TP 2: Heurísticas para el Problema de Ruteo de Vehículos con Capacidad (CVRP)

Integrantes:
Camus Sol
Luchetti Olivia
Martinez Kiara

22 de junio de 2025

Índice

1. Introducción

El Problema de Ruteo de Vehículos con Capacidad (CVRP) es un problema de optimización combinatoria que busca determinar las rutas óptimas para una flota de vehículos de capacidad limitada para servir a un conjunto de clientes desde un depósito central, minimizando el costo total (generalmente la distancia).

1.1. Definición Formal

Dado un grafo $G = (V, E)$, donde $V = \{0, ..., n\}$ es el conjunto de nodos (0 es el depósito) y E es el conjunto de aristas. Cada cliente $i \in V \setminus \{0\}$ tiene una demanda q_i, y la distancia entre nodos i y j es d_{ij}. El objetivo es encontrar un conjunto de rutas de mínimo costo total para una flota de vehículos de capacidad Q, tal que cada cliente sea visitado exactamente una vez.

2. Heurísticas Constructivas Implementadas

2.1. Algoritmo de Ahorros de Clarke & Wright

Es una de las heurísticas más conocidas. Parte de una solución inicial donde cada cliente es servido individualmente y luego fusiona rutas de forma iterativa si esto genera un .[a]horro.[en] la distancia total.

2.1.1. Concepto de Ahorros

El ahorro s_{ij} al conectar los clientes i y j se calcula como: $s_{ij} = d_{0i} + d_{0j} - d_{ij}$. Representa la distancia que se ahorra al no tener que volver al depósito entre las visitas a i y j.

2.1.2. Análisis de Complejidad

- **Cálculo de ahorros:** Se calculan para cada par de nodos, resultando en una complejidad de $O(n^2)$.
- **Ordenamiento de ahorros:** Ordenar la lista de ahorros tiene una complejidad de $O(n^2 \log n)$.
- **Construcción de rutas:** El bucle principal itera sobre la lista de ahorros. Con estructuras de datos optimizadas (como un mapa para localizar nodos), las operaciones internas son rápidas. La complejidad total está dominada por el ordenamiento: $O(n^2 \log n)$.

2.2. Algoritmo del Vecino Más Cercano (Nearest Neighbor)

Es una heurística voraz que construye rutas secuencialmente. Desde el último nodo añadido a una ruta, se selecciona el cliente no visitado más cercano que no viole la restricción de capacidad.

2.2.1. Variante Implementada

Nuestra implementación evalúa una lista restringida de los 5 candidatos más cercanos para añadir a la ruta, en lugar de solo el más cercano, para diversificar la construcción.

2.2.2. Análisis de Complejidad

En cada paso de la construcción de una ruta, se busca el vecino más cercano entre los nodos no visitados. En el peor caso, esto se repite para cada uno de los n nodos, resultando en una complejidad de $O(n^2)$.

3. Operadores de Búsqueda Local (Pendiente)

En esta sección se describirán los operadores de búsqueda local implementados para mejorar las soluciones obtenidas por las heurísticas constructivas.

3.1. Operador de Reubicación (Relocation) (Pendiente)

Descripción del operador que mueve un cliente de una ruta a otra.

3.2. Operador de Intercambio (Swap) (Pendiente)

Descripción del operador que intercambia dos clientes entre rutas (o en la misma ruta).

4. Métodos Híbridos (Pendiente)

En esta sección se propondrá y describirá un método que combine una heurística constructiva con los operadores de búsqueda local para refinar las soluciones iniciales.

5. Diseño e Implementación

5.1. Lenguaje y Estructura

El proyecto fue implementado en C++11, favoreciendo la eficiencia en el manejo de memoria y el rendimiento computacional. Se utilizó un diseño orientado a objetos para modularizar el código y facilitar su mantenimiento.

5.2. Clases Principales

- **VRPLIBReader**: Responsable de leer y parsear las instancias desde archivos de formato VRPLIB.
- **Solution**: Modela una solución al problema, conteniendo el conjunto de rutas y sus métricas (costo, etc.).
- **ClarkeWrightSolver** y **NearestNeighborSolver**: Implementan las lógicas de las heurísticas constructivas.

5.3. Casos de Test (Pendiente)

En esta sección se documentarán los casos de test unitarios implementados para al menos dos clases, justificando las pruebas realizadas. Se pueden utilizar herramientas de IA para el diseño de los tests.

6. Experimentación y Resultados

6.1. Diseño Experimental

- **Instancias:** Se utilizan las instancias de benchmark del directorio '2l-cvrp-0' de VRPLIB.
- **Métricas:** Se evalúan las soluciones según el costo total (distancia) y el número de vehículos (rutas). También se considera el tiempo de ejecución.

6.2. Plan de Comparación de Métodos

La experimentación comparará el rendimiento de (al menos) los siguientes enfoques:

1. Cada heurística constructiva de manera independiente (realizado).
2. Heurística constructiva + un operador de búsqueda local (pendiente).
3. Heurística constructiva + combinación de operadores de búsqueda local (pendiente).

6.3. Resultados Preliminares (Heurísticas Constructivas)

- Para la instancia **E045-04f**:
  - **Clarke & Wright:** Costo = 834.34, Rutas = 5.
  - **Nearest Neighbor:** Costo = 1075.33, Rutas = 4.
  - **Referencia:** Costo = 723.54, Rutas = 4.
- **Observación:** Clarke & Wright obtiene un mejor costo, mientras que Nearest Neighbor utiliza un número de vehículos más cercano al óptimo en este caso.

6.4. Análisis de Resultados Finales (Pendiente)

En esta sección se presentará un análisis exhaustivo de los resultados de todos los métodos propuestos, utilizando tablas y gráficos para comparar su rendimiento en el conjunto de instancias.

7. Conclusiones

7.1. Hallazgos Preliminares

- Las heurísticas constructivas proveen una base sólida, pero sus resultados pueden ser mejorados significativamente.

- Existe un trade-off entre la calidad de la solución (costo) y el número de vehículos utilizados, donde diferentes heurísticas favorecen uno u otro.

7.2. Posibles Mejoras y Trabajo Futuro

- **Búsqueda Local:** La implementación de operadores de búsqueda local es el siguiente paso crucial.
- **Randomización y Metaheurísticas:** Para obtener la nota máxima, se podría implementar una estrategia de randomización en las heurísticas constructivas o una metaheurística como Simulated Annealing o Búsqueda Tabú para guiar la búsqueda local.
- **Paralelización:** Para instancias de gran tamaño, el cálculo de la matriz de ahorros podría ser paralelizado.