



**Escuela Superior
de Ingeniería y Tecnología**
Universidad de La Laguna

Informe de la Práctica 5 de DAA Curso 2024-2025

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

NOMBRE ESTUDIANTE

La Laguna, 19 de marzo de 2025



Figura 1: DALL-E

La Figura 1 muestra una imagen creada por DALL-E, tratando de recrear el VRPT-SWTS abordado en esta práctica.

Agradecimientos

Agradezco a ...

Licencia

© Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial 4.0 Internacional.

Resumen

Resumen del trabajo realizado

Palabras clave: Waste collection, VRPT-SWTS, GRASP, LS, VNS.

Índice general

1. Introducción	1
1.1. Contexto	1
1.1.1. Objetivos	1
1.2. Motivación	1
2. Vehicle Routing Problem with Transshipments for Solid Waste Collection with Transfer Stations (VRPT-SWTS)	2
2.1. Descripción	2
2.1.1. Representación de las soluciones	3
3. Algoritmos	4
3.1. Un algoritmo voraz para construir las rutas de los vehículos de recolección (Algorithm 1)	4
3.2. Un algoritmo constructivo para generar las rutas de los vehículos de transporte (Algorithm 2)	5
3.3. Algoritmo para selecciona el mejor vehículo para una tarea (Algorithm 3) . .	6
4. Experimentos y resultados computacionales	9
4.1. Constructivo voraz	9
4.2. GRASP	9
4.3. GVNS	9
4.4. Análisis compartativo entre algoritmos	9
5. Conclusiones y trabajo futuro	11

Índice de Figuras

1.	DALL-E	2
2.1.	Representación gráfica del esqiema de gestión de los residuo sólidos	3

Índice de Tablas

4.1. Algoritmo voraz. Tabla de resultados	9
4.2. Algoritmo GRASP. Tabla de resultados	10
4.3. Algoritmo GVNS. Tabla de resultados	10

Capítulo 1

Introducción

1.1. Contexto

Contexto y objetivos

1.1.1. Objetivos

Durante el desarrollo de esta práctica, se han cumplido los objetivos listados a continuación:

- Obj1
- Obj2.
- Obj3.

1.2. Motivación

Capítulo 2

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

2.1. Descripción

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 2.1):

- **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.

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

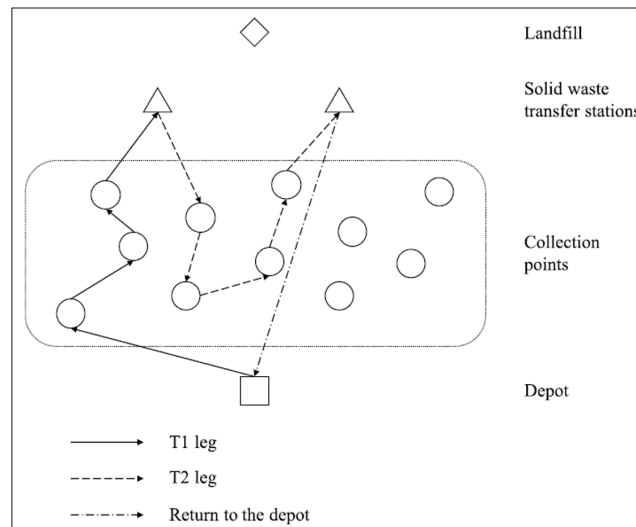


Figura 2.1: Representación gráfica del esquema de gestión de los residuo sólidos

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.

2.1.1. Representación de las soluciones

Escoge la estructura de datos que consideres más adecuada.

Capítulo 3

Algoritmos

3.1. 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**.

3.2. 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:

$$h = (D_h, s_h, \tau_h) \quad (3.1)$$

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:

- 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 $\tau_h - \tau_{h'}$.
- La capacidad restante del vehículo q_e debe ser mayor o igual a la cantidad de residuos D_h .
- 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.

3.3. 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.

Algoritmo 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.

```
1:  $K \leftarrow \emptyset$ 
2:  $k \leftarrow 1$ 
3: while  $C \neq \emptyset$  do
4:    $R_k \leftarrow \{\text{depot}\}$ 
5:    $q_k \leftarrow Q$ 
6:    $T_k \leftarrow L$ 
7:   while true do
8:      $c' \leftarrow$  Closest collection zone to last( $R_k$ ) in  $C$ 
9:      $t' \leftarrow$  time to visit  $c'$ , a SWTS, and the depot
10:    if  $d_{c'} \leq q_k$  and  $t' \leq T_k$  then
11:       $R_k \leftarrow R_k \cup \{c'\}$ 
12:       $q_k \leftarrow q_k - d_{c'}$ 
13:      Update  $T_k$ 
14:       $C \leftarrow C \setminus \{c'\}$ 
15:    else
16:      if  $t' \leq T_k$  then
17:         $s' \leftarrow$  Closest SWTS to  $c'$ 
18:         $R_k \leftarrow R_k \cup \{s'\}$ 
19:         $q_k \leftarrow Q$ 
20:        Update  $T_k$ 
21:      else
22:        break
23:      end if
24:    end if
25:  end while
26:  if last( $R_k$ )  $\notin S$  then
27:     $s' \leftarrow$  Closest SWTS to last( $R_k$ )
28:     $R_k \leftarrow R_k \cup \{s'\} \cup \{\text{depot}\}$ 
29:  else
30:     $R_k \leftarrow R_k \cup \{\text{depot}\}$ 
31:  end if
32:   $K \leftarrow K \cup \{R_k\}$ 
33:   $k \leftarrow k + 1$ 
34: end while
35: return  $K$ 
```

- ▷ Initialize the route for vehicle k
- ▷ Residual vehicle capacity
- ▷ Residual route time

Algoritmo 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.

```
1: Sort  $H$  in ascending order of arrival time  $\tau_h$ .
2:  $E \leftarrow \emptyset$ 
3:  $q_{\min} \leftarrow \min\{D_h : h \in H\}$  ▷ Minimum waste amount
4: while  $H \neq \emptyset$  do
5:    $h \leftarrow$  first element in  $H$ 
6:    $H \leftarrow H \setminus \{h\}$ 
7:    $e \leftarrow \text{ChooseVehicle}(E, h)$ 
8:   if  $e = \text{null}$  then
9:      $e \leftarrow |E| + 1$ 
10:     $R_e \leftarrow \{\text{landfill}, s_h\}$ 
11:     $q_e \leftarrow Q' - D_h$ 
12:    Update  $T_e$  ( $T_e \leftarrow L' - \{\text{time to go from the landfill to } s_h\}$ )
13:     $E \leftarrow E \cup \{e\}$ 
14:   else
15:     $R_e \leftarrow R_e \cup \{s_h\}$ 
16:     $q_e \leftarrow q_e - D_h$ 
17:    Update  $T_e$ 
18:    if  $q_e < q_{\min}$  then
19:       $R_e \leftarrow R_e \cup \{\text{landfill}\}$ 
20:       $q_e \leftarrow Q'$ 
21:    end if
22:   end if
23: end while
24: for  $e \in E$  do
25:   if  $\text{last}(R_e) \neq \text{landfill}$  then
26:      $R_e \leftarrow R_e \cup \{\text{landfill}\}$ 
27:   end if
28: end for
29: return  $E$ 
```

Algoritmo 3 Selecting the Best Vehicle for a Task

Require: Set of vehicles E , task h .

Ensure: Vehicle e to which the task is assigned.

```
1:  $e \leftarrow \text{null}$ 
2:  $\text{bestInsertionCost} \leftarrow +\infty$ 
3: for  $e' \in E$  do
4:    $\text{cost} \leftarrow$  compute best insertion satisfying conditions (a)-(c)
5:   if  $\text{cost} < \text{bestInsertionCost}$  then
6:      $e \leftarrow e'$ 
7:      $\text{bestInsertionCost} \leftarrow \text{cost}$ 
8:   end if
9: end for
10: return  $e$ 
```

Capítulo 4

Experimentos y resultados computacionales

4.1. Constructivo voraz

Algoritmo Voraz				
<i>Instance</i>	<i>#Zonas</i>	<i>#CV</i>	<i>#TV</i>	<i>CPU_Time</i>
<i>instance1</i>				
<i>instance2</i>				
<i>instance3</i>				
<i>instance4</i>				
<i>instance5</i>				
...
<i>instance20</i>				
<i>average</i>				

Cuadro 4.1: Algoritmo voraz. Tabla de resultados

4.2. GRASP

Comparativa probando diferentes búsquedas locales. Debe incluir el diseño de experimentos para el ajuste de parámetros y el análisis de los resultados obtenidos.

4.3. GVNS

Debe incluir el diseño de experimentos para el ajuste de parámetros y el análisis de los resultados obtenidos

4.4. Análisis compartativo entre algoritmos

Realiza un análisis comparativo entre los diferentes algoritmos.

Algoritmo GRASP						
<i>Instance</i>	<i>#Zonas</i>	<i> LRC </i>	<i>Ejecucin</i>	<i>#CV</i>	<i>#TV</i>	<i>CPU_Time</i>
<i>instance1</i>	20	2	1			
<i>instance1</i>	20	2	2			
<i>instance1</i>	20	2	3			
<i>instance1</i>	20	3	1			
<i>instance1</i>	20	3	2			
<i>instance1</i>	20	3	3			
<i>instance2</i>						
<i>instance2</i>						
<i>instance2</i>						
...
<i>instance20</i>						
<i>average</i>						

Cuadro 4.2: Algoritmo GRASP. Tabla de resultados

Algoritmo GVNS						
<i>Instance</i>	<i>#Zonas</i>	<i>k_{max}</i>	<i>Ejecución</i>	<i>#CV</i>	<i>#TV</i>	<i>CPU_Time</i>
<i>instance1</i>	20	2	1			
<i>instance1</i>	20	2	2			
<i>instance1</i>	20	2	3			
<i>instance1</i>	20	3	1			
<i>instance1</i>	20	3	2			
<i>instance1</i>	20	3	3			
<i>instance2</i>						
<i>instance2</i>						
<i>instance2</i>						
...
<i>instance20</i>						
<i>average</i>						

Cuadro 4.3: Algoritmo GVNS. Tabla de resultados

Capítulo 5

Conclusiones y trabajo futuro

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>.