Parcial

Introducción a la IA

Cristian Barrero

Programa Ing. Ciencias de la Computación e Inteligencia Artificial
Universidad Sergio Arboleda

Bogotá, Colombia
07 de marzo del 2025

Introducción

Abordo el reto logístico de viajar desde Bogotá para conocer varias ciudades de Colombia y regresar a su ciudad de origen. Analizo el problema del turista con (TSP o El Problema clasico del Viajante) para determinar tres rutas óptimas: la más rápida (minimización del tiempo), la más corta (minimización de la distancia) y una opción balanceada que pondera ambos factores. Mediante un enfoque de fuerza bruta con permutaciones, se exploran todas las posibles secuencias de rutas, demostrando así cómo aplicar técnicas computacionales para optimizar la planificación en el transporte.

1. Desarrollo

Matriz de distancias entre ciudades (en km)

Ciudad	**Bogotá**	**Tunja**	**Chía**	**La Vega**	**Fusa**	**Girardot**	**Tabio**	**Ibagué**
Bogotá	0	138.2	39.2	56.9	78.1	139.2	45.3	198.4
Tunja	138.2	0	55.1	197.2	210.4	275.0	132.3	342.1
Chía	39.2	55.1	0	69.1	94.3	146.8	9.8	223.4
La Vega	56.9	197.2	69.1	0	117.6	170.1	64.0	230.9
Fusa	78.1	210.4	94.3	117.6	0	70.0	98.0	131.3
Girardot	139.2	275.0	146.8	170.1	70.0	0	151.0	68.5
Tabio	45.3	132.3	9.8	64.0	98.0	151.0	0	227.0
Ibagué	198.4	342.1	223.4	230.9	131.3	68.5	227.0	0

Cuadro 1: Matriz de distancias entre ciudades (en km)

Matriz de tiempos entre ciudades (en horas y minutos)

Ciudad	Bogotá	Tunja	Chía	La Vega	Fusa	Girardot	Tabio	Ibagué
Bogotá	0	2h 16m	52m	1h 14m	2h 2m	3h 30m	1h 4m	4h 18m
Tunja	2h 16m	0	1h 6m	3h 29m	4h 47m	5h 31m	2h 20m	6h 34m
Chía	52m	1h 6m	0	1h 30m	3h 24m	4h 10m	34m	5h 10m
La Vega	1h 14m	3h 29m	1h 30m	0	3h 8m	3h 58m	1h 34m	4h 48m
Fusa	2h 2m	4h 47m	3h 24m	3h 8m	0	1h 18m	2h 51m	2h 13m
Girardot	3h 30m	5h 31m	4h 10m	3h 58m	1h 18m	0	3h 31m	1h 21m
Tabio	1h 4m	2h 20m	34m	1h 34m	2h 51m	3h 31m	0	5h 11m
Ibagué	4h 18m	6h 34m	5h 10m	4h 48m	2h 13m	1h 21m	5h 11m	0

Cuadro 2: Matriz de tiempos entre ciudades (en horas y minutos)

1. Organización de la información

- Se recibieron dos matrices:
  - **Matriz de distancias** (en kilómetros).
  - **Matriz de tiempos** (en horas y minutos).
- Cada fila y columna de estas tablas correspondía a una ciudad, mostrando los valores de distancia y tiempo entre cada par de ciudades.

2. Cálculo de las rutas

- Para cada ruta posible, se sumaron los valores de la tabla correspondiente:
  - Para encontrar la **ruta más corta**, se sumaron los valores de la **matriz de distancias**.
    Bogotá → Fusagasugá → Girardot → Ibagué → La Vega → Tabio → Chía → Tunja → Barbosa → Bogotá
  - Para encontrar la **ruta más rápida**, se sumaron los valores de la **matriz de tiempos**.
    Bogotá → Tunja → Chía → Tabio → La Vega → Fusagasugá → Ibagué → Girardot → Barbosa → Bogotá
  - Para encontrar la **ruta más equilibrada**, se compararon los valores de ambas tablas para obtener un punto intermedio entre tiempo y distancia.
    Bogotá → Tunja → Chía → Tabio → La Vega → Fusagasugá → Ibagué → Girardot → Barbosa → Bogotá

3. Selección de la mejor ruta en cada caso

- Se probaron diferentes órdenes de visita a las ciudades.
- Se seleccionó la ruta con el menor valor en cada criterio.
- La información se revisó para evitar inconsistencias.

Conclusión

El problema se resolvió utilizando únicamente la información de las tablas, sumando los valores y comparando resultados. Esto permitió determinar las mejores rutas aplicando un análisis directo de los datos proporcionados.

2. Algoritmo con Python

El problema planteado se relaciona con el **Problema del Viajante** (*TSP - Traveling Salesman Problem*), un desafío clásico en optimización combinatoria donde un viajero debe recorrer varias ciudades, visitando cada una exactamente una vez y regresando al punto de partida con la mejor ruta posible. En este caso, un turista desea explorar diferentes ciudades del país minimizando el tiempo de viaje, la distancia recorrida o buscando un equilibrio entre ambos factores.

2.1. Metodología

Para resolverlo, el código utiliza un **enfoque de fuerza bruta con permutaciones**, evaluando todas las posibles secuencias de viaje y calculando el costo total de cada una según tres criterios:

- **Ruta más rápida**: Minimiza el tiempo total del recorrido.
- **Ruta más corta**: Minimiza la distancia total recorrida.
- **Ruta balanceada**: Encuentra un equilibrio entre tiempo y distancia.

Se selecciona la mejor ruta en cada categoría, garantizando la solución óptima.

2.2. Análisis de Complejidad

Dado que el número de combinaciones crece factorialmente con la cantidad de ciudades, este método es computacionalmente costoso y poco eficiente para problemas de mayor escala. En escenarios reales, se suelen emplear heurísticas o algoritmos avanzados como **Branch and Bound**, **Algoritmos Genéticos** o **Métodos Aproximados** para encontrar soluciones eficientes en menos tiempo.

```python
from itertools import permutations

# Lista de ciudades a visitar (excluyendo Bogot )
cities = ["Tunja", "Ch a", "La Vega", "Fusagasug ", "Girardot", "Tabio", "Ibagu ", "Barbosa"]

# Tiempos (en minutos) desde Bogot  hacia cada ciudad (y viceversa, asumido sim trico)
times_from_bogota = {
    "Tunja": 154,    # 2h 34min
    "Ch a": 51,      # 51min
    "La Vega": 132,  # 2h 12min
    "Fusagasug ": 211,  # 3h 31min
    "Girardot": 260, # 4h 20min
    "Tabio": 76,     # 1h 16min
    "Ibagu ": 238,   # 3h 58min
    "Barbosa": 238   # 3h 58min
}

# Tiempos de viaje entre ciudades (en minutos)
travel_times = {
    ("Tunja", "Ch a"): 66, ("Tunja", "La Vega"): 209, ("Tunja", "Fusagasug "): 287,
    ("Tunja", "Girardot"): 331, ("Tunja", "Tabio"): 140, ("Tunja", "Ibagu "): 394, ("Tunja", "Barbosa"): 508,
    ("Ch a", "La Vega"): 90, ("Ch a", "Fusagasug "): 204, ("Ch a", "Girardot"): 250,
    ("Ch a", "Tabio"): 34, ("Ch a", "Ibagu "): 310, ("Ch a", "Barbosa"): 524,
    ("La Vega", "Fusagasug "): 188, ("La Vega", "Girardot"): 238, ("La Vega", "Tabio"): 94,
    ("La Vega", "Ibagu "): 288, ("La Vega", "Barbosa"): 434,
    ("Fusagasug ", "Girardot"): 78, ("Fusagasug ", "Tabio"): 171, ("Fusagasug ", "Ibagu "): 133,
    ("Fusagasug ", "Barbosa"): 494,
    ("Girardot", "Tabio"): 211, ("Girardot", "Ibagu "): 81, ("Girardot", "Barbosa"): 432,
    ("Tabio", "Ibagu "): 311, ("Tabio", "Barbosa"): 530,
    ("Ibagu ", "Barbosa"): 491
}
# Hacemos sim trica la matriz (ruta bidireccional)
for (city1, city2), t in list(travel_times.items()):
    travel_times[(city2, city1)] = t

# Distancias (en km) desde Bogot  hacia cada ciudad (valor aproximado)
distances_from_bogota = {
    "Tunja": 140, "Ch a": 22, "La Vega": 64, "Fusagasug ": 68,
    "Girardot": 140, "Tabio": 39, "Ibagu ": 200, "Barbosa": 188
}

# Distancias de viaje entre ciudades (en km)
distance_matrix = {
```

```
    ("Tunja", "Ch a"): 55.1, ("Tunja", "La Vega"): 197.2, ("Tunja", "Fusagasug "): 210.4,
    ("Tunja", "Girardot"): 275.0, ("Tunja", "Tabio"): 132.3, ("Tunja", "Ibagu "): 342.1, ("Tunja", "Barbosa"): 373.8,
    ("Ch a", "La Vega"): 69.1, ("Ch a", "Fusagasug "): 94.3, ("Ch a", "Girardot"): 146.8,
    ("Ch a", "Tabio"): 9.8, ("Ch a", "Ibagu "): 223.4, ("Ch a", "Barbosa"): 474.5,
    ("La Vega", "Fusagasug "): 117.6, ("La Vega", "Girardot"): 170.1, ("La Vega", "Tabio"): 64.0,
    ("La Vega", "Ibagu "): 230.9, ("La Vega", "Barbosa"): 405.8,
    ("Fusagasug ", "Girardot"): 70.0, ("Fusagasug ", "Tabio"): 98.0, ("Fusagasug ", "Ibagu "): 131.3,
    ("Fusagasug ", "Barbosa"): 532.1,
    ("Girardot", "Tabio"): 151.0, ("Girardot", "Ibagu "): 68.5, ("Girardot", "Barbosa"): 464.9,
    ("Tabio", "Ibagu "): 227.0, ("Tabio", "Barbosa"): 466.2,
    ("Ibagu ", "Barbosa"): 471.3
}
# Aseguramos simetr a en la matriz de distancias
for (city1, city2), d in list(distance_matrix.items()):
    distance_matrix[(city2, city1)] = d

# Funci n para calcular el costo total de una ruta dada la funci n de costo (puede ser tiempo, distancia, o combinado)
def total_cost(route, cost_from_origin, cost_between, objective="time"):
    cost = cost_from_origin[route[0]]
    for i in range(len(route) - 1):
        cost += cost_between[(route[i], route[i+1])]
    cost += cost_from_origin[route[-1]]
    return cost

# Definir el costo combinado: para este ejemplo, sumamos tiempo (minutos) y distancia (asumiendo 1 km ~ 1 minuto)
def combined_cost(route):
    time_cost = total_cost(route, times_from_bogota, travel_times, "time")
    distance_cost = total_cost(route, distances_from_bogota, distance_matrix, "distance")
    return time_cost + distance_cost

# Buscamos la ruta  ptima  para cada criterio
min_time = float("inf")
best_time_route = None

min_distance = float("inf")
best_distance_route = None

min_combined = float("inf")
best_combined_route = None

for perm in permutations(cities):
```

```
    cost_time = total_cost(perm, times_from_bogota, travel_times, "time")
    cost_distance = total_cost(perm, distances_from_bogota, distance_matrix, "distance")
    cost_combined = combined_cost(perm)

    if cost_time < min_time:
        min_time = cost_time
        best_time_route = perm

    if cost_distance < min_distance:
        min_distance = cost_distance
        best_distance_route = perm

    if cost_combined < min_combined:
        min_combined = cost_combined
        best_combined_route = perm

# Funci n para formatear la ruta incluyendo Bogot  (punto de partida y llegada)
def format_route(route):
    return "Bogot      " + "     ".join(route) + "     Bogot "

# Imprimimos los resultados
print("1. Ruta M S R PIDA (minimiza el tiempo total):")
print(format_route(best_time_route))
print(f"Tiempo total estimado: {min_time} minutos (~{min_time/60:.2f} horas)")
print("\n2. Ruta M S CORTA (minimiza la distancia total):")
print(format_route(best_distance_route))
print(f"Distancia total estimada: {min_distance:.1f} km")
print("\n3. Ruta BALANCEADA (minimiza el costo combinado de tiempo + distancia):")
print(format_route(best_combined_route))
print(f"Costo combinado estimado: {min_combined} (minutos + km)")
```

```
PS E:\Universidad\Semestre 1\Introduccion a la IA\Parcial> py algoritmo.py
1. Ruta M S R PIDA (minimiza el tiempo total):
Bogot      Tunja     Ch a     Tabio     La Vega     Fusagasug     Ibagu     Girardot     Barbosa     Bogot
Tiempo total estimado: 1420 minutos (~23.67 horas)

2. Ruta M S CORTA (minimiza la distancia total):
Bogot      Fusagasug     Girardot     Ibagu     La Vega     Tabio     Ch a     Tunja     Barbosa     Bogot
Distancia total estimada: 1128.1 km

3. Ruta BALANCEADA (minimiza el costo combinado de tiempo + distancia):
Bogot      Tunja     Ch a     Tabio     La Vega     Fusagasug     Girardot     Ibagu     Barbosa     Bogot
Costo combinado estimado: 2608.3 (minutos + km)
```