarial. El informe describe la formulación del modelo, las herramientas utilizadas para su implementación, los experimentos realizados, y los resultados obtenidos, seguidos de un análisis detallado de las conclusiones y las posibles direcciones futuras para la mejora del modelo.

I. INTRODUCCIÓN

En un escenario en el que una universidad busca expandir su programa de Ingeniería de Sistemas para admitir más estudiantes, surge un desafío el cual es: determinar las ubicaciones óptimas para las nuevas sedes del programa. El problema se plantea como un reto de ubicación de nuevas sedes, donde la decisión debe tomar en cuenta varias restricciones y criterios específicos, tales como la distancia entre las nuevas localizaciones y las existentes, el segmento de población en cada área, y la disponibilidad del entorno empresarial en los municipios seleccionados.

II. OBJETIVO

El principal objetivo de este proyecto es desarrollar un modelo de optimización para la ubicación de nuevas sedes, considerando dos factores clave: el segmento de población y el entorno empresarial. A través de la programación matemática, se busca maximizar la suma combinada de estos dos factores, mientras se respetan las siguientes restricciones:

- No adyacencia con sedes existentes.
- No adyacencia entre nuevas sedes.
- La suma del segmento de población debe ser al menos 25.
- La suma del entorno empresarial debe ser al menos 20.

III. EXPLICACIÓN DEL MODELO

A. *Datos de entrada del modelo*

Se usa un archivo de texto “.txt” para la entrada de datos con las siguiente estructura:

- La primera línea tiene el número de coordenadas ya existentes (número de programas que ya existen)
- Las siguientes líneas tiene las coordenadas de cada sede (x, y)
- La siguiente línea tiene el tamaño de la matriz (N*N)
- Las siguientes N filas y N columnas tienen la matriz de segmento de población
- Las siguientes N filas y N columnas tienen la matriz de entorno empresarial
- La última línea tiene el número de programas que se van a ubicar

B. *Contexto del problema*

Diseñar un modelo que permita ubicar los nuevos programas de ingeniería de sistemas, de tal forma se maximice el segmento de población y el entorno empresarial que se cubre, cumpliendo las restricciones

- Los nuevos programas no pueden ser contiguos a ninguno de los establecidos
- El segmento de población se toma sumando en el punto que va estar el nuevo programa y sumando los contiguos, este no puede ser menor que 25
- El entorno empresarial se toma sumando en el punto que va estar el nuevo programa y sumando los contiguos de la zona y no puede ser menos a 20

C. *Variables del modelo*

```
int: n; % Tamaño de las matrices
array[1..n, 1..n] of int: population; % Matriz de segmento de población
array[1..n, 1..n] of int: enterprise; % Matriz de entorno empresarial

int: programs; % Cantidad de programas que ya existen
array[1..programs, 1..2] of int: positions; % Coordenadas de los programas existentes

int: new_programs; % Cantidad de nuevos programas a ubicar

% Coordenadas de los nuevos programas
array[1..new_programs, 1..2] of var 1..n: new_positions;
```

En el modelo se definieron las siguientes variables

- n: El cual representa el tamaño de las matrices del segmento de población y el entorno empresarial
- population: El cual representa la matriz de segmento de población
- enterprise: El cual representa la matriz de entorno empresarial
- programs: Número de programas existentes
- positions: Coordenadas de cada programa existente
- new_programs: Número de nuevos programas para ubicar
- new_positions: Coordenadas de cada nuevo programa

Para las coordenadas existentes y las nuevas por ubicar se definen como arreglo el cual contienen las coordenadas definidas como (filas, columnas).

Para el segmento de población y entorno empresarial se usan también el arreglo de tamaño N*N.

D. *Funciones del modelo*

```
% Función para calcular la suma de una celda y sus vecinas
function int: sum_neighbors(int: x, int: y, array[1..n, 1..n] of int: matrix) =
  sum(
    i in max(1, x-1)..min(n, x+1),
    j in max(1, y-1)..min(n, y+1)
  ) (
    matrix[i, j]
  );

% Función para verificar si dos puntos son adyacentes
function var bool: is_adjacent(var int: x1, var int: y1, var int: x2, var int: y2) =
  abs(x1 - x2) <= 1 /\ abs(y1 - y2) <= 1;
```

En el modelo se definieron las siguientes funciones

is_adjacent: Esta función se utiliza para verificar si dos ubicaciones están adyacentes entre sí. En otras palabras, esta función determina si dos coordenadas están "cerca" una de la otra, lo cual es útil para asegurarse de que los nuevos programas no se ubiquen demasiado cerca de otros programas existentes o entre ellos mismos.

Parámetros de entrada: Esta función recibe dos coordenadas (filas, columnas).

Salida: Esta función devuelve un booleano True o False.

Lógica: La función calcula la diferencia entre las coordenadas x y y de las ubicaciones, usando la función abs() para obtener el valor absoluto, luego evalúa si la diferencia en x es <= 1 y la diferencia en y es <= 1 entonces quiere decir que dos ubicaciones son adyacentes.

sum_neighbors: Esta función calcula la suma de los valores de la matriz para la celda en la posición (x, y) y sus celdas adyacentes. Es útil para verificar si una ubicación cumple con los requisitos de población y entorno empresarial.

Parámetros de entrada: La función recibe una matriz (arreglo) y una ubicación (filas, columnas).

Salida: La función devuelve un número entero.

Lógica: La función usa la expresión sum() para sumar los valores de la matriz para las celdas adyacentes alrededor de (filas, columnas), para esto usa el rango de filas y columnas usando las expresión max() y min(), esto se asegura de no salir fuera de los límites de la matriz.

E. *Restricciones del modelo*

```
% Matriz precalculada para el segmento de población
array[1..n, 1..n] of int: population_neighbors = array2d(1..n, 1..n, [
  sum_neighbors(x, y, population)
  | x in 1..n, y in 1..n
]);

% Matriz precalculada para el entorno empresarial
array[1..n, 1..n] of int: enterprise_neighbors = array2d(1..n, 1..n, [
  sum_neighbors(x, y, enterprise)
  | x in 1..n, y in 1..n
]);
```

Se definieron las variables population_neighbors y enterprise_neighbors, las cuales son arreglos que representa las matrices de segmento de población y entorno empresarial, donde estas variables almacenan los resultados de las sumas de las celdas adyacentes de cada ubicación, esto con el fin de optimizar el modelo y no se tenga que recalcular varias veces sino que se invoquen las variables cuando se requieran.

```
% Restricciones
constraint
  % Los nuevos programas no pueden ser contiguos a los existentes
  forall(i in 1..new_programs, j in 1..programs) (
    not is_adjacent(new_positions[i, 1], new_positions[i, 2],
                    positions[j, 1], positions[j, 2])
  ) /\

  % Los nuevos programas no pueden ser contiguos entre sí
  forall(i in 1..new_programs, j in i+1..new_programs) (
    not is_adjacent(new_positions[i, 1], new_positions[i, 2],
                    new_positions[j, 1], new_positions[j, 2])
  ) /\

  % La suma del segmento de población debe ser al menos 25
  forall(i in 1..new_programs) (
    population_neighbors[new_positions[i, 1], new_positions[i, 2]] >= 25
  ) /\

  % La suma del entorno empresarial debe ser al menos 20
  forall(i in 1..new_programs) (
    enterprise_neighbors[new_positions[i, 1], new_positions[i, 2]] >= 20
  );
```

En el modelo se definieron las siguientes restricciones

Los nuevos programas no pueden ser contiguos a los programas existentes

Explicación:

- Esta restricción asegura que los nuevos programas no se ubiquen adyacentes a los programas que ya existen (positions).
- new_positions[i, 1], new_positions[i, 2] son las coordenadas de cada nuevo programa, mientras que positions[j, 1], positions[j, 2] son las coordenadas de los programas ya existentes.
- La función is_adjacent(...) verifica si las posiciones de los nuevos programas y los existentes son adyacentes (es decir, si están en celdas vecinas). Si lo son, la restricción impide que se ubiquen en esas posiciones, de modo que not is_adjacent(...) asegura que las posiciones no estén adyacentes.

Los nuevos programas no pueden ser contiguos entre sí

Explicación:

- Esta restricción asegura que los nuevos programas no se ubiquen adyacentes entre sí.
- new_positions[i, 1], new_positions[i, 2] y new_positions[j, 1], new_positions[j, 2] son las coordenadas de los nuevos programas que estamos ubicando.
- is_adjacent(...) verifica si los nuevos programas están adyacentes. Si lo están, not is_adjacent(...) evitará que eso ocurra.

La suma del segmento de población debe ser al menos 25

Explicación:

- Esta restricción asegura que el segmento de población en las celdas adyacentes a la ubicación de cada nuevo programa sea al menos 25.
- sum_neighbors(population, new_positions[i, 1], new_positions[i, 2]) calcula la suma del segmento de población en la celda (new_positions[i, 1], new_positions[i, 2]) y sus celdas adyacentes.
- Si la suma es menor que 25, esta restricción impide que esa ubicación sea considerada para un nuevo programa.

La suma del entorno empresarial debe ser al menos 20

Explicación:

- Esta restricción asegura que el entorno empresarial en las celdas adyacentes a la ubicación de cada nuevo programa sea al menos 20.
- sum_neighbors(enterprise, new_positions[i, 1], new_positions[i, 2]) calcula la suma del entorno empresarial en la celda (new_positions[i, 1], new_positions[i, 2]) y sus celdas adyacentes.
- Si la suma es menor que 20, esta restricción impide que esa ubicación sea considerada para un nuevo programa.

F. *Función objetivo del modelo*

```
% Maximizar la suma combinada del segmento de población y entorno empresarial
var int: total_score =
  sum(i in 1..new_programs)(
    population_neighbors[new_positions[i, 1], new_positions[i, 2]] +
    enterprise_neighbors[new_positions[i, 1], new_positions[i, 2]]
  );

% Ganancia total antes de las nuevas localizaciones
var int: existing_score =
  sum(i in 1..programs)(
    population_neighbors[positions[i, 1], positions[i, 2]] +
    enterprise_neighbors[positions[i, 1], positions[i, 2]]
  );

% Ganancia total después de las nuevas localizaciones
var int: final_score = existing_score + total_score;

solve maximize total_score;
```

En el modelo se definió la siguiente función objetivo

Explicación:

- La función objetivo busca maximizar el impacto combinado de la población y el entorno empresarial en las ubicaciones de los nuevos programas.
- Se calcula sumando, para cada nuevo programa, la población y el entorno empresarial en la ubicación seleccionada.
- Maximizar esta función objetivo asegura que los nuevos programas se ubiquen en lugares que no solo tengan suficiente población, sino que también estén en zonas con un entorno empresarial favorable.

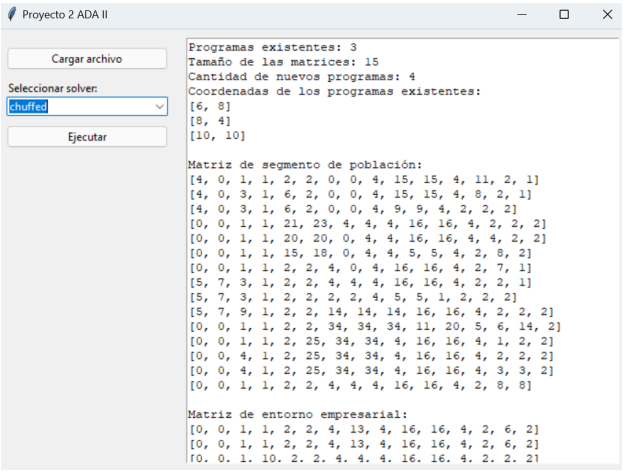
$$Max\ Z\ =\ \Sigma\ (P(new\_positions\_i)\ +\ E(new\_positions\_i))$$

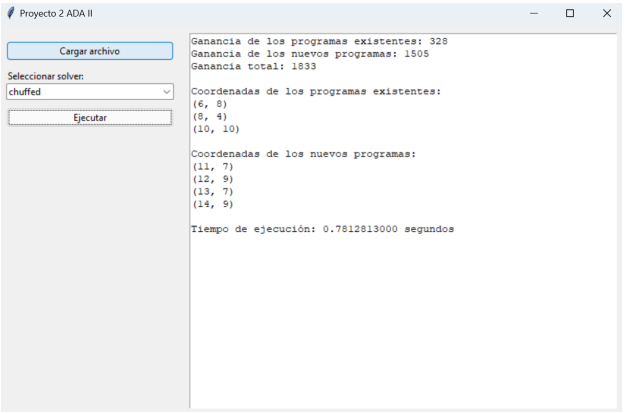
Donde Z sería la función objetivo, P representa el segmento de población y E el entorno empresarial.

IV. SECCIÓN DE PRUEBAS

La sección de pruebas tiene como objetivo validar que el modelo de ubicación de las nuevas sedes se comporte de acuerdo con las expectativas y entregue soluciones viables que maximicen la ganancia combinada de la población y el entorno empresarial. Para esto, se realizaron diversas pruebas con diferentes configuraciones de entrada. A continuación se detallan las pruebas realizadas.

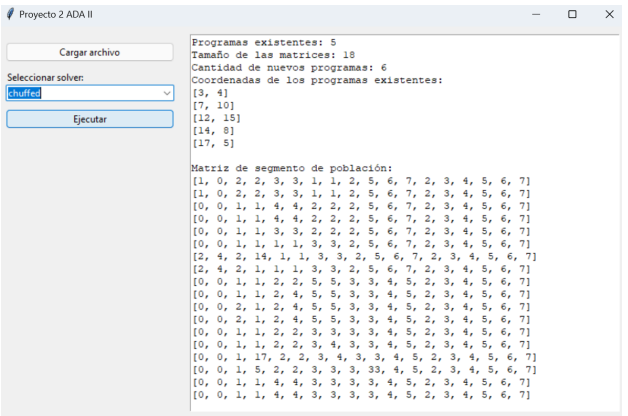
Ejemplo 1

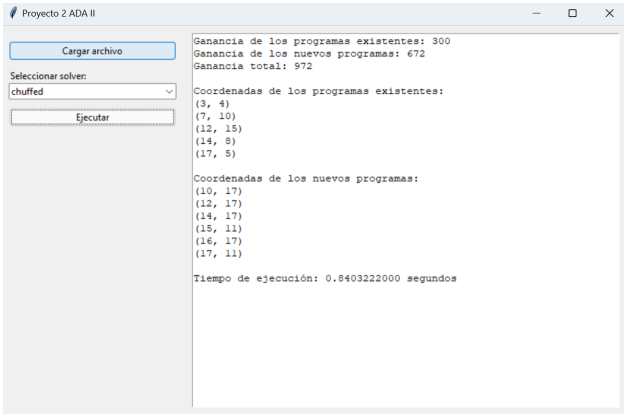




Analizando los resultados para el ejemplo 1, podemos ver que la ganancia total de los nuevos programas (1505) representa un aumento significativo en comparación con la ganancia de los programas existentes (328), además las coordenadas de los nuevos programas (11, 7), (12, 9), (13, 7) y (14, 9) muestran una disposición que mantiene una relativa cercanía entre las nuevas sedes, pero con una alineación que parece optimizar los recursos disponibles en su entorno inmediato.

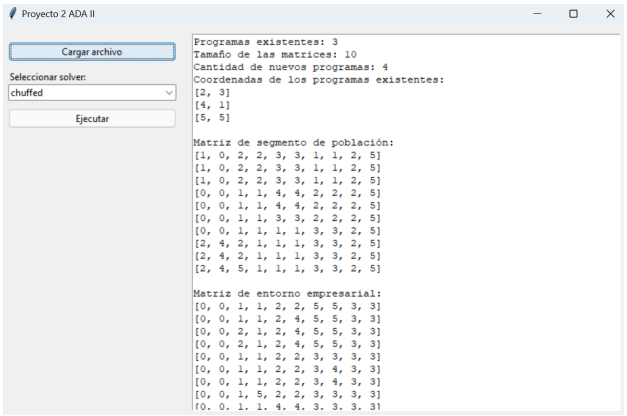
Ejemplo 2

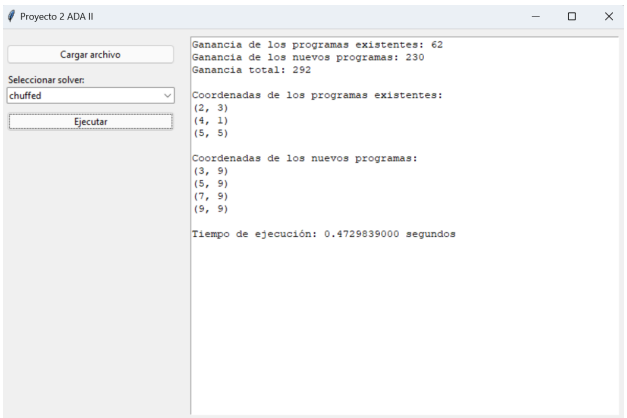




Analizando los resultados para ejemplo 2, podemos ver que la ganancia de los nuevos programas (672) supera ampliamente la ganancia de los programas existentes (300), además las coordenadas de los nuevos programas (10, 17), (12, 17), (14, 17), (15, 11), (16, 17) y (17, 11) se concentran principalmente en una zona de la matriz, específicamente en las coordenadas 17, lo que indica que las nuevas sedes se agrupan cerca de ciertas ubicaciones estratégicas, probablemente en áreas con un alto potencial de ganancia.

Ejemplo 3





Analizando los resultados para el ejemplo 3, podemos ver que la ganancia total de los nuevos programas (230) representa un aumento significativo en comparación con la ganancia de los programas existentes (62), además las coordenadas de los nuevos programas (3, 9), (5, 9), (7, 9) y (9, 9) están todas concentradas en una misma fila (y = 9), lo que podría indicar que el modelo ha priorizado las ubicaciones con alta ganancia en una zona específica del espacio disponible, probablemente donde la densidad de población o las oportunidades empresariales son más atractivas.

Comparando los resultados, se puede concluir que las nuevas ubicaciones para cada ejemplo satisface la maximización del segmento de población y el entorno empresarial, esto tambien depende principalmente de los datos de entrada, pero se pudo ver que los nuevos programas obtenían ganancias más altas que los programas existentes, por otro lado dependiendo de donde se concentre más el segmento de población y el entorno empresarial los nuevos programas quedarán ubicados muy cerca entre sí, pero cumpliendo la restricción que no sean contiguos, además al probar los 3 diferentes solvers, los cuales son gecode, chuffed y cp-sat, el solver con mejor rendimiento en términos de tiempo fue chuffed.

También cabe resaltar que en algunos casos es posible que no se obtenga una solución al problema, porque posiblemente no se cumplan las restricciones definidas, para estos casos el programa devuelve que no encontro una solucion al problema, esto depende principalmente de la entrada de datos, por ejemplo es posible que en ninguna ubicación se cumpla la sumatoria >= 25 para el segmento de población o >= 20 para el entorno empresarial, también que no se pueda ubicar los programas entre sí de tal forma que cumpla que no sean contiguos.

V. CONCLUSIONES

El presente informe ha abordado el problema de determinar las ubicaciones óptimas para las nuevas sedes del programa académico de Ingeniería de Sistemas, A través de un enfoque matemático basado en la programación con restricciones, se establecieron criterios claros para asegurar que las nuevas ubicaciones cumplieran con requisitos específicos, como no ser contiguas a otras posiciones preexistentes ni entre sí, además de cumplir con valores mínimos de población y entorno empresarial en las celdas seleccionadas y sus

adyacentes tanto la población como el entorno empresarial.

En conclusión, el modelo presentado constituye un paso importante en la búsqueda de soluciones optimizadas para la expansión de programas académicos.

VI. REFERENCIAS

[1] MiniZinc Python, MiniZinc Python 0.9.0 documentation. [En línea]. Disponible en: https://minizinc-python.readthedocs.io/en/0.9.0/ (accedido: 30-nov-2024).

[2] The MiniZinc Handbook 2.8.7, "2.1. Modelado básico en MiniZinc". [En línea]. Disponible en: https://docs.minizinc.dev/en/stable/modelling.html (accedido: 30-nov-2024).

[3] MiniZinc Team, Modelling with MiniZinc – Chapter 2: Basic Modelling. [En línea]. Disponible en: https://docs.minizinc.dev/en/stable/modelling2.html (accedido: 30-nov-2024).

[4] Introduction to MiniZinc, YouTube, 2021. [En línea]. Disponible en: https://youtu.be/z1-rTlIu2aM (accedido: 30-nov-2024).

[5] MiniZinc Team, MiniZinc-Python: Access to all MiniZinc functionality directly from Python. GitHub. [En línea]. Disponible en: https://github.com/MiniZinc/minizinc-python (accedido: 30-nov-2024).