Informe de Proyecto 2 de Análisis de Algoritmos II

Kevin Alejandro Marulanda – 238097 - 3743

Universidad del Valle del Cauca

Análisis de Algoritmos II

Profesor Carlos Andrés Delgado Saavedra

27/12/2024

Desarrollo de una Aplicación de Interfaz Gráfica para Optimización con Python, Tkinter y MiniZinc

1. Introducción

El informe presenta una aplicación de interfaz gráfica de usuario (GUI) desarrollada con Tkinter para resolver problemas de optimización utilizando el lenguaje de modelado MiniZinc. El objetivo es crear una herramienta fácil de usar que permita a los usuarios cargar archivos de entrada, definir un modelo MiniZinc y obtener soluciones optimizadas de forma interactiva. La aplicación se basa en la librería PyMiniZinc para la interacción con el lenguaje de modelado y el solver Gecode para la resolución de los problemas.

2. Secciones de Muestra

1. Diseño de la Interfaz

La El programa GUI se compone de dos elementos principales:

- **Botón "Abrir Archivo"**: Permite al usuario seleccionar un archivo de texto (.txt) o .dzn que contiene los datos de entrada.
- **Área de texto**: Muestra los resultados del modelo MiniZinc después de la ejecución en el área de texto.

2. Clasificación de Datos de Entrada

La aplicación emplea la clase DataClassifier para manejar los datos que se ingresan. Esta clase examina los datos y los organiza en estructuras que son compatibles con el modelo MiniZinc. El proceso de clasificación abarca la determinación del tamaño de la matriz, la localización de las coordenadas, la cantidad de programas nuevos, así como la población y el contexto empresarial.

2.1 Método `classify_data_txt`

Este método tiene la función de organizar y clasificar los datos contenidos en un archivo de texto. Se requiere que este archivo siga un formato específico, el cual se detalla a continuación: la primera línea del archivo debe indicar la cantidad total de posiciones disponibles. Las líneas subsiguientes enumeran las coordenadas correspondientes a cada una de estas posiciones. Después de las coordenadas, se especifica el tamaño de la matriz. Las líneas posteriores representan la matriz de segmentos de población, seguidas por la matriz del entorno empresarial. Finalmente, la última línea del archivo señala el número de nuevos programas. Este método se encarga de procesar y almacenar la información en variables de instancia de la clase, tales como num_positions, coordinates, matrix_size, population_matrix, business_matrix, y num_new_programs.

```
19      def classify_data_txt(self, content):
23          """
24          try:
25              content = [line.strip() for line in content if line.strip()]  # Limpia líneas vacías
26              self.num_positions = int(content[0])
27
28              # Coordenadas de localizaciones existentes
29              self.coordinates = [list(map(int, content[i + 1].split())) for i in range(self.num_positions)]
30
31              # Tamaño de la matriz
32              matrix_size_line_index = 1 + self.num_positions
33              self.matrix_size = int(content[matrix_size_line_index])
34
35              # Matriz de segmento de población
36              n = self.matrix_size
37              population_matrix_start = matrix_size_line_index + 1
38              population_matrix_end = population_matrix_start + n
39              self.population_matrix = [list(map(int, content[i].split())) for i in range(population_matrix_start, population_matrix_end)]
40
41              # Matriz de entorno empresarial
42              business_matrix_start = population_matrix_end
43              business_matrix_end = business_matrix_start + n
44              self.business_matrix = [list(map(int, content[i].split())) for i in range(business_matrix_start, business_matrix_end)]
45
46              # Número de programas nuevos
47              self.num_new_programs = int(content[business_matrix_end])
```

2.2 Método `classify_data_dzn`

Este método clasifica los datos de un archivo DZN, que es un formato estándar para la entrada de datos en MiniZinc. El método utiliza expresiones regulares para extraer la información relevante del archivo, como el tamaño de la matriz, las coordenadas, la población, el entorno empresarial y el número de programas nuevos.

```
52      def classify_data_dzn(self, content):
57          try:
58              data = "".join(content)
59
60              def extract_array(name):
61                  import re
62                  pattern = f"{name}\s*=\s*array2d\(.*?,\s*\[(.*?)\]\);"
63                  match = re.search(pattern, data, re.DOTALL)
64                  if not match:
65                      raise ValueError(f"{name} no encontrado")
66                  array_data = match.group(1).replace("\n", "").split(",")
67                  return [int(val.strip()) for val in array_data]
68
69              self.matrix_size = int(data.split("n = ")[1].split(";")[0].strip())
70              self.num_positions = int(data.split("num_existing_positions = ")[1].split(";")[0].strip())
71
72              coord_data = data.split("existing_positions = [| ")[1].split(" |]")[0].strip()
73              self.coordinates = [list(map(int, pair.split(","))) for pair in coord_data.split("|")]
74
75              population_flat = extract_array("population_segment")
76              self.population_matrix = [population_flat[i:i + self.matrix_size] for i in range(0, len(population_flat), self.matrix_size)]
77
78              business_flat = extract_array("business_environment")
79              self.business_matrix = [business_flat[i:i + self.matrix_size] for i in range(0, len(business_flat), self.matrix_size)]
80
81              self.num_new_programs = int(data.split("num_new_programs = ")[1].split(";")[0].strip())
```

3. Ejecución del Modelo MiniZinc

La aplicación utiliza la función `execute_minizinc_model` para ejecutar el modelo MiniZinc seleccionado por el usuario. La función recibe la ruta del modelo y los datos formateados. Luego, crea una instancia del modelo, proporciona los datos y resuelve el problema utilizando el solver Gecode.

3.1 Método `format_for_minizinc`

Este procedimiento organiza los datos categorizados en un formato que es compatible con MiniZinc. Produce una cadena de texto que incluye las variables y valores requeridos para llevar a cabo el modelo.

```
86      def format_for_minizinc(self):
87          """Formatea los datos clasificados para que se ajusten a la entrada esperada en MiniZinc."""
88          formatted_data = []
89          n = self.matrix_size
90
91          # Tamaño de la matriz
92          formatted_data.append(f"n = {n};")
93          # Número de posiciones existentes
94          formatted_data.append(f"num_existing_positions = {self.num_positions};")
95          # Coordenadas en formato MiniZinc
96          formatted_coords = " | ".join([f"{x}, {y}" for x, y in self.coordinates])
97          formatted_data.append(f"existing_positions = [| {formatted_coords} |];")
98          # Número de nuevos programas
99          formatted_data.append(f"num_new_programs = {self.num_new_programs};")
100         # Matriz de segmento de población
101         flat_population = [str(item) for row in self.population_matrix for item in row]
102         formatted_data.append(f"population_segment = array2d(0..{n-1}, 0..{n-1}, [{', '.join(flat_population)}]);")
103         # Matriz de entorno empresarial
104         flat_business = [str(item) for row in self.business_matrix for item in row]
105         formatted_data.append(f"business_environment = array2d(0..{n-1}, 0..{n-1}, [{', '.join(flat_business)}]);")
106
107         return "\n".join(formatted_data)
```

3.2 Método `execute_minizinc_model`

Este procedimiento es fundamental para la implementación del modelo MiniZinc. La secuencia de acciones que se lleva a cabo incluye: la carga del solucionador Gecode, el formateo de la información de entrada para MiniZinc, la creación de una instancia del modelo MiniZinc, la transferencia de los datos formateados a dicha instancia, la impresión de la información formateada para fines de depuración, la resolución del modelo MiniZinc, y finalmente, la presentación del resultado del modelo en el área de texto.

```
109     def execute_minizinc_model(self, model_path):
110         """Ejecuta el modelo MiniZinc usando los datos formateados."""
111         try:
112             # Cargar el solver
113             solver = Solver.lookup("gecode")  # Asegúrate de que Gecode esté instalado
114 
115             # Formatear los datos para MiniZinc
116             input_data = self.format_for_minizinc()
117             input_file = os.path.abspath("input_data.dzn")
118             model_path = os.path.abspath(model_path)
119 
120             # Escribir los datos en un archivo temporal
121             with open(input_file, "w") as file:
122                 file.write(input_data)
123 
124             # Cargar el modelo
125             model = Model(model_path)
126 
127             # Crear una instancia del modelo
128             instance = Instance(solver, model)
```

```
130             # Proporcionar datos a la instancia
131             instance["n"] = self.matrix_size
132             instance["num_existing_positions"] = self.num_positions
133             instance["existing_positions"] = self.coordinates
134             instance["num_new_programs"] = self.num_new_programs
135             instance["population_segment"] = self.population_matrix
136             instance["business_environment"] = self.business_matrix
137 
138             # Imprimir los datos de entrada formateados
139             print("Datos de entrada formateados para MiniZinc:")
140             print(f"n: {self.matrix_size}")
141             print(f"num_existing_positions: {self.num_positions}")
142             print(f"existing_positions: {self.coordinates}")
143             print(f"num_new_programs: {self.num_new_programs}")
144             print(f"population_segment: {self.population_matrix}")
145             print(f"business_environment: {self.business_matrix}")
146 
147             # Resolver el modelo
148             result = instance.solve()
149 
150             # Mostrar resultados
151             print("Resultados de la ejecución del modelo MiniZinc:")
152             print(result)
```

```
150             # Mostrar resultados
151             print("Resultados de la ejecución del modelo MiniZinc:")
152             print(result)
153 
154             # Guardar resultados en un archivo .txt
155             output_file = "resultados.txt"  # Nombre del archivo de salida
156             with open(output_file, "w") as file:
157                 file.write("Resultados del modelo:\n\n")
158                 file.write(str(result))
159 
160             print(f"Resultados guardados en {output_file}")
161 
162             return result
163 
164         except Exception as e:
165             # Captura la salida de error
166             error_message = f"Error ejecutando MiniZinc: {str(e)}"
167             self.log_error(error_message)
168             raise RuntimeError(error_message)
```

4. Modelo MiniZinc

El modelo MiniZinc empleado por la aplicación busca determinar la ubicación ideal para un grupo de nuevos programas, considerando restricciones de cercanía a las ubicaciones ya existentes, así como límites poblacionales y factores del entorno empresarial.

4.1 Parámetros de entrada

El modelo recibe los siguientes parámetros de entrada:

- `n`: El tamaño de la matriz.
- `num_existing_positions`: El número de posiciones existentes.
- `existing_positions`: Las coordenadas de las posiciones existentes.
- `num_new_programs`: El número de programas nuevos.
- `population_segment`: La matriz de segmento de población.
- `business_environment`: La matriz de entorno empresarial.

```
% Parámetros de entrada
int: n; % Tamaño de la matriz
int: num_existing_positions; % Número de posiciones existentes
array[1..num_existing_positions, 1..2] of int: existing_positions; % Coordenadas de las posiciones actuales
int: num_new_programs; % Número de programas nuevos
array[0..n-1, 0..n-1] of int: population_segment; % Matriz de segmento de población
array[0..n-1, 0..n-1] of int: business_environment; % Matriz de entorno empresarial
```

4.2 Restricciones

El modelo incluye las siguientes restricciones:

- Los nuevos programas no pueden estar contiguos a las posiciones existentes.
- Los nuevos programas no pueden estar contiguos entre sí.
- Cada nueva posición debe cumplir las condiciones de la matriz de población (mínimo de 25).
- Cada nueva posición debe cumplir las condiciones de la matriz de entorno empresarial (mínimo de 20).

```
  % Restricciones
  constraint
      % Los nuevos programas no pueden estar contiguos a las posiciones existentes
      forall(p in 0..num_new_programs-1) (
          forall(e in 0..num_existing_positions-1) (
              abs(new_program_positions[p, 0] - adjusted_existing_positions[e, 0]) >= 3 \/
              abs(new_program_positions[p, 1] - adjusted_existing_positions[e, 1]) >= 3
          )
      )
      /\
      % Los nuevos programas no pueden estar contiguos entre sí
      forall(p1, p2 in 0..num_new_programs-1 where p1 < p2) (
          abs(new_program_positions[p1, 0] - new_program_positions[p2, 0]) >= 3 \/
          abs(new_program_positions[p1, 1] - new_program_positions[p2, 1]) >= 3
      )
      /\
```

```
      % Cada nueva posición debe cumplir las condiciones de la matriz de población
      forall(p in 0..num_new_programs-1) (
          sum([population_segment[x, y] |
              x in max(0, new_program_positions[p, 0] - 1)..min(n-1, new_program_positions[p, 0] + 1),
              y in max(0, new_program_positions[p, 1] - 1)..min(n -1, new_program_positions[p, 1] + 1)]) >=
min_population
      )
      /\
      % Cada nueva posición debe cumplir las condiciones de la matriz de entorno empresarial
      forall (p in 0..num_new_programs- 1) (
          sum([business_environment[x, y] |
              x in max(0, new_program_positions[p, 0] - 1)..min(n-1, new_program_positions[p, 0] + 1),
              y in max(0, new_program_positions[p, 1] - 1)..min(n-1, new_program_positions[p, 1] + 1)]) >= min_business
      );
```

4.3 Función objetivo

La función objetivo del modelo es maximizar la suma del segmento de población y el entorno empresarial en las nuevas posiciones seleccionadas. Esto se logra mediante la definición de variables de decisión que representan las coordenadas de los nuevos programas y la aplicación de restricciones que aseguran que se cumplan los requisitos establecidos.

```
  % Función objetivo: Maximizar la suma del segmento de población y entorno empresarial
  var int: total_population = sum([
      sum([population_segment[x, y] |
          x in max(0, sorted_new_program_positions[e, 1] - 1)..min(n-1, sorted_new_program_positions[e, 1] + 1),
          y in max(0, sorted_new_program_positions[e, 0] - 1)..min(n-1, sorted_new_program_positions[e, 0] + 1)])
      | e in 0..num_new_programs-1]);

  var int: total_business = sum([
      sum([business_environment[x, y] |
          x in max(0, sorted_new_program_positions[e, 1] - 1)..min(n-1, sorted_new_program_positions[e, 1] + 1),
          y in max(0, sorted_new_program_positions[e, 0] - 1)..min(n-1, sorted_new_program_positions[e, 0] + 1)])
      | e in 0..num_new_programs-1]);

  solve maximize total_population + total_business;
```

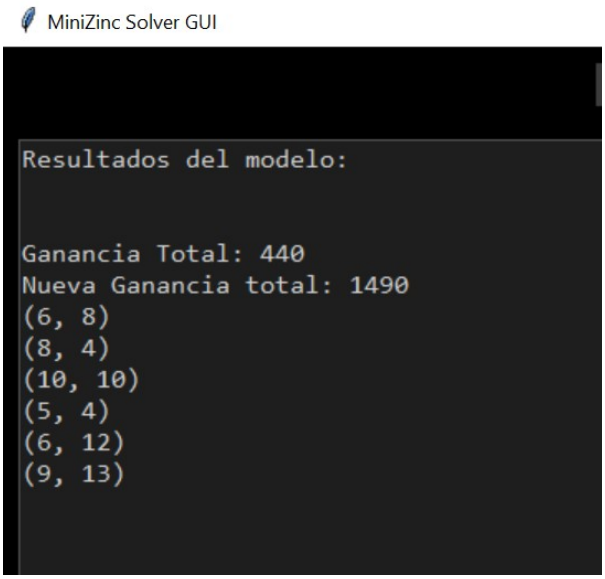
4.4 Salida

El modelo genera una salida que muestra la ganancia total de la población existente y del entorno empresarial, así como la nueva ganancia total que incluye las contribuciones de las nuevas posiciones. La salida se presenta en un formato legible que incluye las coordenadas de las posiciones existentes y las nuevas.

```
solve maximize total_population + total_business;

% Salida: Mostrar los valores alrededor de las posiciones existentes
output  ["\nGanancia Total: " ++ show(total_existing_population + total_existing_business)]
++ ["\nNueva Ganancia total: "++ show(total_existing_population + total_existing_business+total_population +
total_business) ++ "\n"]
++ [
    "(" ++ show(existing_positions[i, 1]) ++ ", " ++ show(existing_positions[i, 2]) ++ ")\n"
    | i in 1..num_existing_positions]
++ [
    "(" ++ show(sorted_new_program_positions[p, 0]) ++ ", " ++ show(sorted_new_program_positions[p, 1]) ++ ")\n"
    | p in 0..num_new_programs-1
];
```

5. Muestra de Prueba.



La Ejecución del código con los respectivos datos de entrada:

3

6 8

8 4

10 10

15

4 0 1 1 2 2 0 0 4 15 15 4 11 2 1

4 0 3 1 6 2 0 0 4 15 15 4 8 2 1

4 0 3 1 6 2 0 0 4 9 9 4 2 2 2

0 0 1 1 21 23 4 4 4 16 16 4 2 2 2

0 0 1 1 20 20 0 4 4 16 16 4 4 2 2

0 0 1 1 15 18 0 4 4 5 5 4 2 8 2

0 0 1 1 2 2 4 0 4 16 16 4 2 7 1

5 7 3 1 2 2 4 4 4 16 16 4 2 2 1

5 7 3 1 2 2 2 2 4 5 5 1 2 2 2

5 7 9 1 2 2 14 14 14 16 16 4 2 2 2

0 0 1 1 2 2 34 34 34 11 20 5 6 14 2

0 0 1 1 2 25 34 34 4 16 16 4 1 2 2

0 0 4 1 2 25 34 34 4 16 16 4 2 2 2

0 0 4 1 2 25 34 34 4 16 16 4 3 3 2

0 0 1 1 2 2 4 4 4 16 16 4 2 8 8

0 0 1 1 2 2 4 13 4 16 16 4 2 6 2

0 0 1 1 2 2 4 13 4 16 16 4 2 6 2

0 0 1 10 2 2 4 4 4 16 16 4 2 2 2

0 0 1 1 21 23 4 4 4 16 16 4 2 2 2

0 0 1 1 20 20 4 4 4 16 16 4 4 5 2

0 0 1 1 15 18 4 4 4 16 16 4 4 5 2

0 0 1 1 2 9 4 4 4 16 16 4 2 2 2

18 18 1 1 9 2 11 4 4 16 16 4 2 2 2

35 18 1 1 2 2 12 4 4 16 16 4 6 2 2

18 18 10 1 8 2 4 4 4 16 16 4 2 2 2

0 0 1 1 2 2 4 4 4 16 16 4 2 14 2

0 0 9 1 2 25 34 50 4 16 16 4 13 2 2

0 0 9 1 2 25 44 34 4 16 16 4 2 9 2

0 0 1 1 5 25 34 34 4 16 16 4 2 9 2

0 0 1 1 5 2 4 4 4 16 16 4 2 18 18

3

Nos muestra las 3 nuevas posiciones óptimas para abrir un nuevo programa de ingeniería de sistemas en todo el país, siendo por mucho superior en ganancia por si mismas las nuevas ubicaciones de programa que las posiciones existentes.

6. Conclusiones

La aplicación de interfaz gráfica de usuario (GUI) creada ofrece una manera accesible e interactiva para abordar problemas de optimización mediante MiniZinc. Esta GUI cuenta con una interfaz intuitiva que simplifica la interacción con el modelo MiniZinc y la interpretación de los resultados. Además, la aplicación emplea PyMiniZinc junto con el solucionador Gecode, lo que permite a los usuarios acceder a soluciones efectivas de manera eficiente.

7. Bibliografía

- Documentación de MiniZinc
- Documentación de PyMiniZinc
- Documentación de Tkinter