DISEÑO Y ANÁLISIS DE ALGORITMOS

Algoritmos constructivos y búsquedas por entornos

Vehicle Routing Problem with Transshipments for Solid Waste Collection with Transfer Stations (VRPT-SWTS)

Belén Melián Batista
Curso 2024-2025

Objetivo:
Proponer, implementar y evaluar algoritmos constructivos y búsquedas por entornos para el *Vehicle Routing Problem with Transshipments for Solid Waste Collection with Transfer Stations (VRPT-SWTS)*.

Tareas:
Además de las tareas descritas en el presente documento, los estudiantes tendrán que realizar las modificaciones que se planteen durante la defensa final de la práctica.

Entregas parciales y modificación final:

- Entrega parcial 1 (25 de marzo - 1 de abril): Voraz y fase constructiva de GRASP.
- Entrega parcial 2 (8 de abril): GRASP con búsquedas locales.
- Entrega final y defensa (22 de abril): Práctica completa (incluye GVNS) e informe en Latex (Overleaf).

Evaluaciones parciales:
Código fuente y defensa del trabajo realizado: hasta 10 puntos; si el día de la corrección falta algún código o este es incorrecto, la práctica se calificará como No Apta.

Evaluación final:
Código fuente y memoria: hasta 4 puntos; si el día de la corrección falta algún código o este es incorrecto, la práctica se calificará como No apta.

Modificación propuesta el día de la corrección: hasta 4 puntos. Será necesario realizar la modificación para superar la práctica.

Defensa oral de la práctica: hasta 2 puntos.

Lenguaje de programación:
Java o C++.

Ponderación de notas de las tres entregas:
Las defensas parciales supondrán un 25 % de la nota final de la práctica, 12,5 % cada una de ellas.

Vehicle Routing Problem with Transshipments for Solid Waste Collection with Transfer Stations (VRPT-SWTS)

En esta práctica estudiaremos un Problema de Rutas de Vehículos con Trasbordos para la recogida de residuos sólidos con estaciones de transferencia (VRPT-SWTS) Ghiani et al., 2021, que trata sobre la optimización de la recolección de residuos sólidos urbanos considerando estaciones de transferencia intermedias (*Solid Waste Transfer*

Stations, SWTS). Debido a regulaciones ambientales más estrictas y al aumento de la distancia de los vertederos con respecto a los centros urbanos, se ha incentivado el uso de estas estaciones, donde los residuos son transferidos de vehículos de recolección pequeños (*Collection Vehicles*, CV) a vehículos de transporte de mayor capacidad (*Transportation Vehicles*, TV).

El problema se modela como un *Vehicle Routing Problem with Transshipment* (VRPT), en el que se deben determinar las rutas de los vehículos de recolección y transporte, así como su sincronización en las estaciones de transferencia. Se descompone en dos fases (ver Figura 1):

- **Fase de recolección:** Se deben diseñar las rutas de los vehículos de recolección pequeños (CV), asegurando que visiten los puntos de recolección asignados y transfieran su carga en una estación SWTS antes de regresar al depósito. Cada ruta puede constar de múltiples trayectos, dependiendo del tiempo disponible y la capacidad del vehículo.
- **Fase de transporte:** Se determinan las rutas de los vehículos de transporte de mayor capacidad (TV), los cuales recogen los residuos de las estaciones SWTS y los transportan al vertedero. Estas rutas deben sincronizarse con la llegada de los vehículos de recolección.

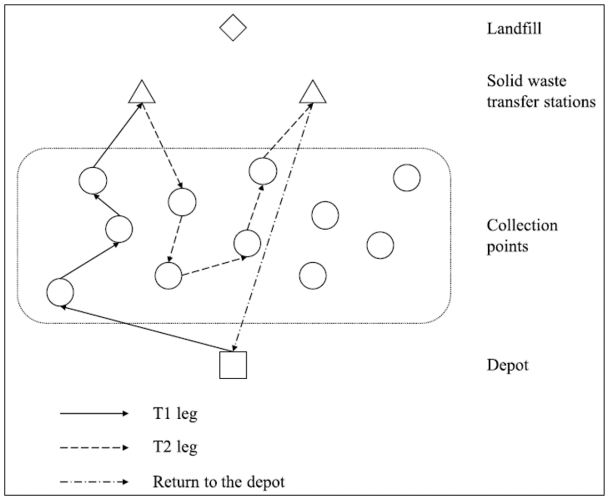

Figura 1: Representación gráfica del esqiema de gestión de los residuo sólidos

El problema introduce desafíos adicionales con respecto al *Vehicle Routing Problem* (VRP) clásico, debido a la necesidad de coordinación entre los diferentes tipos de vehículos y la gestión eficiente del transbordo de carga en las estaciones de transferencia. El problema consiste en diseñar las rutas para los vehículos de recolección de residuos y para los vehículos de transporte requeridos para realizar las tareas de recogida de residuos sólidos, con el **objetivo de minimizar el número de vehículos a utilizar**.

Para resolverlo, haremos uso de algoritmos heurísticos y metaheurísticos, como son los algoritmos voraces, GRASP, búsqueda locales y GVNS.

Un algoritmo voraz para construir las rutas de los vehículos de recolección (Algorithm 1)

En esta sección se muestra un algoritmo constructivo voraz que permite generar las rutas de recolección. Cuando el tamaño del problema se acerca a un caso real, la **complejidad del modelo matemático de optimización** hace que sea difícil obtener una solución óptima en un tiempo razonable. Para abordar esta limitación, se usa una **heurística constructiva voraz rápida**, cuyo pseudocódigo se presenta en Algorithm 1.

La idea principal del algoritmo es **construir las rutas de los vehículos agregando una zona de recolección a la vez**.

1. **Inicio de una nueva ruta**
    - Se crea una nueva ruta desde el **depósito**, asignándole la **máxima capacidad disponible** y el **tiempo máximo permitido** para operar.
2. **Asignación de zonas de recolección**
    - Mientras haya **zonas de recolección sin asignar**, se selecciona la zona más cercana a la última ubicación de la ruta.
    - Si agregar esta zona no **excede la capacidad del vehículo** ni el **tiempo máximo de ruta**, se **añade a la ruta**, se elimina del conjunto de zonas pendientes y se actualizan los valores de capacidad y tiempo restantes.
    - Si no es posible agregarla porque **el vehículo se llenó**, se dirige a la **estación de transferencia de residuos (SWTS) más cercana**, se vacía la carga y se actualizan los valores de capacidad y tiempo.
    - Si la zona no puede añadirse porque **no hay suficiente tiempo restante**, se **finaliza la ruta**.
3. **Finalización de la ruta**
    - Si la última parada del vehículo no fue una estación de transferencia, se añade la **más cercana** antes de regresar al depósito.
    - Se guarda la ruta y se asigna un **nuevo vehículo** para continuar con las zonas restantes.
    - Este proceso **se repite hasta que todas las zonas hayan sido atendidas**.

El **objetivo de esta heurística** es construir rutas eficientes sin necesidad de resolver el problema de forma exacta, permitiendo encontrar soluciones en **menos tiempo**.

Un algoritmo constructivo para generar las rutas de los vehículos de transporte (Algorithm 2)

Los vehículos de transporte inician su ruta en el vertedero, visitan una o más SWTS para recoger residuos de los vehículos de recolección y regresan al vertedero para vaciar su carga. Las rutas de los vehículos de recolección determinan un conjunto de tareas

H para los vehículos de transporte, donde cada tarea $h \in H$ se representa como una tripleta:

$$h = (D_h, s_h, \tau_h) \tag{1}$$

Donde:

- D_h: Cantidad de residuos transportados por el vehículo de recolección en la SWTS s_h.
- s_h: SWTS visitada por el vehículo de recolección.
- τ_h: Tiempo en el que el vehículo de recolección llega a la SWTS.

Una tarea h solo puede ser asignada a un vehículo de transporte $e \in E$ si se cumplen las siguientes condiciones:

(a) Si la ruta R_e ya tiene tareas asignadas y h' es la última tarea asignada a e, entonces el tiempo de viaje desde $s_{h'}$ hasta s_h debe ser menor o igual a la diferencia $\tau_h - \tau_{h'}$.

(b) La capacidad restante del vehículo q_e debe ser mayor o igual a la cantidad de residuos D_h.

(c) La duración total de la ruta R_e, considerando el regreso al vertedero, no debe exceder un umbral L'.

El algoritmo 2 sigue los siguientes pasos:

1. Ordenar las tareas H en orden creciente según τ_h.
2. Inicializar un conjunto vacío de vehículos de transporte E.
3. Para cada tarea $h \in H$:
   a) Buscar el vehículo disponible que minimice el costo de inserción cumpliendo las restricciones (a)-(c).
   b) Si no hay un vehículo disponible, crear un nuevo vehículo.
   c) Agregar la tarea a la ruta del vehículo seleccionado.
   d) Si la capacidad restante del vehículo cae por debajo de un umbral mínimo, programar un viaje al vertedero antes de asignarle nuevas tareas.
4. Asegurar que todas las rutas terminan en el vertedero.

Este método garantiza que los vehículos de transporte operen de manera eficiente y estén sincronizados con los vehículos de recolección, optimizando la logística del sistema de gestión de residuos.

Algoritmo para selecciona el mejor vehículo para una tarea (Algorithm 3)

Esta sección muestra en el Algoritmo 3 el procedimiento utilizado para determinar el vehículo de transporte al que se asignará una tarea.

Implementación

Las instancias del problema se suministrarán en un fichero de texto con el formato mostrado en las instancias proporcionadas.

Tareas

a) Diseñar e implementar el algoritmo constructivo voraz descrito en los Algoritmos 1, 2 y 3.

b) Diseñar e implementar un algoritmo GRASP para el problema. Implementar sólo la fase constructiva. La fase de mejora se definirá e implementará en el punto c).

c) Diseñar e implementar un Método Multiarranque para el problema. Para ello se deberán definir las estructuras de entorno correspondientes a los siguientes 4 movimientos para las rutas de recolección: reinserción de tareas, intercambio de tareas y 2-opt.

d) Diseñar e implementar una Búsqueda por Entorno Variable General para el problema (GVNS).

Qué debe presentar el alumno

a) Código fuente, debidamente comentado, y fichero ejecutable.

b) Una memoria en Latex en la que se describan brevemente los algoritmos diseñados enumerando las estructuras de datos usadas, las estructuras de entorno empleadas en las correspondientes búsquedas y cualquier elemento necesario para comprender el diseño propuesto. **Nota.- Al aula virtual se debe subir un fichero pdf, en el que aparezca claramente el enlace de Overleaf a la memoria en Latex**.

c) Tablas o gráficas de resultados que muestren el comportamiento de los algoritmos sobre diferentes instancias del problema. A continuación se muestra un modelo de tablas de resultados que el alumno puede usar. No obstante, se trata solo de un modelo que puede modificarse si se cree que otro es mejor.

Algoritmo Voraz

Instance	*#Zonas*	*#CV*	*#TV*	*CPU_Time*
*instance*1				
*instance*2				
*instance*3				
*instance*4				
*instance*5				
...	...	...	...	...
*instance*20				
average				

Cuadro 1: Algoritmo voraz. Tabla de resultados

Algoritmo GRASP

Instance	*#Zonas*	*|LRC|*	*Ejecucin*	*#CV*	*#TV*	*CPU_Time*
*instance*1	20	2	1			
*instance*1	20	2	2			
*instance*1	20	2	3			
*instance*1	20	3	1			
*instance*1	20	3	2			
*instance*1	20	3	3			
*instance*2						
*instance*2						
*instance*2						
...	...	...	...	...	...	...
*instance*20						
average						

Cuadro 2: Algoritmo GRASP. Tabla de resultados

Algoritmo GVNS

Instance	*#Zonas*	k_{max}	*Ejecución*	*#CV*	*#TV*	*CPU_Time*
*instance*1	20	2	1			
*instance*1	20	2	2			
*instance*1	20	2	3			
*instance*1	20	3	1			
*instance*1	20	3	2			
*instance*1	20	3	3			
*instance*2						
*instance*2						
*instance*2						
...	...	...	...	...	...	...
*instance*20						
average						

Cuadro 3: Algoritmo GVNS. Tabla de resultados

Algorithm 1 A Constructive Heuristic for Collection Vehicle Routes

Require: The set of collection zones C, the waste quantity d_c for each $c \in C$, vehicle capacity Q, maximum route duration L, the set of SWTS S.
Ensure: A set K of collection vehicles.

$K \leftarrow \emptyset$
$k \leftarrow 1$
while $C \neq \emptyset$ **do**
    $R_k \leftarrow$ {depot} ▷ Initialize the route for vehicle k
    $q_k \leftarrow Q$ ▷ Residual vehicle capacity
    $T_k \leftarrow L$ ▷ Residual route time
    while true **do**
        $c' \leftarrow$ Closest collection zone to last(R_k) in C
        $t' \leftarrow$ time to visit c', a SWTS, and the depot
        if $d_{c'} \leq q_k$ and $t' \leq T_k$ **then**
            $R_k \leftarrow R_k \cup \{c'\}$
            $q_k \leftarrow q_k - d_{c'}$
            Update T_k
            $C \leftarrow C \setminus \{c'\}$
        else
            if $t' \leq T_k$ **then**
                $s' \leftarrow$ Closest SWTS to c'
                $R_k \leftarrow R_k \cup \{s'\}$
                $q_k \leftarrow Q$
                Update T_k
            else
                break
            end if
        end if
    end while
    if last(R_k) $\notin S$ **then**
        $s' \leftarrow$ Closest SWTS to last(R_k)
        $R_k \leftarrow R_k \cup \{s'\} \cup$ {depot}
    else
        $R_k \leftarrow R_k \cup$ {depot}
    end if
    $K \leftarrow K \cup \{R_k\}$
    $k \leftarrow k + 1$
end while
return K

Algorithm 2 A Constructive Heuristic for Transportation Vehicle Routes

Require: The set of tasks H, transportation vehicle capacity Q', maximum route duration L'.
Ensure: A set E of transportation vehicles.

Sort H in ascending order of arrival time τ_h.
$E \leftarrow \emptyset$
$q_{\text{mín}} \leftarrow \text{mín}\{D_h : h \in H\}$ ▷ Minimum waste amount
while $H \neq \emptyset$ **do**
    $h \leftarrow$ first element in H
    $H \leftarrow H \setminus \{h\}$
    $e \leftarrow$ ChooseVehicle(E, h)
    if $e =$ null **then**
        $e \leftarrow |E| + 1$
        $R_e \leftarrow \{\text{landfill}, s_h\}$
        $q_e \leftarrow Q' - D_h$
        Update T_e ($T_e \leftarrow L' -$ {time to go from the landfill to s_h})
        $E \leftarrow E \cup \{e\}$
    else
        $R_e \leftarrow R_e \cup \{s_h\}$
        $q_e \leftarrow q_e - D_h$
        Update T_e
        if $q_e < q_{\text{mín}}$ **then**
            $R_e \leftarrow R_e \cup \{\text{landfill}\}$
            $q_e \leftarrow Q'$
        end if
    end if
end while
for $e \in E$ **do**
    if last$(R_e) \neq$ landfill **then**
        $R_e \leftarrow R_e \cup \{\text{landfill}\}$
    end if
end for
return E

Algorithm 3 Selecting the Best Vehicle for a Task

Require: Set of vehicles E, task h.
Ensure: Vehicle e to which the task is assigned.

$e \leftarrow$ null
bestInsertionCost $\leftarrow +\infty$
for $e' \in E$ **do**
    cost $\leftarrow$ compute best insertion satisfying conditions (a)-(c)
    if *cost* $<$ *bestInsertionCost* **then**
        $e \leftarrow e'$
        bestInsertionCost $\leftarrow$ *cost*
    end if
end for
return e

Bibliografía

Ghiani, Gianpaolo et al. (2021). "Optimizing a waste collection system with solid waste transfer stations". En: *Computers Industrial Engineering* 161, pág. 107618. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2021.107618.