

Nombre y Apellido:.....  
sección:

## EXAMEN I

**Problema 1.** Un ladrón roba una bicicleta y huye con ella a  $4 \text{ m/s}$ . Un ciclista que lo ve, sale detrás del mismo 1 minutos más tarde a  $6 \text{ m/s}$ .

- Diga distancia le saca de ventaja el ladrón en 1 minuto si se supone que parten ambos de donde estaba la bicicleta.
- Plantear la función de posición para cada cuerpo.
- ¿Al cabo de cuánto tiempo lo alcanzará?
- ¿En qué lugar lo alcanzará ?

### Problema 2

Supóngase que un día caluroso la temperatura es de  $34^\circ$  centígrados durante 16 horas y  $28^\circ$  centígrados durante 8 horas. ¿Cuál es la temperatura promedio para este día?

**Problema 3** Una clase ave realiza un viaje durante 5 horas, durante 3 horas viaja a  $60 \text{ km/h}$  y en las dos últimas horas viaja a  $20 \text{ km/h}$ . a) ¿Podrías decir cuál es la velocidad promedio? y b) ¿Diga cuánto se ha desplazado el ave?

**Problema 4.** Dos cuerpos A y B situados a 800 m de distancia salen en dirección opuesta simultáneamente a una velocidad de  $2 \text{ m/s}$  y  $-3 \text{ m/s}$  respectivamente, siendo la aceleración de A, de  $3 \text{ m/s}^2$ , mientras que B se mueve con MRU Calcular:

- tiempo que tardan en encontrarse, y b) sus velocidades en el momento del encuentro.

**Problema 5.** Un proyectil se dispara desde el extremo de un risco a 100 m sobre el nivel del suelo, con una rapidez inicial de  $5 \text{ m/s}$  y un ángulo de  $30^\circ$  con respecto a la horizontal.  
a) Determine el tiempo que le toma al proyectil golpear el suelo, b) Determine el alcance del proyectil medido desde la base del risco.

# Trabajo Fin de Grado

## Grado en Ingeniería de Tecnologías Industriales

Aplicación de multiagentes en Python para optimización de rutas.

Autor: Sergio Fuentes Moreno

Tutor: José Miguel León Blanco

Dpto. de Organización Industrial y Gestión de Empresas I  
Escuela Técnica Superior de Ingeniería  
Universidad de Sevilla

Sevilla, 2021





Trabajo Fin de Grado  
Grado en Ingeniería de Tecnologías Industriales

# **Aplicación de multiagentes en Python para optimización de rutas.**

Autor:

Sergio Fuentes Moreno

Tutor:

José Miguel León Blanco

Profesor Contratado Doctor

Dpto. de Organización Industrial y Gestión de Empresas I  
Escuela Técnica Superior de Ingeniería  
Universidad de Sevilla  
Sevilla, 2021



Trabajo Fin de Grado: Aplicación de multiagentes en Python para optimización de rutas.

Autor: Sergio Fuentes Moreno

Tutor: José Miguel León Blanco

El tribunal nombrado para juzgar el Trabajo arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

*A mi familia  
A mis maestros*

# Agradecimientos

---

Agradecido por estos años vividos en la escuela donde he conocido a grandes amigos. Amigos que me han acompañado a lo largo de la carrera, compartiendo trabajos, aprobados y suspensos, personas que me han acompañado en momentos de alegría y de tristeza.

Agredecer también a aquellos profesores que me han ayudado, tanto de la escuela como de la academia, profesores que sin ellos no estaría aquí redactando un trabajo de fin de grado. Sobre todo, a Javi de informática, gracias a él entendí la programación y me gustó hasta el punto de que, aquí estoy, realizando un trabajo que lleva por detrás un programa escrito por mí.

También quería agradecer a mi tutor José Miguel León Blanco por ayudarme a realizar este trabajo, por darme siempre ánimos, prestarme la ayuda necesaria y darme la oportunidad de realizar un trabajo que realmente me gusta. También darle las gracias por ayudarme en otras asignaturas en las que hemos coincidido a lo largo de la carrera.

Por último, agredecer sobre todo a mis padres Antonio Fuentes e Isabel Moreno, que me han animado a lo largo de la carrera. También agradecer a mi padre por intentar ayudarme con el trabajo, explicándome términos sobre la logística, el e-commerce y hablar de del momento actual que esta viviendo el sector del transporte.

*Sergio Fuentes Moreno*

# **Resumen**

---

El trabajo consiste en la aplicación de un sistema multiagente para la optimización de rutas, en este caso un problema de rutado de vehículos con ventanas temporales. La primera parte es teórica, en ella se describen diferentes conceptos que son imprescindibles para la realización del trabajo.

El resto del trabajo describe cómo se ha realizado el sistema de multiagentes y qué algoritmos se utilizan. Finalmente se realizan numerosas experimentaciones con diversas instancias para poder sacar conclusiones al respecto y presentar una serie de resultados.

# **Abstract**

---

The work consists of the application of a multiagent system for route optimization, in this case a vehicle routing problem with time windows. The first part is theoretical, it describes different concepts that are essential to the work.

The rest of the work describes how the multiagent system has been made and what algorithms are used. Finally, many experiments are carried out with differents instances in order to get conclusions and offer some results.

# Índice

---

<b>Agradecimientos</b>	<b>vii</b>
<b>Resumen</b>	<b>viii</b>
<b>Abstract</b>	<b>ix</b>
<b>Índice</b>	<b>x</b>
<b>Índice de Tablas</b>	<b>xii</b>
<b>Índice de Figuras</b>	<b>xiii</b>
<b>Notación</b>	<b>xiv</b>
<b>1 Objeto y alcance</b>	<b>1</b>
<b>2 Introducción</b>	<b>2</b>
2.1 <i>Variantes del VRP</i>	2
2.2 <i>VRPTW</i>	2
<b>3 Revisión de literatura</b>	<b>4</b>
3.1 <i>Aplicación de sistemas multiagentes a un problema VRP.</i>	4
3.1.1 Heurísticas.	5
3.1.2 Métodos para obtener la población inicial de soluciones.	5
3.1.3 Heurísticas de búsqueda local.	5
3.1.4 Metaheurísticas.	6
3.2 <i>Programas considerados para programar el problema.</i>	7
3.2.1 AnyLogic	7
3.2.2 Python	8
3.3 <i>Librerías/entornos para programar sistemas multiagentes en Python</i>	8
3.3.1 Librería Spade	8
3.3.2 Librería Aima [29]	8
3.3.3 Librería Mesa [30]	9
3.3.4 Resumen de librerías analizadas	9
3.3.5 Elección final	10
<b>4 Aplicación práctica</b>	<b>11</b>
4.1 <i>Datos del problema VRPTW</i>	11
4.2 <i>Población inicial de soluciones</i>	12
4.3 <i>Agentes Optimizadores</i>	14
4.3.1 OA Vecinos	14
4.3.2 OA Buscar solitario	14
4.3.3 OA String Cross	15
4.3.4 OA Insertion	15
4.3.5 OA Intercambio	16
4.4 <i>Solution Manager y entorno</i>	16
<b>5 Experiencia computacional</b>	<b>18</b>
5.1 <i>Características del ordenador utilizado.</i>	18
5.2 <i>Programa y versión de Python</i>	18
5.3 <i>Criterios para la recogida de datos</i>	18

<b>6</b>	<b>Resultados y conclusiones</b>	<b>19</b>
6.1	<i>Obtención de resultados.</i>	19
6.2	<i>Resultados y conclusiones respecto a la población inicial de soluciones.</i>	29
6.3	<i>Resultados y conclusiones del porcentaje de mejora.</i>	31
6.4	<i>Resultados y conclusiones de la solución final.</i>	39
6.5	<i>Propuestas de mejora para el sistema multiagente.</i>	42
<b>7</b>	<b>Conclusiones generales</b>	<b>44</b>
	<b>Referencias</b>	<b>45</b>

# ÍNDICE DE TABLAS

---

Tabla 1 Comparativa librerías para programar multiagentes	9
Tabla 2 Resultados medios de la instancia C101.50	20
Tabla 3 Resultados medios de la instancia C102.50	20
Tabla 4 Resultados medios de la instancia C103.50	21
Tabla 5 Resultados medios de la instancia RC101.50	21
Tabla 6 Resultados medios de la instancia RC102.50	22
Tabla 7 Resultados medios de la instancia RC103.50	22
Tabla 8 Resultados medios de la instancia R201.50	23
Tabla 9 Resultados medios de la instancia R202.50	23
Tabla 10 Resultados medios de la instancia R205.50	24
Tabla 11 Resultados medios de la instancia C201.50	24
Tabla 12 Resultados medios de la instancia C202.50	25
Tabla 13 Resultados medios de la instancia C203.50	25
Tabla 14 Resultados medios de la instancia RC201.50	26
Tabla 15 Resultados medios de la instancia RC201.25	26
Tabla 16 Resultados medios de la instancia RC202.25	27
Tabla 17 Resultados medios de la instancia R101.50	27
Tabla 18 Resultados medios de la instancia R102.50	28
Tabla 19 Resultados medios de la instancia R103.50	28
Tabla 20 Comparación de las soluciones obtenidas con las óptimas encontradas.	39
Tabla 21 Soluciones iniciales con nuevo algoritmo, parte 1	42
Tabla 22 Soluciones iniciales con nuevo algoritmo, parte 2	43
Tabla 23 Comparación resultados del algoritmo 3 con el nuevo algoritmo, todos con el criterio “elección=aleatorio (1,3)”	43
Tabla 24 Comparación mejores resultados encontrados con el nuevo algoritmo, este último con el criterio “elección=aleatorio (1,3)”	43

# ÍNDICE DE FIGURAS

---

Ilustración 1 Diagrama de comunicación del proceso de resolución de JABAT [8]	4
Ilustración 2 Intercambio Or-Opt de 2 clientes	6
Ilustración 3 Interfaz de usuario de AnyLogic (help.anylogic.com)	7
Ilustración 4 Ejemplo de parte del código de Aima	9
Ilustración 5 Distribución de instancia C103_025	11
Ilustración 6 Distribución de instancia R101_050	11
Ilustración 7 Distribución de instancia RC101_100	12
Ilustración 8 Ejemplo de gráfica con dos óptimos y dos soluciones iniciales.	12
Ilustración 9 Esquema SolutionManager. Ejemplo de intercambio de información	17
Ilustración 10 Gráfico del valor de la población inicial para cada instancia.	29
Ilustración 11 Gráfico del número de veces de la población inicial en un puesto.	30
Ilustración 12 Gráfico que relaciona la solución inicial con la mejor solución final.	30
Ilustración 13 Porcentaje de mejora para las instancias de tipo C1	31
Ilustración 14 Porcentaje de mejora para las instancias de tipo RC1	32
Ilustración 15 Porcentaje de mejora para las instancias de tipo R2	32
Ilustración 16 Porcentaje de mejora para las instancias de tipo C2	33
Ilustración 17 Porcentaje de mejora para las instancias de tipo RC2	33
Ilustración 18 Porcentaje de mejora para las instancias de tipo R1	34
Ilustración 19 Porcentaje de mejora medio para cada instancia	34
Ilustración 20 Porcentaje de mejora según el criterio Elección=1	35
Ilustración 21 Porcentaje de mejora según el criterio Elección=2	35
Ilustración 22 Porcentaje de mejora según el criterio Elección=aleatorio (1,3)	36
Ilustración 23 Porcentaje de mejora medio según el criterio de elección	36
Ilustración 24 Porcentaje de mejora medio del tipo de instancia para el criterio Elección=1	37
Ilustración 25 Porcentaje de mejora medio del tipo de instancia para el criterio Elección=2	37
Ilustración 26 Porcentaje de mejora medio del tipo de instancia para el criterio Elección=aleatorio (1,3)	38
Ilustración 27 Representación gráfica de la solución inicial 2 para la instancia C103.50	40
Ilustración 28 Representación gráfica de la solución final 2 para la instancia C103.50	40
Ilustración 29 Representación gráfica de la solución final 1 para la instancia C103.50	41

# Notación

---

c.q.d.	Como queríamos demostrar
e.o.c.	En cualquier otro caso
OA	Agentes Optimizadores
sa	Sujeto a
Min	Minimizar
max	Máximo
min	Mínimo
um	Unidades monetarias

# 1 OBJETO Y ALCANCE

---

Este 2020 y 2021 han sido especiales o diferentes respecto a otros años debido a la situación de pandemia en la que nos encontramos. La aparición de un nuevo virus a finales del año 2019 ha provocado la caída de la mayoría de los sectores comerciales. Sin embargo, el sector del transporte es aún más importante de lo que ya era antes, pues debido a esta pandemia, han aumentado las ventas en el canal e-commerce (comercio electrónico). Este canal de ventas ha aumentado en una gran cantidad de países, incluso en alguno de ellos como Turquía y México el aumento ha sido superior al 100% según informe del BBVA [1]. Este aumento obliga a las empresas de transporte a mejorar su modelo de distribución para poder mantener una cierta rentabilidad.

El objetivo es desarrollar un sistema multiagente capaz de abordar un problema de rutado de vehículos con ventanas de tiempo. Este tipo de programas no suelen utilizarse en problemas de optimización ya que, dotar de cierta inteligencia a un programa supone aumentar el tiempo de computación requerido. Sin embargo, los métodos de resolución exactos que existen para resolver el problema que se quiere abordar, tienen muchas limitaciones. Estos motivos y los mencionados en el apartado anterior, nos motivan a realizar el trabajo. Al final, se llevará a cabo un análisis de los resultados obtenidos, resultados que se compararán con las mejores soluciones encontradas para esos problemas.

## 2 INTRODUCCIÓN

---

Antes de determinar cuál es el problema que se quiere resolver, se debe conocer los distintos tipos de problemas de enrutamientos de vehículos llamados VRP (Vehicle Routing Problem) que existen, pero sin entrar en detalles. El problema de enrutamiento de vehículos es una generalización del problema del viajante (TSP, Travelling Salesman Problem [2]). La primera investigación sobre los problemas VRP fue en 1959 en un artículo de George Dantiz y John Ramser [3]: “El artículo trata de encontrar la ruta óptima de una flota de camiones de suministro de gasolina entre un terminal de granel y un largo número de estaciones de servicio suministrados por el terminal”. Para solucionar este tipo de problemas se proponen distintas heurísticas y metaurísticas, pero en este proyecto se aplicará un sistema multiagentes (se entrará en detalle más adelante) para poder resolver el problema.

### 2.1 Variantes del VRP

Algunas de las variantes del problema VRP son las siguientes [4]:

- Problema de enrutamiento de vehículos con recogida y entrega (VRPPD): el vehículo tiene que ir hasta un punto de recogida para llevar la mercancía a un punto de entrega.
- Problema de enrutamiento de vehículos con ventanas temporales (VRPTW): las entregas se realizan en una ventana temporal específica.
- Problema de enrutamiento de vehículos con capacidad (CVRP).
- Problema de enrutamiento de vehículos con viajes múltiples (VRPMT): un vehículo puede realizar más de una ruta o viaje.
- Problema de enrutamiento de vehículos abiertos (OVRP): los vehículos no tienen por qué terminar en el depósito.
- Problema de enrutamiento de vehículos de flota mixta (MFVRP): en este caso no existe un solo tipo de vehículo, por tanto, la capacidad y el consumo de energía será diferente para cada tipo.
- Problema de enrutamiento de vehículos con múltiples depósitos (MDVRPR).
- Problema de enrutamiento de vehículos periódicos (PVRP): las entregas se realizan en distintos días.
- Problema de enrutamiento de vehículos con entrega dividida (SDVRP): la mercancía de un cliente puede dividirse en varios vehículos.

### 2.2 VRPTW

El problema por resolver es el VRP con ventanas temporales o VRPTW. En este apartado se definen las distintas variables, restricciones y datos de nuestro problema.

Para el problema, se propone utilizar el modelo propuesto por Cordone y Calvo en 2001 [5]:

Sea  $G = (N, A)$  un grafo dirigido donde  $N = \{0, 1, \dots, n\}$  son nodos y  $A = \{(i, j) : i, j \in N, i \neq j\}$  el conjunto de arcos. El nodo 0 representa el almacén (depot) y  $N' = \{1, \dots, n\}$  representa a los clientes o lugares a visitar. Cada arco  $(i, j)$  es asociado a un coste de viaje  $c_{ij} \geq 0$  y a un tiempo  $t_{ij} \geq 0$ . Cada nodo  $i$  tiene una demanda  $q_i$ , un tiempo de servicio  $s_i$  y una ventana de tiempo  $[e_i, l_i]$ . Todos los vehículos tienen la misma capacidad  $Q$  y el mismo coste  $h \geq 0$ . A continuación, se asume que  $c_{ij} = t_{ij}, \forall (i, j) \in A$  y que  $h$  es lo suficientemente alto para garantizar que la minimización del número de vehículos es el principal objetivo. La ventana de tiempo debe estrecharse estableciendo  $e_i = \max(e_0 + t_{0i}, e_i)$  y  $l_i = \min(l_0 - t_{0i}, l_i)$ .

Establecer  $x_{ij} = 1$  si se usa el arco  $(i, j)$  y 0 en caso contrario;  $p_i$  representa el comienzo del servicio en el nodo

$i$ ;  $y_i$  representa la carga del vehículo al abandonar el nodo  $i$ . La formulación matemática queda del siguiente modo:

$$\text{minimizar } \sum_{j \in N'} h x_{0j} + \sum_{(i,j) \in N'} c_{ij} x_{ij} \quad (1)$$

s. a:

$$\sum_{j \in N} x_{ij} = 1 \quad \forall i \in N' \quad (2)$$

$$\sum_{i \in N} x_{ij} = 1 \quad \forall j \in N' \quad (3)$$

$$si \quad x_{ij} = 1 \rightarrow p_i + s_i + t_{ij} \leq p_j \quad \forall (i,j) \in A \quad (4)$$

$$e_i \leq p_i \leq l_i \quad \forall i \in N' \quad (5)$$

$$si \quad x_{ij} = 1 \rightarrow y_i + q_j \leq y_j \quad \forall (i,j) \in A \quad (6)$$

$$q_i \leq y_i \leq Q \quad \forall i \in N' \quad (7)$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in A \quad (8)$$

Las restricciones (2) y (3) aseguran que cada cliente sea servido por una única ruta. Las ecuaciones (4) y (5) representan las restricciones de las ventanas de tiempo mientras que las ecuaciones (6) y (7) van referidas a la capacidad.

# 3 REVISIÓN DE LITERATURA

Para resolver el problema se aplica un sistema multiagente. La idea es aprovechar la capacidad de los sistemas de multiagentes de dividir un problema en problemas más pequeños, así como la comunicación entre los distintos agentes para acelerar la búsqueda de soluciones y en este caso para evitar óptimos locales. Jeffrey Wooldridge define a los sistemas multiagentes de la siguiente forma [6]: “Los sistemas multiagentes son sistemas compuestos por múltiples interacciones entre elementos, conocidos como agentes. Los agentes son sistemas con al menos dos importantes capacidades. La primera es que tengan cierta capacidad de realizar acciones de manera autónoma. La segunda, que puedan interactuar con otros agentes.” ¿Pero qué es un sistema? Según la Real Academia Española (RAE), un sistema tiene varias definiciones:

- Conjunto de reglas o principios sobre una materia racionalmente enlazados entre sí.
- Conjunto de cosas que relacionadas entre sí ordenadamente contribuyen a determinado objeto.

## 3.1 Aplicación de sistemas multiagentes a un problema VRP.

En los últimos años se han hecho estudios sobre como implementar un sistema multiagente para resolver un problema VRP. Una de las aproximaciones más exitosas es la del concepto de un equipo asíncrono (A-Team), un grupo de agentes que intentan evolucionar una serie de soluciones. Dariusz Barbucha y Piotr Jedrzejowicz [7] proponen usar “JADE-based A-Team llamado JABAT”. JADE por sus siglas en inglés significa Java Agent Development Framework.

JABAT usa un grupo de agentes, cada uno representa un algoritmo para solucionar el problema. Para evitar estancarse en un óptimo local, estas soluciones son mejoradas por otros agentes independientes y así tener mayor posibilidad de encontrar el óptimo global [3] [8].

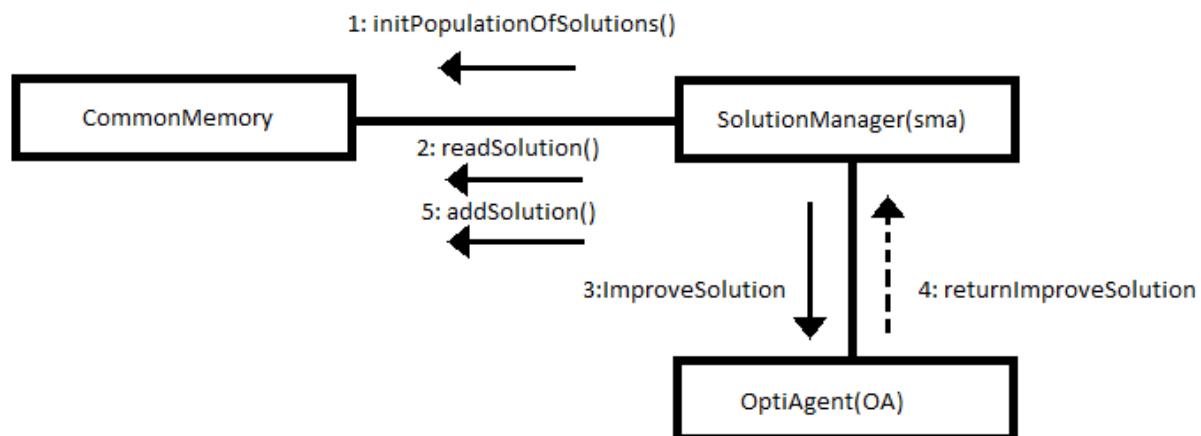


Ilustración 1 Diagrama de comunicación del proceso de resolución de JABAT [8]

En el anterior diagrama se puede observar los dos agentes de los que se hace uso:

- OptiAgent: formado por distintos algoritmos de mejora (optimization agents).
- SolutionManager: selecciona una solución de la población inicial de soluciones, la envía a los agentes de optimización y la sustituye en caso de mejora.

### **3.1.1 Heurísticas.**

Antes de hablar de las heurísticas, se debe mencionar que existen métodos exactos para el VRPTW, pero estos métodos solo suelen encontrar soluciones para problemas de hasta 50 clientes. El objetivo de este trabajo es desarrollar un sistema multiagente capaz de resolver el VRPTW sin limitaciones de clientes y, por tanto, no se tendrán en cuenta estos métodos para el desarrollo del trabajo.

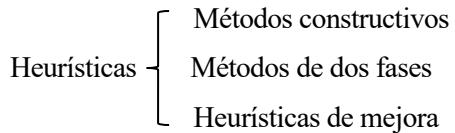
Linda Bibiana Rocha, Elsa Cristina González y Javier Arturo Orjuela definen las heurísticas como procedimientos que proporcionan soluciones de aceptable calidad mediante una exploración limitada del espacio de búsqueda. En su artículo clasifican las heurísticas en tres grupos: métodos constructivos, métodos de dos fases y heurísticas de mejora [9]. Edwin Montes Orozco explica muy bien estos tres grupos en su trabajo de fin de grado [10], diferenciándolos de la siguiente forma.

Métodos constructivos: Rafael Martí los define como métodos que construyen paso a paso la solución a un problema. Basados en su mayoría, en la mejor elección en cada iteración [11].

Métodos de dos fases: Algunos de estos métodos son de asignación elemental, el algoritmo de ramificación y acotamiento truncados, los procedimientos de búsqueda local, etc.

Heurísticas de mejora: conocidos como procedimientos de búsqueda local. Partiendo de una solución inicial, se busca una solución vecina mejor y se reemplaza.

Esquema de la clasificación de heurísticas:

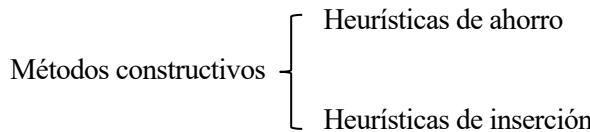


### **3.1.2 Métodos para obtener la población inicial de soluciones.**

Para obtener una solución inicial, Marius M. Solomon en un artículo [12] propone dos tipos de heurísticas basadas en métodos constructivos para resolver el problema VRPTW.

La primera heurística es una heurística de ahorro. Se parte de una solución inicial en la que cada cliente es visitado por un vehículo, y se intenta reducir el número de vehículos uniendo las rutas en caso de que se produzca un ahorro en esa unión. Uno de los algoritmos más conocidos es el de Clarke-Wright [13].

La segunda es una heurística de inserción. Se parte de cero rutas o vehículos, se insertan clientes de uno en uno en la ruta según una serie de criterios, si deja de cumplir las restricciones del problema se abre una nueva ruta.



### **3.1.3 Heurísticas de búsqueda local.**

Algunos de los algoritmos de búsqueda local utilizados para resolver el VRPTW que aparecen en [10], son los siguientes.

- El operador de intercambio  $\lambda$ : Propuesto por Lin en 1965. Se dice que un intercambio  $\lambda$  consiste en eliminar  $\lambda$  aristas de la solución y reconectar los  $\lambda$  segmentos restantes. El valor de  $\lambda$  depende del intercambio que se quiera hacer, y puede tomar distintos valores. Una solución es óptima si no puede ser mejorada utilizando  $\lambda$  intercambios [10].
- El algoritmo de Lin-Kernighan: Propuesto en 1973. Para el algoritmo es necesario partir de una solución inicial  $T$ . Se inicia  $i = 1$  y se selecciona  $x_i$  e  $y_i$  como el par más alejado en el paso  $i$ . Tras realizar una serie de iteraciones, si la mejor solución se da para  $i = k$  entonces se intercambia  $x_1, \dots, x_i$  con  $y_1, \dots, y_i$  para dar una nueva solución  $T$  [14].
- El operador Or-opt: Propuesto por Ilhan Or en 1976 [15]. Consiste en eliminar un segmento de  $k$  clientes de una ruta y colocar ese segmento en otra parte de la ruta, sin variar el orden de los clientes dentro del segmento extraído [10].



Ilustración 2 Intercambio Or-Opt de 2 clientes

- GENI y US: Propuesto por M. Gendreau, A. Hertz y G. Laporte en 1992. GENI es un método de inserción generalizado (GENeralized Insertion). Adicionalmente, se propone un algoritmo posterior a la optimización llamado US, por sus siglas en inglés, Unstringing y Stringing [16].
- Algoritmos de transferencias cíclicas: Thompson y Psaraftis proponen este algoritmo en 1993 [17]. Estos algoritmos buscan eliminar clientes de una ruta y colocarlos en otra de manera cíclica [18].
- Operadores de Van Breedam: En 1995 Van Breedam propone dos operadores para intercambiar clientes entre dos rutas diferentes. Estos operadores son denominados como String Relocation y String Exchange [19] [18].

### 3.1.4 Metaheurísticas.

En este apartado se explicarán algunas de las metaheurísticas más utilizadas para la resolución de un problema VRP.

En el blog de la Universidad Politécnica de Valencia [20] se definen las metaheurísticas como la combinación de manera inteligente de diversas técnicas con el fin de explorar el espacio de soluciones. También se proporciona la definición de Osman y Kelly (1996) [21]: “*Los procedimientos metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son ni efectivos ni eficientes. Los metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y la mecánica estadística*”. A continuación, se explican algunas metaheurísticas.

- Algoritmos genéticos: fueron propuestos por Holland en 1975. En el año 1989 Goldberg los define como [22]: “*los Algoritmos Genéticos son algoritmos de búsqueda basados en la mecánica de*

*selección natural y de la genética natural. Combinan la supervivencia del más apto entre estructuras de secuencias con un intercambio de información estructurado, aunque aleatorizado, para constituir así un algoritmo de búsqueda que tenga algo de las genialidades de las búsquedas humanas”*

- Recocido simulado: propuesto por primera vez en 1983 por Kirkpatrick, Gelatt y Vecchi, resumen su artículo de la siguiente manera [23]: “*existe una conexión profunda y útil entre la mecánica estadística (el comportamiento de sistemas con muchos grados de libertad en equilibrio térmico a una temperatura finita) y optimización multivariante o combinatoria (encontrar el mínimo de una función dada dependiendo de muchos parámetros). Una analogía detallada con el recocido en sólidos proporciona un marco para la optimización de las propiedades de sistemas grandes y complejos. Esta conexión con la mecánica estadística proporciona una perspectiva desconocida sobre los problemas y métodos tradicionales de optimización.*”
- Algoritmos de Hormigas: propuesto por Marcos Dorigo en 1992 [24]. El algoritmo se basa en el comportamiento de las hormigas a la hora de buscar comida. Las hormigas siguen a otras gracias a un rastro de feromonas que dejan a su paso. Cuanto más corto sea el camino, mayor es la concentración de feromonas, esto les permite encontrar la ruta más corta del hormiguero a la comida.
- Búsqueda Tabú: propuesto por Glover en 1986 [25]. Este algoritmo busca realizar métodos heurísticos de búsqueda local añadiendo una memoria, de manera que dota de cierta inteligencia a la heurística.

## 3.2 Programas considerados para programar el problema.

### 3.2.1 AnyLogic

AnyLogic es un programa que permite construir modelos de simulación de tres tipos: sistemas dinámicos, eventos discretos y basados en agentes. Crear un modelo es aparentemente sencillo, para modelos básicos no es necesario saber programación, se pueden crear seleccionando elementos con el ratón y arrastrándolos hacia la pantalla. Sin embargo, para modelos más realistas, es necesario conocer el lenguaje de programación llamado Java, ya que AnyLogic está basado en este.

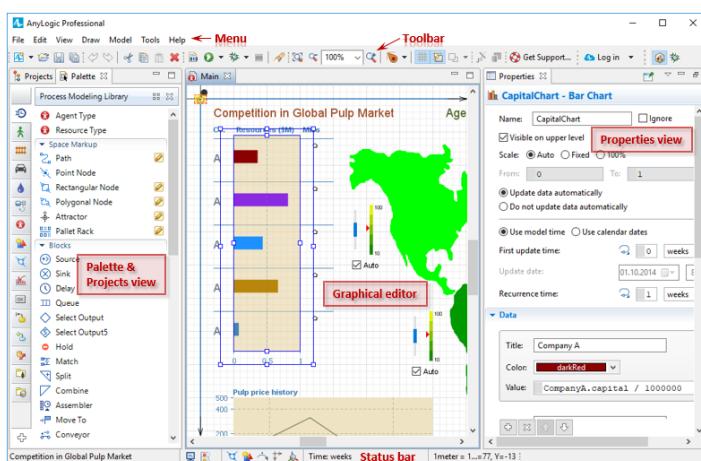


Ilustración 3 Interfaz de usuario de AnyLogic ([help.anylogic.com](http://help.anylogic.com))

### 3.2.1.1 Java

Java fue creado por James Goslin y su equipo en 1991, y posteriormente publicado por Sun Microsystems en el año 1995. El objetivo era crear un programa similar a C++ pero con su propia máquina virtual. Una de las principales y más importantes características de Java es su independencia de la plataforma (sistema operativo), es decir, que programas escritos en Java pueden ejecutarse en cualquier entorno (hardware) ya sea Linux, Windows, macOS, etc.

### 3.2.2 Python

Python es un lenguaje de alto nivel cuya primera aparición fue en el año 1991 de la mano de Guido van Rossum. En los años ochenta, Guido comenzó a trabajar en un equipo encargado de desarrollar un lenguaje llamado ABC. Finalmente, no tuvo éxito y Guido empezó en otro proyecto en 1986 llamado Amoeba, fue ahí cuando empezó a crear una variante del ABC seleccionando las mejores características de este y tratando de evitar sus fallos. Además, hizo posible que Python fuera un programa independiente de la plataforma y capaz de soportar librerías, características que no se encontraban en ABC [26].

Python es administrado por Python Software Foundation y tiene una licencia de código abierto, por lo que se puede hacer uso del programa de manera gratuita.

El conocimiento previo del lenguaje Python y el desarrollo previo de un sistema multiagente en Java, empujan al desarrollo del trabajo en Python.

## 3.3 Librerías/entornos para programar sistemas multiagentes en Python

Una librería es un archivo que puede usarse en un ejecutable y que tiene una serie de funcionalidades, el archivo aporta variables y funciones definidas por un tercero. Por ejemplo, una de las librerías más conocidas de Python es “Matplotlib” que sirve para generar gráficos a partir de unos datos. En nuestro trabajo, se ha usado Matplotlib para facilitar la interpretación de los resultados y así tener una referencia visual de la disposición de los clientes, almacén, rutas etc.

### 3.3.1 Librería Spade

Smart Python Agent Development Environment [27] es una plataforma para sistemas multiagentes escrita en Python y basada en XMPP (un protocolo de mensajería de respuesta instantánea). Algunas de sus características son las siguientes:

- Requiere una versión igual o superior a Python 3.6
- Basado en Asyncio (librería para entornos asíncronos) [28]
- Interfaz web gráfica
- Permite añadir características o funciones nuevas.
- Cualquier servidor XMPP es válido (hay que instalar un servidor)

### 3.3.2 Librería AIMA [29]

El archivo (librería) contiene un resumen de los algoritmos que aparecen en el libro *Artificial Intelligence: A Modern Approach*. El código es de libre uso y requiere una versión igual o superior a Python 2.2.

```

"""Implement Agents and Environments (Chapters 1-2).

The class hierarchies are as follows:

Object ## A physical object that can exist in an environment
Agent
    Wumpus
    RandomAgent
    ReflexVacuumAgent
    ...
Dirt
Wall
...

Environment ## An environment holds objects, runs simulations
XYEnvironment
    VacuumEnvironment
    WumpusEnvironment

EnvFrame ## A graphical representation of the Environment

"""

from utils import *
import random, copy

```

---

```

class Object:
    """This represents any physical object that can appear in an Environment.
    You subclass Object to get the objects you want. Each object can have a
    __name__ slot (used for output only)."""
    def __repr__(self):
        return '<%s>' % getattr(self, '__name__', self.__class__.__name__)

```

Ilustración 4 Ejemplo de parte del código de Aima

### 3.3.3 Librería Mesa [30]

Mesa es un entorno modular para construir, analizar y visualizar modelos de agentes. Para instalarlo se recomienda usar un entorno virtual y requiere de la versión 3 de Python. Para construir un modelo se puede:

- Escribir el código en un editor de texto.
- Crearlo de manera interactiva con Jupyter Notebook, un programa web de libre uso.

### 3.3.4 Resumen de librerías analizadas

	Spade	Aima	Mesa
Versión de Python	3.6	2.2	3
Herramienta de análisis	No	Sí	Sí
Interfaz gráfica web	Sí	No	Sí
Únicamente usa XMPP server	Sí	No	No
Claridad en la explicación de uso del código	Bien	Mal	Bien

Tabla 1 Comparativa librerías para programar multiagentes

### **3.3.5 Elección final**

Tras comenzar a programar los algoritmos de mejora que acabarían siendo los agentes optimizadores, se ha observado que es más sencillo programar el entorno de intercambio de información entre los agentes sin el uso de una librería para ello y así no es necesario amoldar el código a la librería. De esta manera, se ha programado una función que, con un bucle, simula el intercambio de información entre un agente encargado de seleccionar e intercambiar soluciones con los agentes optimizadores, es decir, simula el diagrama de comunicación del proceso de resolución de JABAT que se puede observar en la ilustración 1.

# 4 APLICACIÓN PRÁCTICA

## 4.1 Datos del problema VRPTW

Los datos que se utilizan se obtienen de las instancias de Solomon [12] y [31]. Estas instancias son problemas del VRPTW que vienen clasificados en diferentes grupos. Según estos grupos varía la distribución de los clientes o las ventanas temporales, y para cada grupo se tienen instancias de 25, 50 y 100 clientes. Estas instancias vienen dadas por los grupos C1, C2, R1, R2, RC1 y RC2. Las instancias tipo C tienen los clientes de manera agrupada, en los tipos R están dispersos de manera uniforme y los tipos RC son una mezcla de ambos. En cuanto al número que sigue a la letra, va referido a las ventanas temporales.

En las siguientes ilustraciones se puede ver la distribución de clientes de algunas de las instancias de los grupos mencionados anteriormente. Estas gráficas han sido construidas con Matplotlib y representan un plano en dos dimensiones compuesto por los ejes de coordenadas x e y. El punto rojo indica el almacén de partida, el resto son los clientes, se representa una vista de planta.

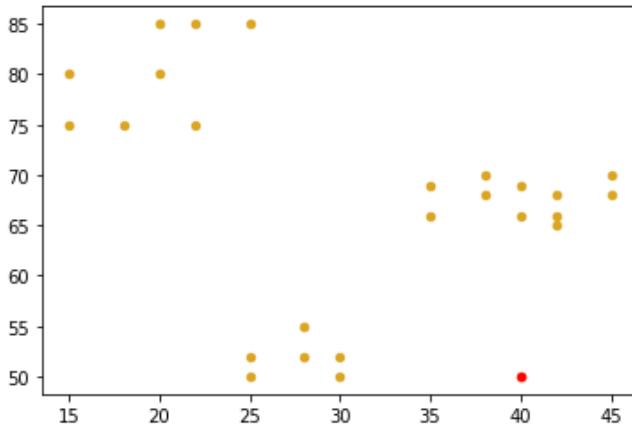


Ilustración 5 Distribución de instancia C103\_025

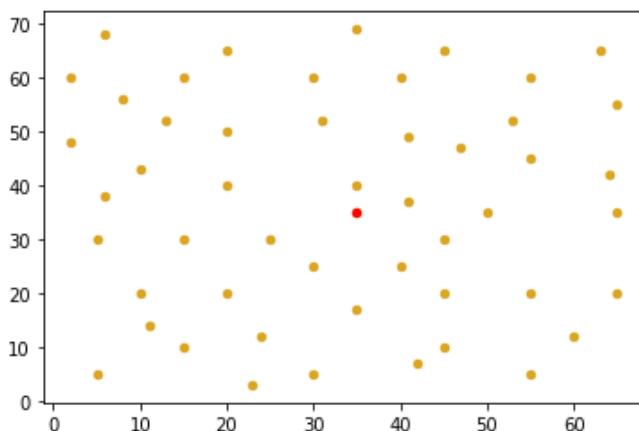


Ilustración 6 Distribución de instancia R101\_050

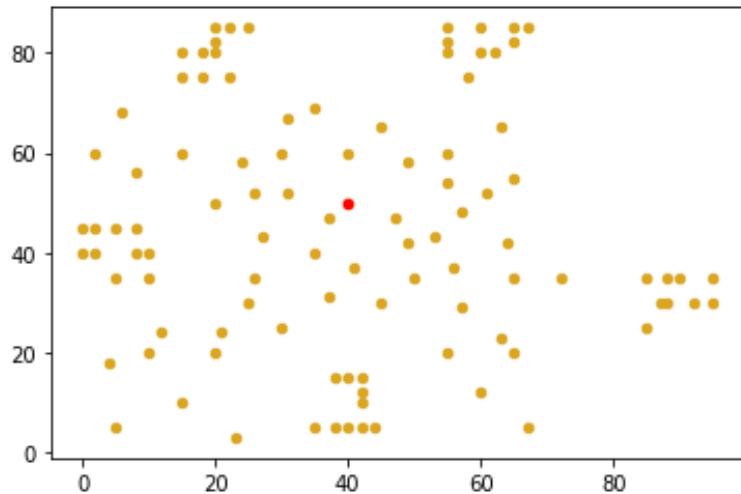


Ilustración 7 Distribución de instancia RC101\_100

## 4.2 Población inicial de soluciones

El objetivo es resolver un problema de rutado de vehículos con ventanas temporales utilizando multiagentes. Para ello el primer paso es obtener una población inicial de soluciones para posteriormente mejorarlas. La población inicial es necesaria para evitar caer en un óptimo local. Que una solución inicialmente sea mejor que otra no quiere decir que tras aplicar una serie de mejoras y alcanzar un óptimo, este sea mejor que el óptimo alcanzado para la otra solución inicialmente peor. En la siguiente ilustración se puede observar un ejemplo de por qué es necesario partir de una población de soluciones.

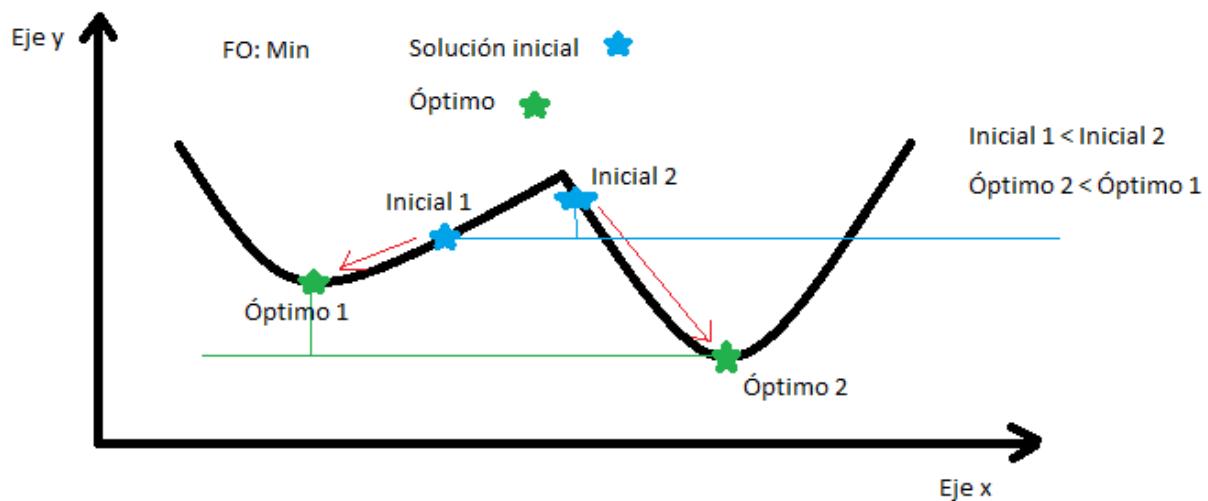


Ilustración 8 Ejemplo de gráfica con dos óptimos y dos soluciones iniciales.

Para obtener una solución inicial, hemos optado por heurísticas de inserción [12]. En total han sido cuatro soluciones las que se han obtenido, siguiendo una misma filosofía, pero cambiando el criterio (variando el paso 1 del pseudocódigo). Inicialmente todos los clientes están sin ruta. A continuación, podemos ver los pseudocódigos de los cuatro algoritmos utilizados.

Pseudocódigo del algoritmo correspondiente a la primera solución. Más cercano.

Hasta que todos los clientes tengan una ruta asignada:

1. Para iniciar la ruta, buscar el cliente sin ruta más cercano al almacén y añadir a la ruta si cumple las restricciones.
2. Buscar el cliente sin ruta más cercano al último cliente añadido en la ruta.
3. Comprobar si al añadir el cliente se cumplen las restricciones del problema.
4. Si cumple las restricciones se vuelve al paso 2, e.o.c iniciar una nueva ruta y volver al paso 1.
5. Devolver el valor de la función objetivo.

Pseudocódigo del algoritmo correspondiente a la segunda solución. Ventana estrecha.

Hasta que todos los clientes tengan una ruta asignada:

1. Para iniciar la ruta, buscar el cliente sin ruta con la ventana temporal más estrecha (esto quiere decir menor diferencia entre el tiempo final e inicial para realizar la entrega) y añadir a la ruta si cumple las restricciones.
2. Buscar el cliente sin ruta más cercano al último cliente añadido en la ruta.
3. Comprobar si al añadir el cliente se cumplen las restricciones del problema y calcular el valor de la función objetivo.
4. Si cumple las restricciones se vuelve al paso 2, e.o.c iniciar una nueva ruta y volver al paso 1.
5. Devolver el valor de la función objetivo.

Pseudocódigo del algoritmo correspondiente a la tercera solución. Más lejano.

Hasta que todos los clientes tengan una ruta asignada:

1. Para iniciar la ruta, buscar el cliente sin ruta más lejano al almacén y añadir a la ruta si cumple las restricciones.
2. Buscar el cliente sin ruta más cercano al último cliente añadido en la ruta.
3. Comprobar si al añadir el cliente se cumplen las restricciones del problema.
4. Si cumple las restricciones se vuelve al paso 2, e.o.c iniciar una nueva ruta y volver al paso 1.
5. Devolver el valor de la función objetivo.

### Pseudocódigo del algoritmo correspondiente a la cuarta solución. Menor tiempo de inicio de entrega

Hasta que todos los clientes tengan una ruta asignada:

- 1.Para iniciar la ruta, buscar el cliente sin ruta con el tiempo de inicio de entrega más temprano y añadir a la ruta si cumple las restricciones.
2. Buscar el cliente sin ruta más cercano al último cliente añadido en la ruta.
3. Comprobar si al añadir el cliente se cumplen las restricciones del problema.
4. Si cumple las restricciones se vuelve al paso 2, e.o.c iniciar una nueva ruta y volver al paso 1.
5. Devolver el valor de la función objetivo.

## 4.3 Agentes Optimizadores

Para la creación de los distintos agentes optimizadores, aparte de los conocimientos adquiridos en la carrera, se han utilizado estrategias de búsqueda local similares a las desarrolladas en [7] , [32] y [10] . Algunas de estas heurísticas han sido ya comentadas en el apartado 3.1.3.

### 4.3.1 OA Vecinos

Con este algoritmo, los agentes tratan de mejorar el objetivo intercambiando el orden en que se visitan clientes vecinos dentro de una misma ruta.

#### Pseudocódigo

Para cada ruta i desde j=1

- 1.Intercambiar la posición del cliente j con el cliente j+1 de la ruta i.
- 2.Comprobar si cumple las restricciones y calcular el valor de la función objetivo.
- 3.En caso de no cumplir las restricciones invertir el cambio hecho, e.o.c mantener el cambio.
- 4.Si hay clientes para seguir intercambiando, volver al paso 1 incrementando el valor de j, es decir, intercambiar con el siguiente cliente.
- 5.Una vez se agotan los clientes a intercambiar en la ruta, pasar a la siguiente ruta incrementando el valor de i en el algoritmo y comenzar de nuevo por el primer cliente.
6. Devolver el valor de la función objetivo una vez finalizado el algoritmo con todas las rutas.

### 4.3.2 OA Buscar solitario

Tras obtener la población inicial de soluciones, se ha observado que se obtienen o se pueden obtener rutas en las que se visita a un solo cliente. Debido a que el coste de utilizar un vehículo es muy elevado, uno de los objetivos es minimizar el número de vehículos o rutas (hay un vehículo para cada ruta). Para ello se ha propuesto este algoritmo que busca rutas de un solo cliente y trata de colocar ese cliente como primer cliente a visitar en otra ruta.

#### Pseudocódigo

Mientras no encuentre una solución o se agoten las rutas.

- 1.Buscar la primera ruta en la que solo tenga un cliente y guardar el cliente en la variable solitario.
- 2.Para cada ruta distinta a la del solitario empezando por la primera, desplazar todos los clientes un espacio a la derecha.
- 3.Insertar el cliente solitario en la primera posición de la ruta.
- 4.Comprobar restricciones y función objetivo.
- 6.Si no cumple las restricciones o la solución es peor, deshacer el cambio y cambiar a la siguiente ruta volviendo al paso 2. Terminar el algoritmo e.o.c.
- 7.Si tras probar en todas las rutas no se ha encontrado una solución mejor para insertar el solitario encontrado, volver al paso 1 partiendo de la ruta posterior a la original del solitario anterior.
8. Devolver el valor de la función objetivo.

#### **4.3.3 OA String Cross**

Para este algoritmo nos hemos basado en el algoritmo llamado OA\_StringCross que se propone en JABAT [7]. La diferencia entre ambos algoritmos se da en que en JABAT se contemplan todos los posibles puntos de reconexión, mientras que este algoritmo los genera de manera aleatoria.

#### Pseudocódigo

1. Generar dos rutas aleatorias, ruta\_i y ruta\_j, que al menos tengan un cliente y que sean distintas.
2. Generar dos puntos aleatorios pi y pj en cada ruta diviendo cada ruta en dos partes.
3. Crear una nueva ruta i formada por la primera parte de ruta\_i desde 1 hasta pi, y la segunda de la ruta j formada desde pj+1 hasta el último cliente n de j (1...pi, pj+1...n).
4. Crear una nueva ruta j formada por la primera parte de ruta\_j desde 1 hasta pj, y la segunda de la ruta\_i formada desde pi+1 hasta el último cliente m de i (1...pj, pi+1...m).
5. Comprobar las restricciones y calcular el valor de la función objetivo.
6. Si no cumple las restricciones o el valor de la función objetivo es mayor, deshacer los cambios.
7. Devolver el valor de la función objetivo.

#### **4.3.4 OA Insertion**

El agente llamado Insertion es un algoritmo de inserción, el algoritmo propone un cliente aleatorio de una ruta y varía el orden de visita del cliente hasta lograr una mejora (en caso de existir esa mejora), manteniendo el resto de los clientes en el mismo orden.

#### Pseudocódigo

1. Generar una ruta aleatoria que tenga al menos 1 cliente en su ruta.
2. Generar una posición aleatoria pos e iniciamos j=1.
3. Mientras no exista una mejor solución y haya clientes sin intercambiar, intercambiar el cliente j con el cliente de la posición pos.
4. Comprobar restricciones y calcular el valor de la función objetivo.

5. Si no cumple las restricciones o el valor de la función objetivo es peor, deshacer el cambio, incrementar  $j$  y volver al paso 3. Finalizar el algoritmo e.o.c.
6. Devolver el valor de la función objetivo.

#### 4.3.5 OA Intercambio

Este algoritmo proviene de la idea de mezclar el algoritmo de Insertion con el algoritmo String Cross. El algoritmo funciona igual que Insertion, pero esta vez intercambia las posiciones con una ruta distinta y no con la misma.

##### Pseudocódigo

1. Generar dos rutas aleatorias, ruta<sub>i</sub> y ruta<sub>j</sub>, con al menos un cliente en la ruta.
2. Generar una posición aleatoria para la ruta<sub>i</sub> llamada pos e inicializar  $j$  en 1.
3. Mientras haya clientes para intercambiar en ruta<sub>j</sub> y no exista una solución mejor. Intercambiar el cliente  $j$  con el cliente de la ruta<sub>i</sub> en la posición pos.
4. Comprobar restricciones y función objetivo.
5. Si no cumple las restricciones o el valor de la función objetivo es peor, deshacer el cambio, incrementar  $j$  y volver al paso 3. Finalizar el algoritmo e.o.c.
6. Devolver el valor de la función objetivo.

### 4.4 Solution Manager y entorno

Como se dijo anteriormente, no se hará uso de librerías de multiagentes. Para simular la interacción entre los agentes optimizadores y el agente Solution Manager, se ha creado este último como una función en la que se llamará a otras funciones las cuales serán los agentes optimizadores. Para intercambiar información lo único que se deberá hacer es introducir la función SolutionManager en un bucle, los agentes optimizadores son funciones que devuelven el valor de la función objetivo y modifican las matrices de rutas y arcos, mientras que SolutionManager modifica un vector formado por la población inicial de soluciones y por otro lado las distintas matrices de rutas y arcos para cada solución. El programa en su conjunto simula la interacción entre diferentes agentes. A continuación, se puede observar el pseudocódigo y en la ilustración 8 un ejemplo de código.

##### Pseudocódigo

1. Seleccionar una solución con una probabilidad que depende de la calidad de la solución. Ordenando un vector que contiene las soluciones de mayor a menor valor de la función objetivo, se le da una puntuación a cada solución correspondiente a su posición en dicho vector, siendo la puntuación de uno a cinco puntos. De esta forma se genera un número aleatorio de uno a diez con el cuál podremos seleccionar la solución correspondiente.
2. Copiar en quince variables auxiliares (porque hay cinco agentes optimizadores y tres datos a copiar) el valor de la función objetivo, la matriz de ruta y la matriz de arcos correspondientes a la solución seleccionada.
3. Pasar los datos copiados a los agentes optimizadores para que proporcionen una solución igual o mejor a la original.
4. Seleccionar una de las cinco soluciones obtenidas según un criterio (aleatorio, la mejor solución y la segunda peor solución). En [7] proponen hasta seis criterios diferentes tanto para seleccionar una solución a mejorar, como para añadirla a la población de soluciones. El criterio se selecciona dando un valor para la variable “elección” comprendido entre 1 y 3. Para la resolución de los problemas, “elección” se ha asignado de las siguientes tres formas; fijando el valor a 1, fijando el valor a 2 y

generando un valor aleatorio entre 1 y 3.

4.1. Elección = 1: se selecciona la solución de manera aleatoria entre las cinco soluciones proporcionadas por los agentes.

4.2. Elección = 2: se selecciona la mejor solución entre las proporcionadas por los agentes.

4.3. Elección = 3: se selecciona la segunda peor solución entre las proporcionadas por los agentes.

5. Sustituir la solución de la población de soluciones que se había seleccionado por la nueva solución obtenida de los agentes proveniente del paso cuatro.

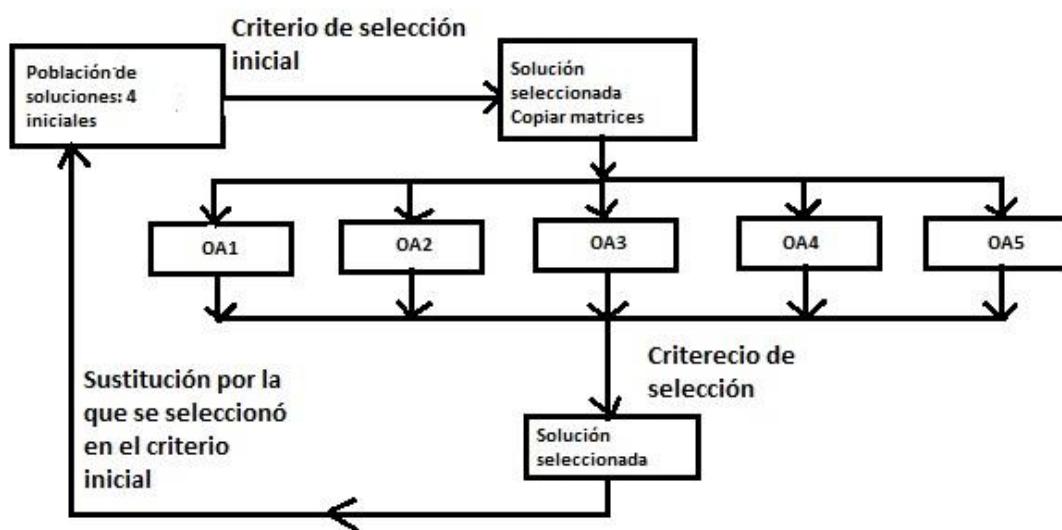


Ilustración 9 Esquema SolutionManager. Ejemplo de intercambio de información

# 5 EXPERIENCIA COMPUTACIONAL

---

En este apartado se darán las características del ordenador utilizado, el programa y la versión de Python, así como los distintos criterios utilizados a la hora de solucionar las distintas instancias y recoger los datos.

## 5.1 Características del ordenador utilizado.

Para este trabajo, se ha utilizado un ordenador de sobremesa con la última versión de Windows 10 de 64 bits. El ordenador está compuesto por un procesador Ryzen 5 1600 AF y una gráfica AMD Radeon RX 570.

## 5.2 Programa y versión de Python

El código para la resolución del problema ha sido desarrollado en el programa Spyder, un entorno de desarrollo gratuito escrito en Python. En concreto se hace uso de la versión 5 de Spyder y la versión 3.7.9 de Python.

## 5.3 Criterios para la recogida de datos

El código incluye aleatoriedad, por tanto, la decisión ha sido generar treinta soluciones con el mismo criterio para obtener posteriormente una media. Cada instancia se resuelve con los tres criterios comentados en el apartado 4.4 para dar un valor a la variable “elección”. De esta manera se han obtenido tres resultados para “elección” =1, tres para “elección” =2 y tres para “elección” =aleatorio (1,3).;

La función SolutionManager se ha introducido en un bucle de 300 iteraciones para simular el intercambio de información, en algún caso se incrementará el número de iteraciones para observar como mejora la solución en periodo de tiempo mayor. El número de iteraciones en el bucle equivale a tiempo de intercambio de información entre agentes.

En cuanto a las instancias utilizadas, se utilizan tres instancias de cada grupo de instancias de Solomon ya comentadas en el apartado 4.1. Las instancias son de 50 clientes, no se ha optado por las instancias de 100 clientes para no incrementar el tiempo de ejecución. Los resultados para poder realizar una comparación se obtienen de [33], para poder utilizarlos es necesario tener en cuenta el número de vehículos ya que no se tienen en cuenta en la función objetivo de [33] y sí en la de este trabajo. En algunos casos se utiliza la instancia de 25 clientes debido a que no se dispone de resultados de la instancia de 50, el motivo por el cual esas instancias no aparecen es porque los métodos utilizados en [33] no han conseguido resolverlas y no se han encontrado en otros documentos.

Para las restricciones, se supone un valor de 100 para cada vehículo o ruta y una velocidad de 1, es decir, si la distancia es 10 tardará en recorrer la distancia un total de 10 unidades temporales.

# 6 RESULTADOS Y CONCLUSIONES

---

Para la obtención de los resultados, se han utilizado las siguientes instancias de 50 clientes de Solomon:

C101.50, C102.50, C103.50, RC101.50, RC102.50, RC103.50, R201.50, R202.50, R205.50, C201.50, C202.50, C203.50, RC201.50, RC201.25, RC202.25, R101.50, R102.50 y R103.50.

En todo momento se utiliza el punto como separador decimal y la coma como separador de miles.

## 6.1 Obtención de resultados.

En el apartado 5.3 se explicaron los criterios para la obtención de resultados. Cuando se habla de solución se habla del valor de la función objetivo. A continuación, se definen los diferentes términos que se utilizan en las tablas.

- Inicial x: solución inicial número x en um. Hay un total de 4.
- Mejorada x: solución inicial número x en um. tras correr el código.
- % Mejora x:  $\frac{\text{Inicial } x - \text{Mejorada } x}{\text{Inicial } x} * 100$
- Tiempo ejec: es el tiempo de ejecución del código en segundos.
- Selec elección 1: el criterio para seleccionar la solución recibida de los agentes es escoger una solución de manera aleatoria.
- Selec elección 2: el criterio para seleccionar la solución recibida de los agentes es escoger la mejor solución obtenida.
- Selec elección rand: el criterio para seleccionar la solución recibida de los agentes es, de manera aleatoria seguir el criterio de selección 1, de selección 2 o escoger la segunda peor solución obtenida de los agentes.

A continuación, se observan las diferentes tablas que se han generado para cada instancia. Las tablas muestran la media de las treinta experimentaciones que se han realizado para cada criterio e instancia. En amarillo se subraya la mejor solución de las que conforman la tabla.

	C101.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	2220.90	2220.90	2220.90
Inicial 2	3650.41	3650.41	3650.41
Inicial 3	4963.21	4963.21	4963.21
Inicial 4	1857.55	1857.55	1857.55
Mejorada 1	1961.90	1933.97	2072.46
Mejorada 2	2416.81	3438.66	2086.28
Mejorada 3	4024.74	4728.64	4017.10
Mejorada 4	1551.32	1569.78	1583.25
% Mejora 1	11.66%	12.92%	6.68%
% Mejora 2	55.55%	9.53%	70.43%
% Mejora 3	42.26%	10.56%	42.60%
% Mejora 4	13.79%	12.96%	12.35%
Tiempo ejec	43.02	41.93	42.51

Tabla 2 Resultados medios de la instancia C101.50

	C102.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	1997.35	1997.35	1997.35
Inicial 2	3353.34	3353.34	3353.34
Inicial 3	3952.80	3952.80	3952.80
Inicial 4	2474.19	2474.19	2474.19
Mejorada 1	1732.58	1674.99	1750.71
Mejorada 2	2168.96	2343.79	2312.98
Mejorada 3	2787.13	2568.16	2682.97
Mejorada 4	2003.56	1906.60	1910.61
% Mejora 1	13.26%	16.14%	12.35%
% Mejora 2	59.30%	50.54%	52.09%
% Mejora 3	58.36%	69.32%	63.58%
% Mejora 4	23.56%	28.42%	28.22%
Tiempo ejec	44.25	43.32	43.44

Tabla 3 Resultados medios de la instancia C102.50

	C103.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	1427.08	1427.08	1427.08
Inicial 2	2890.85	2890.85	2890.85
Inicial 3	2815.50	2815.50	2815.50
Inicial 4	1925.50	1925.50	1925.50
Mejorada 1	1264.29	1084.69	1281.67
Mejorada 2	1840.46	1659.30	1946.05
Mejorada 3	2052.00	2027.23	1953.62
Mejorada 4	1522.94	1488.54	1466.02
% Mejora 1	11.41%	23.99%	10.19%
% Mejora 2	73.60%	86.30%	66.21%
% Mejora 3	53.50%	55.24%	60.40%
% Mejora 4	28.21%	30.62%	32.20%
Tiempo ejec	44.63	44.59	45.22

Tabla 4 Resultados medios de la instancia C103.50

	RC101.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	4441.00	4441.00	4441.00
Inicial 2	4552.80	4552.80	4552.80
Inicial 3	4298.73	4298.73	4298.73
Inicial 4	3538.67	3538.67	3538.67
Mejorada 1	4191.58	4097.72	4086.55
Mejorada 2	4323.53	4262.28	4132.05
Mejorada 3	4234.14	4058.27	4261.56
Mejorada 4	3037.18	3021.22	3212.19
% Mejora 1	5.62%	7.73%	7.98%
% Mejora 2	5.16%	6.54%	9.47%
% Mejora 3	1.45%	5.41%	0.84%
% Mejora 4	11.29%	11.65%	7.35%
Tiempo ejec	43.10	42.59	44.06

Tabla 5 Resultados medios de la instancia RC101.50

RC102.50			
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	4116.95	4116.95	4116.95
Inicial 2	3914.73	3914.73	3914.73
Inicial 3	3607.88	3607.88	3607.88
Inicial 4	3250.66	3250.66	3250.66
Mejorada 1	3733.67	3640.73	3167.60
Mejorada 2	3542.24	3682.38	3867.56
Mejorada 3	3533.02	3320.58	3534.20
Mejorada 4	2887.41	3000.90	3014.51
% Mejora 1	9.31%	11.57%	23.06%
% Mejora 2	9.05%	5.64%	1.15%
% Mejora 3	1.82%	6.98%	1.79%
% Mejora 4	8.82%	6.07%	5.74%
Tiempo ejec	44.34	44.29	44.16

Tabla 6 Resultados medios de la instancia RC102.50

RC103.50			
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	3008.03	3008.03	3008.03
Inicial 2	3204.26	3204.26	3204.26
Inicial 3	3404.33	3404.33	3404.33
Inicial 4	2905.55	2905.55	2905.55
Mejorada 1	2830.12	2387.05	2247.36
Mejorada 2	2787.93	2949.32	2606.88
Mejorada 3	3362.43	3186.19	3346.48
Mejorada 4	2422.40	2443.75	2471.73
% Mejora 1	5.91%	20.64%	25.29%
% Mejora 2	13.84%	8.48%	19.86%
% Mejora 3	1.39%	7.25%	1.92%
% Mejora 4	16.06%	15.35%	14.42%
Tiempo ejec	43.32	43.45	43.57

Tabla 7 Resultados medios de la instancia RC103.50

	R201.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	3754.14	3754.14	3754.14
Inicial 2	4890.56	4890.56	4890.56
Inicial 3	4251.89	4251.89	4251.89
Inicial 4	3112.54	3112.54	3112.54
Mejorada 1	3229.29	3143.04	2971.45
Mejorada 2	4372.95	4580.28	4375.35
Mejorada 3	3566.70	3888.65	3607.32
Mejorada 4	2696.25	2849.89	2484.30
% Mejora 1	13.98%	16.28%	20.85%
% Mejora 2	13.79%	8.27%	13.72%
% Mejora 3	18.25%	9.68%	17.17%
% Mejora 4	11.09%	7.00%	16.73%
Tiempo ejec	43.38	42.09	43.29

Tabla 8 Resultados medios de la instancia R201.50

	R202.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	2961.83	2961.83	2961.83
Inicial 2	4120.89	4120.89	4120.89
Inicial 3	3288.38	3288.38	3288.38
Inicial 4	2787.15	2787.15	2787.15
Mejorada 1	2478.43	2257.79	2381.78
Mejorada 2	3720.09	3730.58	2849.86
Mejorada 3	2832.65	2869.86	2501.77
Mejorada 4	2385.75	2191.96	2217.86
% Mejora 1	16.32%	23.77%	19.58%
% Mejora 2	13.53%	13.18%	42.91%
% Mejora 3	15.39%	14.13%	26.56%
% Mejora 4	13.55%	20.10%	19.22%
Tiempo ejec	43.75	43.56	44.18

Tabla 9 Resultados medios de la instancia R202.50

	R205.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	3201.02	3201.02	3201.02
Inicial 2	3692.48	3692.48	3692.48
Inicial 3	3366.50	3366.50	3366.50
Inicial 4	2710.15	2710.15	2710.15
Mejorada 1	2572.78	2489.17	2882.71
Mejorada 2	2874.54	2460.26	2235.87
Mejorada 3	2579.54	3085.05	2236.07
Mejorada 4	2386.98	2165.25	2184.01
% Mejora 1	19.63%	22.24%	9.94%
% Mejora 2	25.55%	38.49%	45.50%
% Mejora 3	24.58%	8.79%	35.31%
% Mejora 4	10.10%	17.02%	16.44%
Tiempo ejec	43.67	43.34	43.70

Tabla 10 Resultados medios de la instancia R205.50

	C201.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	3227.87	3227.87	3227.87
Inicial 2	3889.33	3889.33	3889.33
Inicial 3	3867.72	3867.72	3867.72
Inicial 4	1754.56	1754.56	1754.56
Mejorada 1	2208.47	2861.24	2442.95
Mejorada 2	3071.83	3455.87	2549.84
Mejorada 3	2229.13	3590.35	1871.55
Mejorada 4	1734.58	1580.67	1717.04
% Mejora 1	31.58%	11.36%	24.32%
% Mejora 2	25.33%	13.43%	41.50%
% Mejora 3	50.76%	8.59%	61.84%
% Mejora 4	0.62%	5.39%	1.16%
Tiempo ejec	43.40	41.87	44.45

Tabla 11 Resultados medios de la instancia C201.50

	C202.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	3155.24	3155.24	3155.24
Inicial 2	3030.95	3030.95	3030.95
Inicial 3	2919.50	2919.50	2919.50
Inicial 4	2461.33	2461.33	2461.33
Mejorada 1	2461.25	1775.34	1537.72
Mejorada 2	2058.41	1912.69	2101.22
Mejorada 3	1445.12	1457.05	1529.55
Mejorada 4	2149.20	1947.89	1939.25
% Mejora 1	21.99%	43.73%	51.26%
% Mejora 2	30.82%	35.44%	29.47%
% Mejora 3	46.73%	46.35%	44.05%
% Mejora 4	9.89%	16.27%	16.55%
Tiempo ejec	44.05	43.96	44.74

Tabla 12 Resultados medios de la instancia C202.50

	C203.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	2510.83	2510.83	2510.83
Inicial 2	2374.43	2374.43	2374.43
Inicial 3	2193.46	2193.46	2193.46
Inicial 4	2258.81	2258.81	2258.81
Mejorada 1	1950.03	1723.72	1783.51
Mejorada 2	1690.45	1751.68	1473.25
Mejorada 3	1233.92	1422.17	1318.95
Mejorada 4	1709.95	1522.92	1646.64
% Mejora 1	22.33%	31.35%	28.97%
% Mejora 2	27.24%	24.80%	35.89%
% Mejora 3	38.22%	30.72%	34.83%
% Mejora 4	21.86%	29.31%	24.38%
Tiempo ejec	44.42	45.10	45.28

Tabla 13 Resultados medios de la instancia C203.50

	RC201.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	3756.97	3756.97	3756.97
Inicial 2	4063.63	4063.63	4063.63
Inicial 3	3868.40	3868.40	3868.40
Inicial 4	2985.49	2985.49	2985.49
Mejorada 1	3104.46	3374.33	3082.57
Mejorada 2	3673.64	3705.00	3066.71
Mejorada 3	3347.65	3561.90	2708.30
Mejorada 4	2930.82	2903.09	2824.11
% Mejora 1	17.37%	10.18%	17.95%
% Mejora 2	10.38%	9.55%	26.54%
% Mejora 3	13.86%	8.16%	30.88%
% Mejora 4	1.46%	2.19%	4.30%
Tiempo ejec	44.38	42.79	43.53

Tabla 14 Resultados medios de la instancia RC201.50

	RC201.25		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	2237.63	2237.63	2237.63
Inicial 2	2239.04	2239.04	2239.04
Inicial 3	2020.57	2020.57	2020.57
Inicial 4	1364.01	1364.01	1364.01
Mejorada 1	1470.17	1674.48	1486.72
Mejorada 2	1924.36	1967.15	2056.46
Mejorada 3	1969.59	1798.68	1961.56
Mejorada 4	1161.24	1165.76	1170.65
% Mejora 1	34.30%	25.17%	33.56%
% Mejora 2	14.06%	12.15%	8.16%
% Mejora 3	2.28%	9.92%	2.64%
% Mejora 4	9.06%	8.86%	8.64%
Tiempo ejec	13.26	13.42	13.02

Tabla 15 Resultados medios de la instancia RC201.25

	RC202.25		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	1556.21	1556.21	1556.21
Inicial 2	1884.39	1884.39	1884.39
Inicial 3	1895.82	1895.82	1895.82
Inicial 4	1664.75	1664.75	1664.75
Mejorada 1	1014.02	1323.93	1017.19
Mejorada 2	1371.07	1471.69	1545.17
Mejorada 3	1505.79	1687.58	1143.14
Mejorada 4	1495.73	1309.24	1353.81
% Mejora 1	34.84%	14.93%	34.64%
% Mejora 2	32.99%	26.52%	21.80%
% Mejora 3	25.06%	13.38%	48.37%
% Mejora 4	10.86%	22.84%	19.98%
Tiempo ejec	13.43	13.46	13.37

Tabla 16 Resultados medios de la instancia RC202.25

	R101.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	4886.90	4886.90	4886.90
Inicial 2	5189.93	5189.93	5189.93
Inicial 3	5528.42	5528.42	5528.42
Inicial 4	4036.71	4036.71	4036.71
Mejorada 1	4455.30	4617.69	4655.33
Mejorada 2	5107.66	4925.60	5089.99
Mejorada 3	5331.06	5289.81	5285.50
Mejorada 4	3839.63	3808.39	3518.37
% Mejora 1	8.83%	5.51%	4.74%
% Mejora 2	1.68%	5.41%	2.05%
% Mejora 3	4.04%	4.88%	4.97%
% Mejora 4	4.03%	4.67%	10.61%
Tiempo ejec	40.62	40.35	41.94

Tabla 17 Resultados medios de la instancia R101.50

	R102.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	4093.94	4093.94	4093.94
Inicial 2	4330.44	4330.44	4330.44
Inicial 3	4710.70	4710.70	4710.70
Inicial 4	3361.53	3361.53	3361.53
Mejorada 1	3235.63	3778.20	3125.11
Mejorada 2	3893.71	3936.87	4022.62
Mejorada 3	4505.38	4393.20	4464.81
Mejorada 4	2796.29	2853.84	2736.56
% Mejora 1	20.97%	7.71%	23.66%
% Mejora 2	10.67%	9.61%	7.52%
% Mejora 3	5.02%	7.76%	6.01%
% Mejora 4	13.81%	12.40%	15.27%
Tiempo ejec	42.66	41.83	43.07

Tabla 18 Resultados medios de la instancia R102.50

	R103.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	3525.19	3525.19	3525.19
Inicial 2	3626.70	3626.70	3626.70
Inicial 3	3736.01	3736.01	3736.01
Inicial 4	2968.47	2968.47	2968.47
Mejorada 1	2398.20	2712.86	2507.38
Mejorada 2	3120.09	3193.98	3166.05
Mejorada 3	3478.02	3431.31	3481.16
Mejorada 4	2516.14	2389.88	2217.92
% Mejora 1	31.97%	23.04%	28.87%
% Mejora 2	14.37%	12.27%	13.07%
% Mejora 3	7.32%	8.64%	7.23%
% Mejora 4	12.83%	16.41%	21.29%
Tiempo ejec	43.62	42.76	43.06

Tabla 19 Resultados medios de la instancia R103.50

## 6.2 Resultados y conclusiones respecto a la población inicial de soluciones.

Como ya se explicó, la población inicial de soluciones se obtiene al variar el paso del algoritmo encargado de encontrar el primer cliente para añadir a la ruta. En la tabla “Inicial 1” corresponde al algoritmo cuyo primer cliente en la ruta es el más cercano al almacén, “Inicial 2” al cliente con la ventana temporal más estrecha, “Inicial 3” al cliente más lejano y, por último, “Inicial 4” al cliente con menor tiempo (fecha) de inicio de entrega.

A continuación, se observa una serie de gráficas en relación con la población inicial de soluciones y que ha sido obtenida de los resultados expuestos anteriormente. Para la obtención de las gráficas, se ha seguido el siguiente orden para representar las instancias en el “eje x”: C101.50, C102.50, C103.50, RC101.50, RC102.50, RC103.50, R201.50, R202.50, R205.50, C201.50, C202.50, C203.50, RC201.50, RC201.25, RC202.25, R101.50, R102.50 y R103.50.

En la siguiente tabla, se representa el valor de la función objetivo de las soluciones iniciales generadas mediante métodos constructivos (recordemos que hay 4, en cada valor del eje x hay 4 valores representados en el eje y) para cada una de las 18 instancias (eje x), siendo C101.50 el valor x=1 y R103.50 el valor x=18.

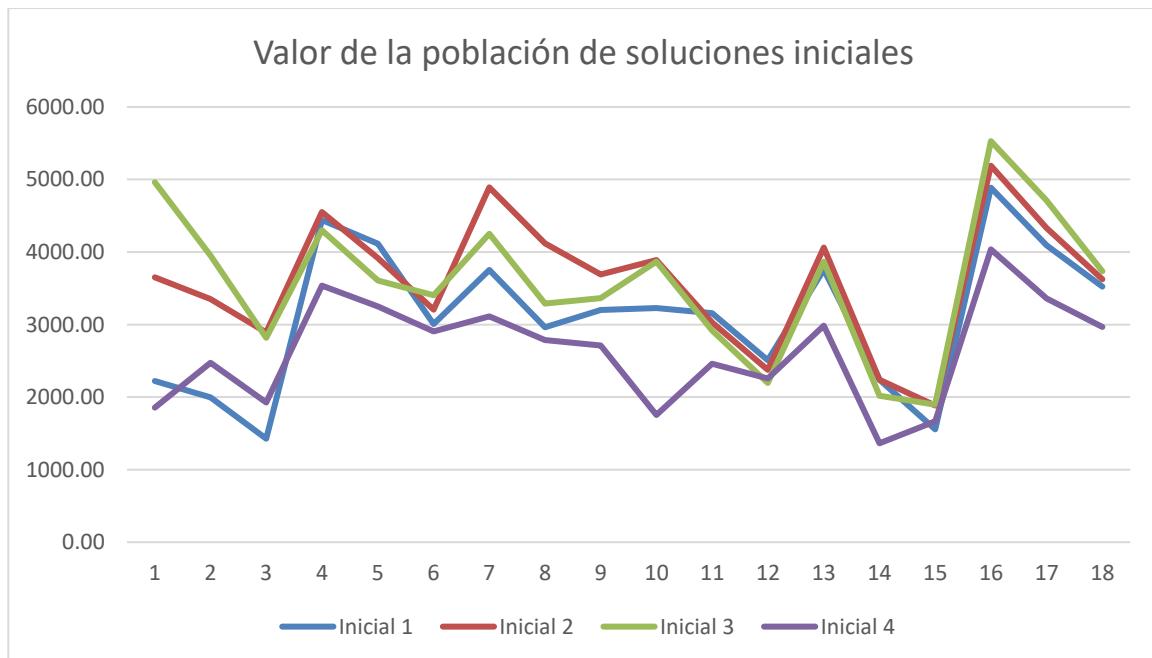


Ilustración 10 Gráfico del valor de la población inicial para cada instancia.

El siguiente gráfico muestra el número de veces en el “eje y” en el que una solución inicial ha quedado, siendo los puestos del 1º al 4º. Por ejemplo, la solución inicial obtenida a raíz del algoritmo 4 ha sido la mejor en 14 ocasiones tras finalizar el programa.

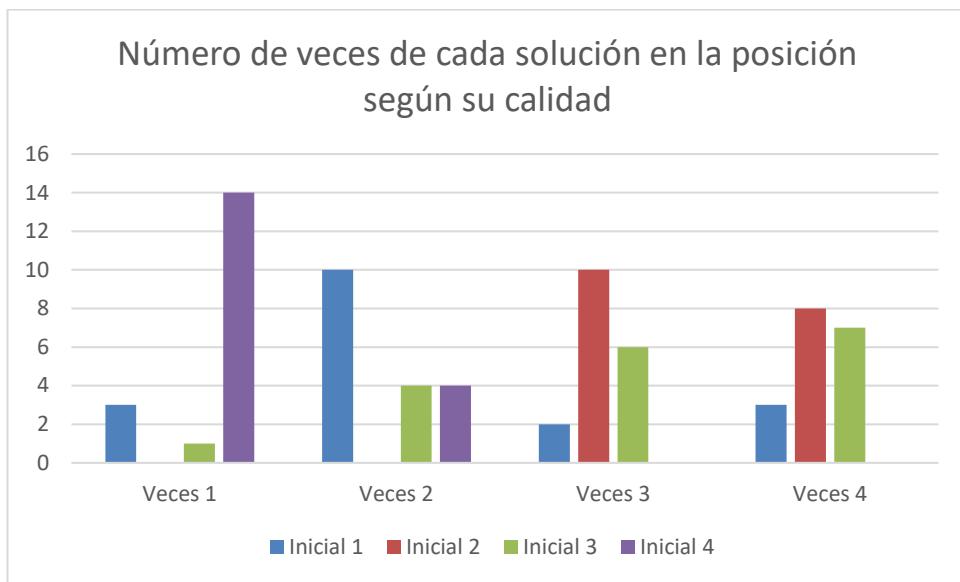


Ilustración 11 Gráfico del número de veces de la población inicial en un puesto.

Este gráfico muestra el número de veces en el “eje y” en el que una solución inicial que era la mejor/peor/otra ha terminado siendo la mejor al finalizar el programa. De este modo, sabemos que en 13 ocasiones la solución inicial que era la mejor al comienzo del programa ha terminado siendo la mejor.

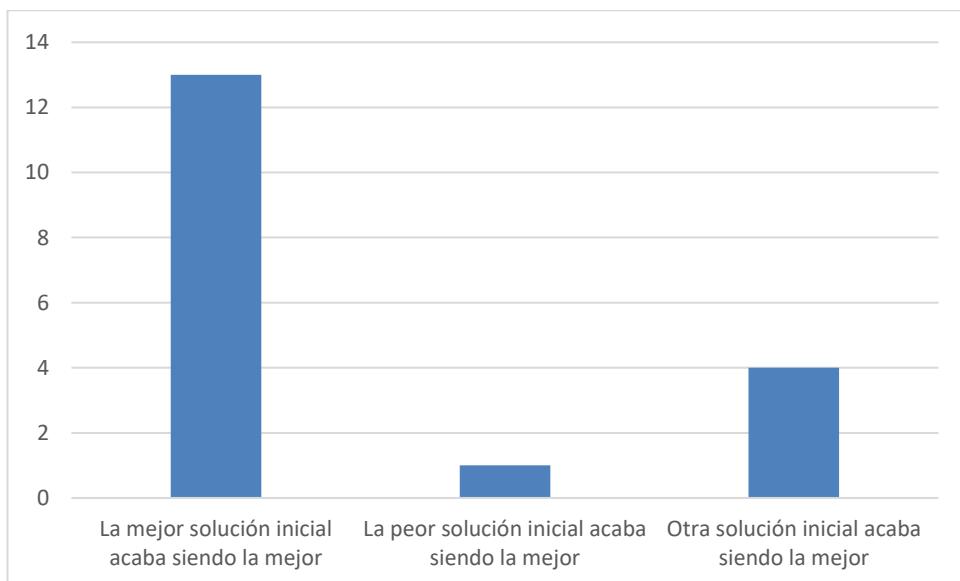


Ilustración 12 Gráfico que relaciona la solución inicial con la mejor solución final.

De estos gráficos podemos concluir que, de los métodos usados, el mejor para obtener una solución inicial corresponde al último algoritmo, cuyo criterio es seleccionar como primer cliente aquel con una fecha inicial de entrega menor. Por tanto, se puede decir que es más importante el poder entregar antes la mercancía que la cercanía al cliente. En cuanto al algoritmo que utiliza como criterio buscar al cliente más lejano, se puede concluir que no es una solución inicial buena.

Tras estas conclusiones sobre la solución inicial, no se debe pensar en trabajar con una única solución de partida. Aproximadamente un tercio de las soluciones finales han sido obtenidas de soluciones de partida que no eran las mejores, incluso en un caso se trataba de la peor. Esto demuestra que el algoritmo trata de no quedarse en un óptimo local.

### 6.3 Resultados y conclusiones del porcentaje de mejora.

En este apartado se exponen diferentes gráficos relacionados con los porcentajes de mejora de las soluciones. El objetivo es realizar un análisis del funcionamiento del programa en cuanto a la mejora de las soluciones.

Recordemos que el % Mejora x se calcula como la diferencia entre “Inicial x” y “Mejorada x”, y todo ello dividido por “Inicial x”.

Las ilustraciones que se encuentran a continuación representan en el “eje y” el porcentaje de mejora medio de cada criterio e instancia. De este modo, en el “eje x” se representa el resultado medio para los 3 criterios de cada instancia, siendo el número total de instancias de 3 por cada tipo (C1, RC1, etc.) haciendo un total de 9 puntos representados en este eje. Por tanto, para las tres instancias de cada tipo se tendrían los rangos 1-3 para la primera instancia, 4-6 para la segunda y 7-9 para la tercera.

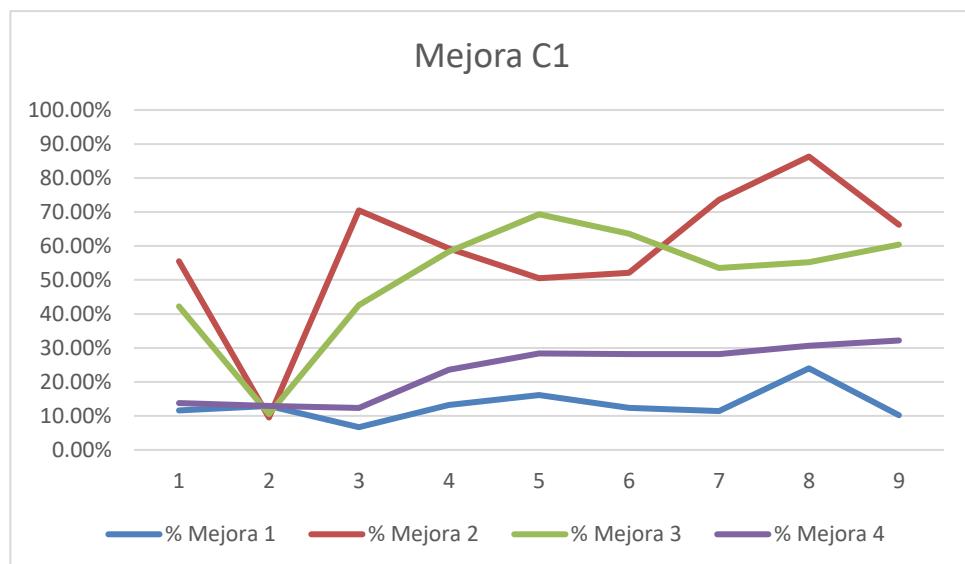


Ilustración 13 Porcentaje de mejora para las instancias de tipo C1

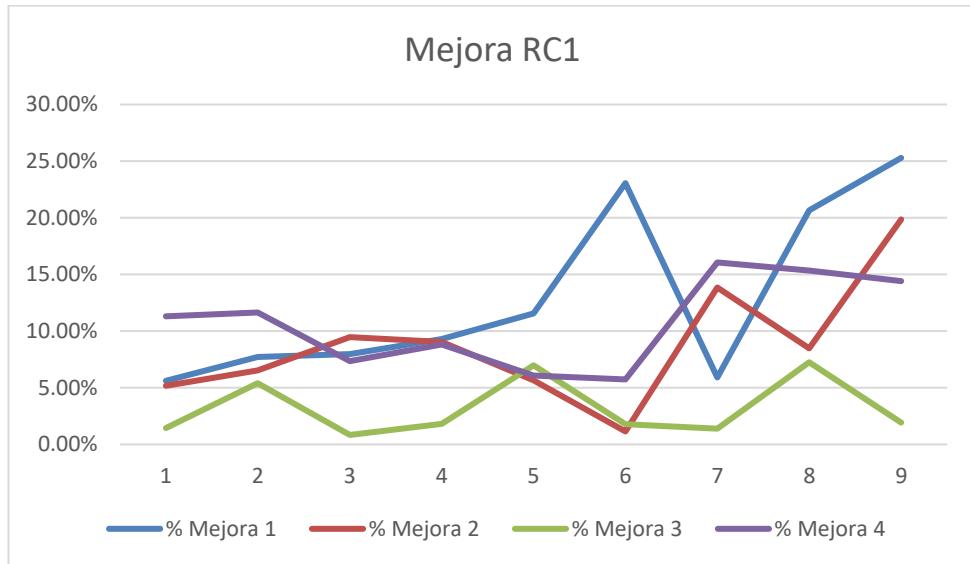


Ilustración 14 Porcentaje de mejora para las instancias de tipo RC1

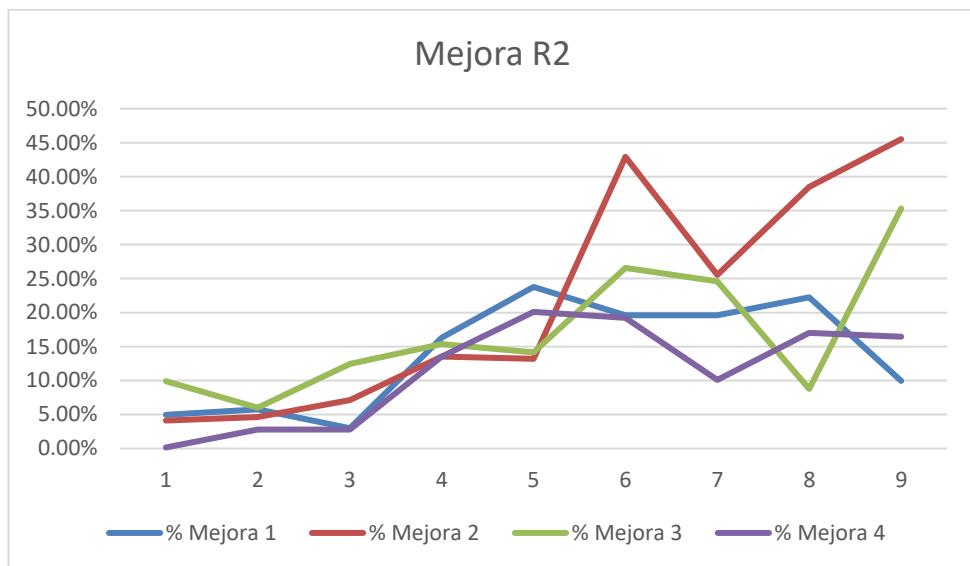


Ilustración 15 Porcentaje de mejora para las instancias de tipo R2

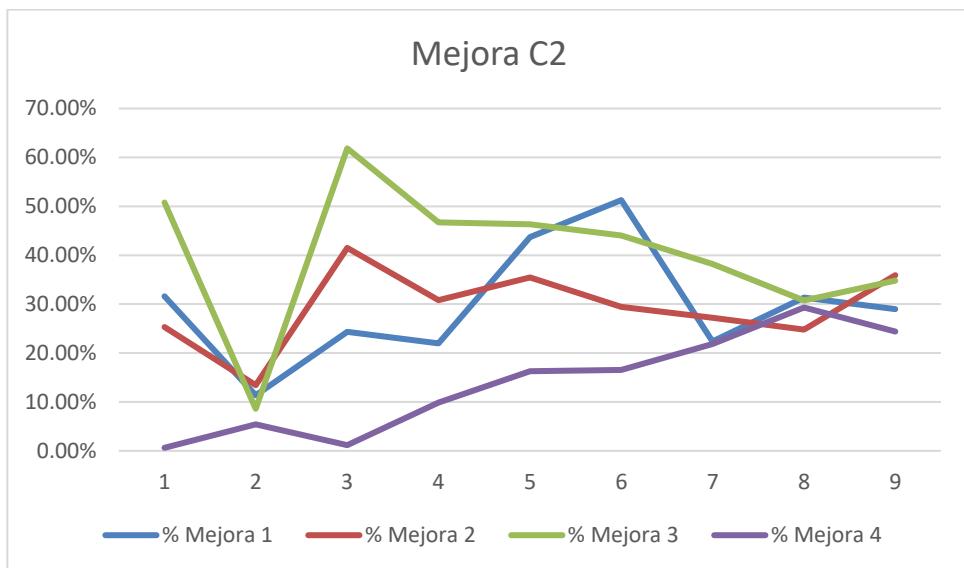


Ilustración 16 Porcentaje de mejora para las instancias de tipo C2

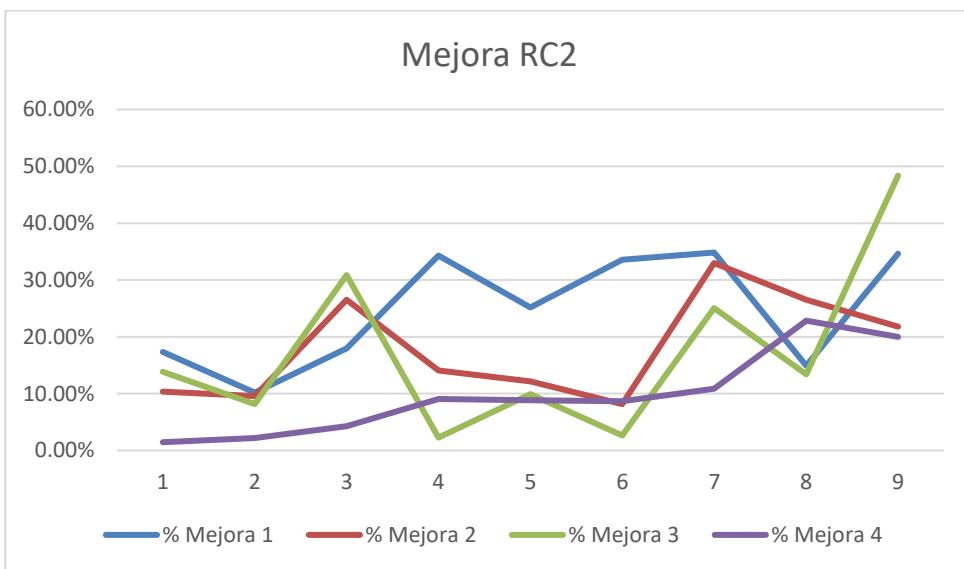


Ilustración 17 Porcentaje de mejora para las instancias de tipo RC2

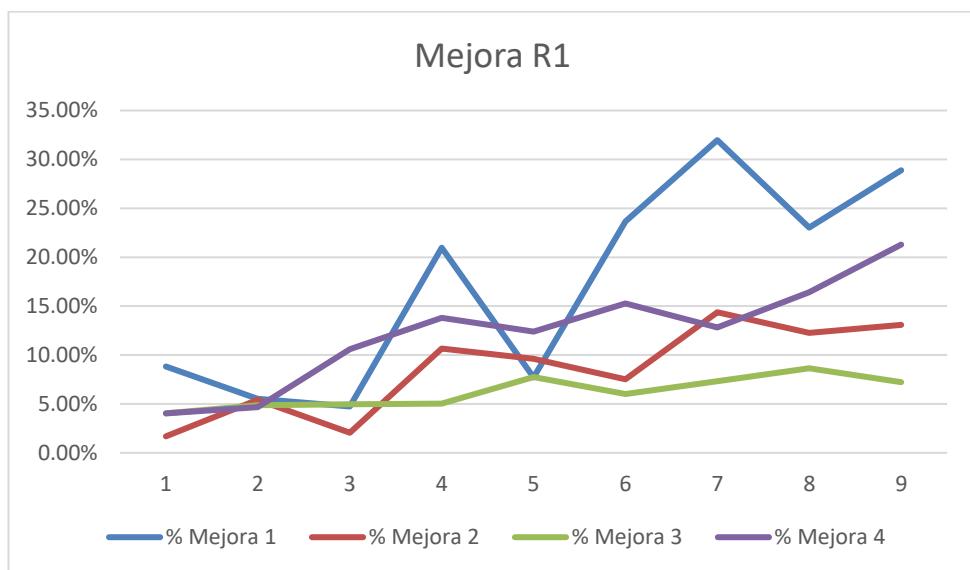


Ilustración 18 Porcentaje de mejora para las instancias de tipo R1

A continuación, se representa el porcentaje de mejora medio para cada solución inicial por tipo de instancia.

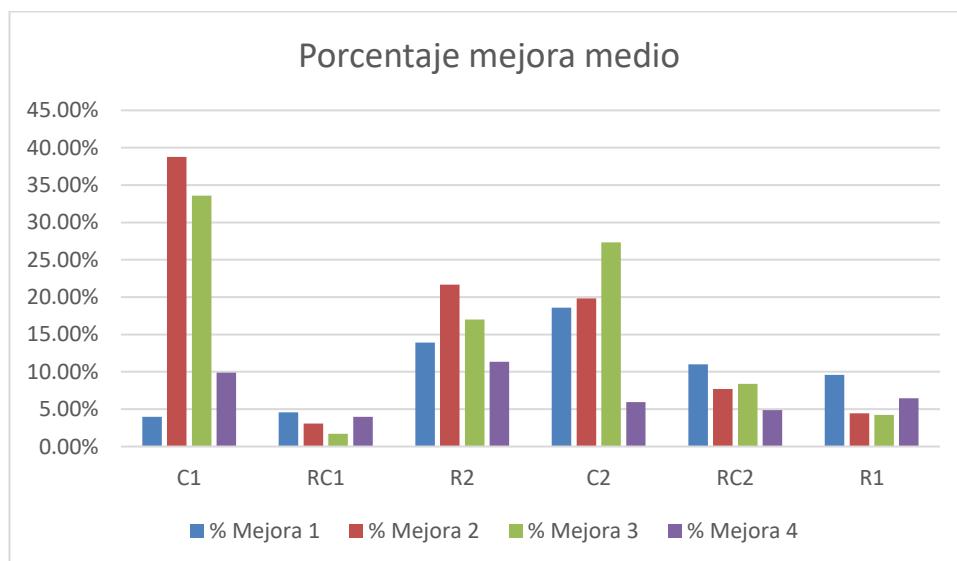


Ilustración 19 Porcentaje de mejora medio para cada instancia

En las siguientes ilustraciones se puede observar el porcentaje de mejora medio de cada solución según la instancia y el criterio (viene definido en el título del gráfico).

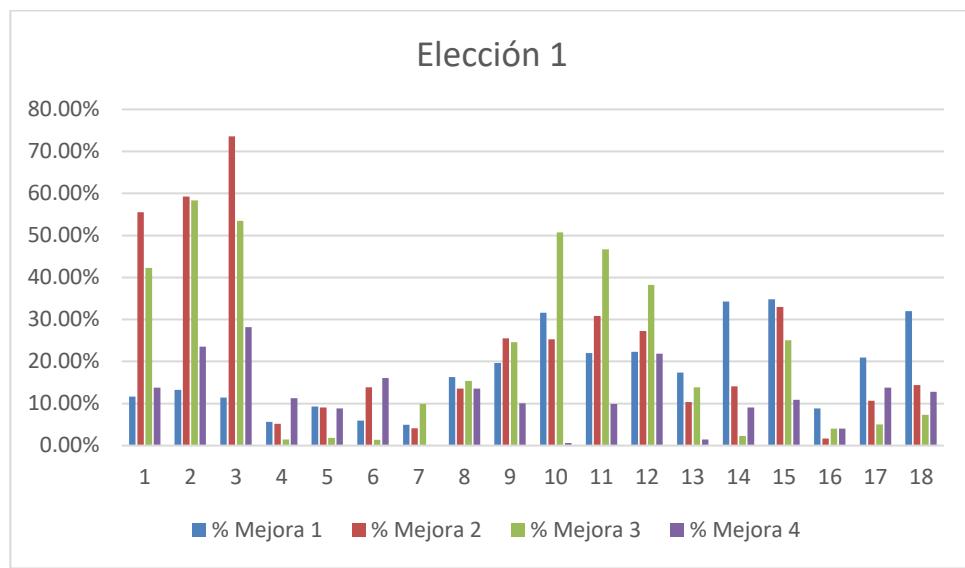


Ilustración 20 Porcentaje de mejora según el criterio Elección=1

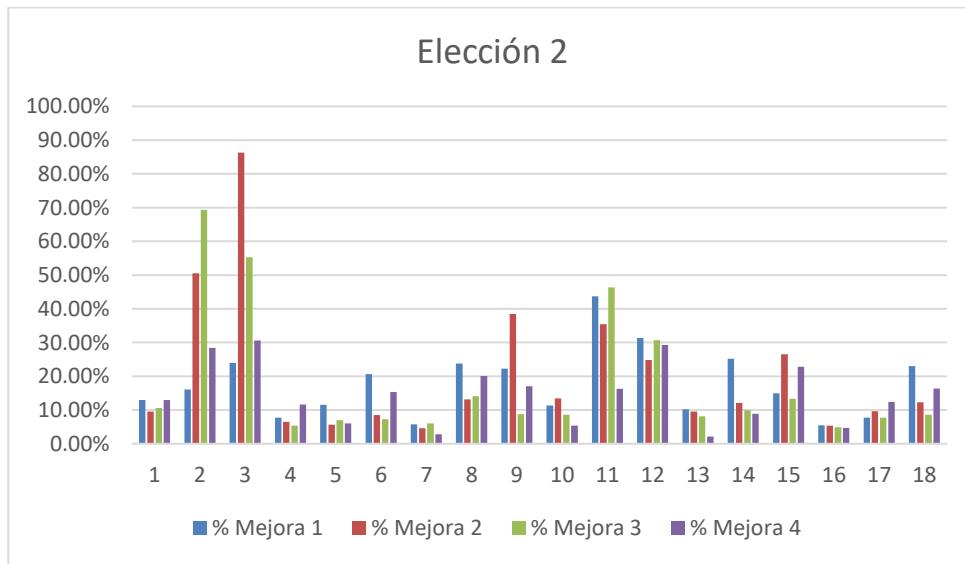


Ilustración 21 Porcentaje de mejora según el criterio Elección=2

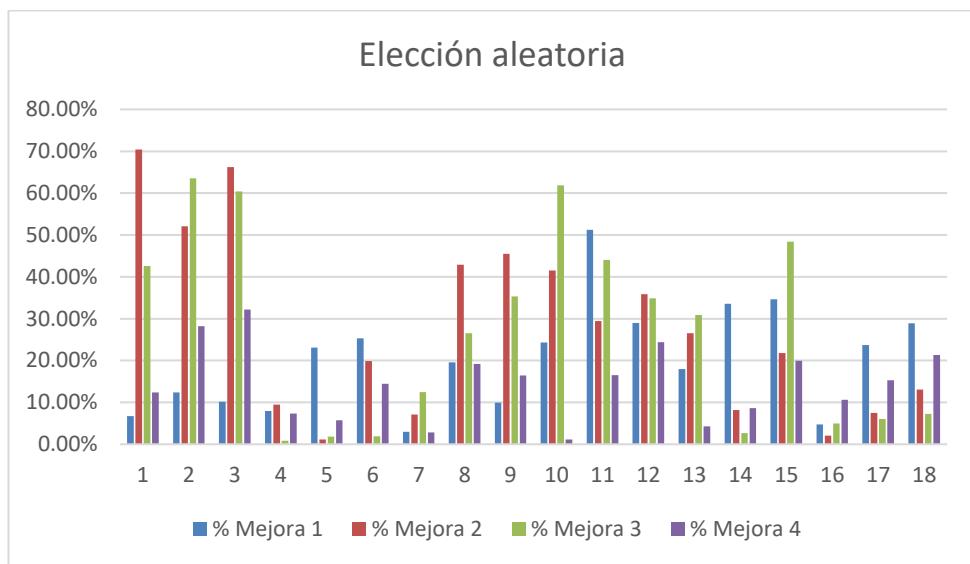


Ilustración 22 Porcentaje de mejora según el criterio Elección=aleatorio (1,3)

Gráfica que representa para cada solución inicial, la media de los porcentajes de mejora según el criterio escogido.

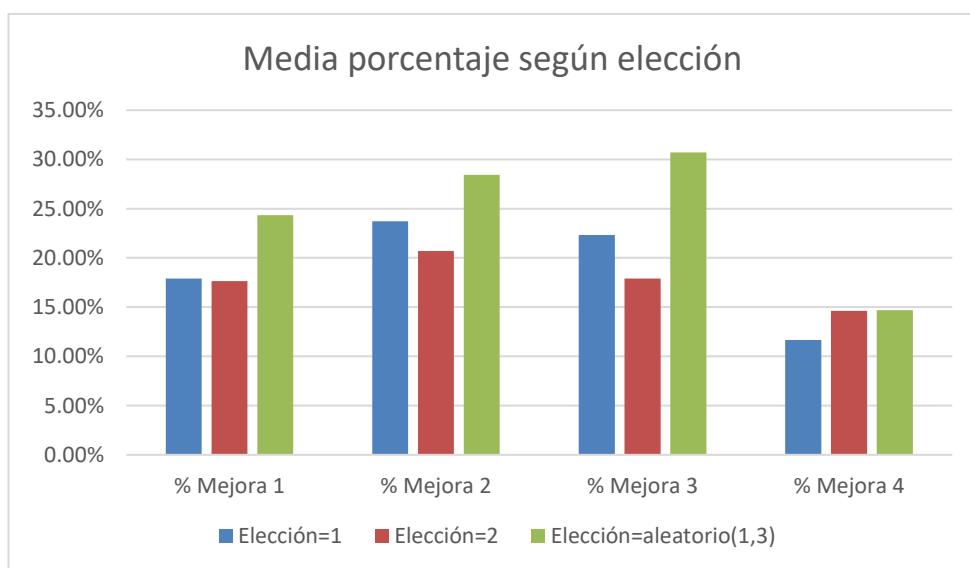


Ilustración 23 Porcentaje de mejora medio según el criterio de elección

Representación de la media de porcentaje de mejora para cada solución inicial según el tipo de instancia y criterio (el criterio viene definido en el título del gráfico).

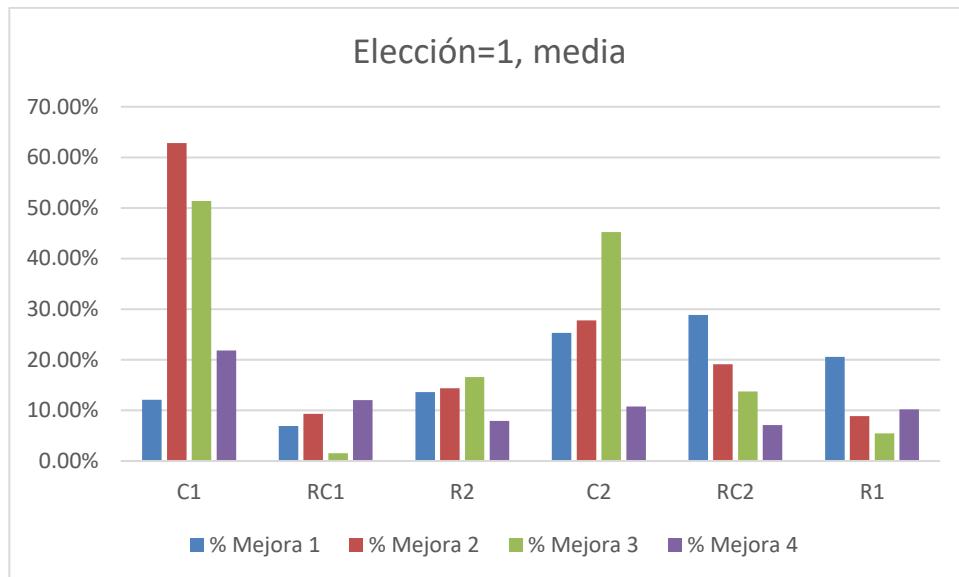


Ilustración 24 Porcentaje de mejora medio del tipo de instancia para el criterio Elección=1

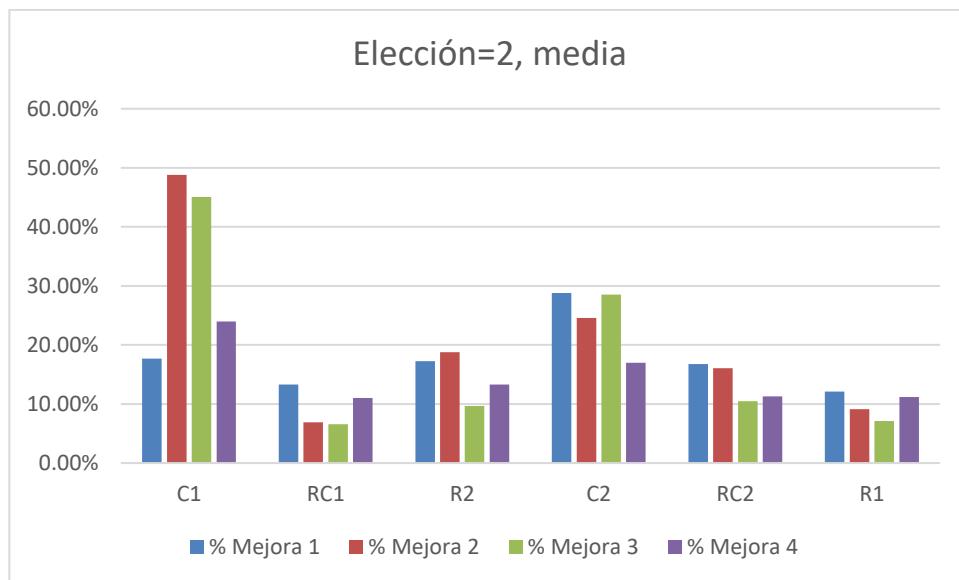


Ilustración 25 Porcentaje de mejora medio del tipo de instancia para el criterio Elección=2

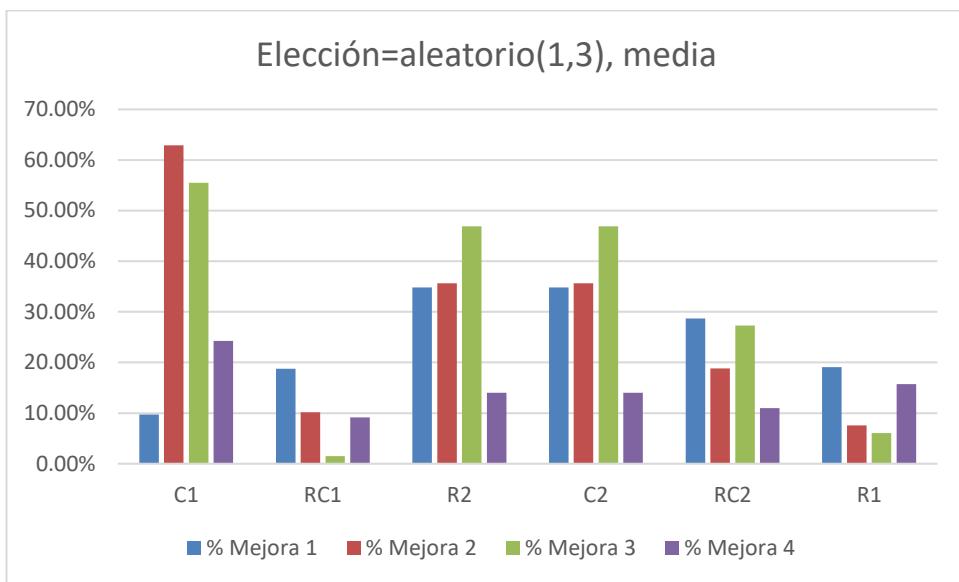


Ilustración 26 Porcentaje de mejora medio del tipo de instancia para el criterio Elección=aleatorio (1,3)

Se puede observar que en general, el porcentaje de mejora medio es mayor para los problemas de clientes agrupados, seguidos de los de tipo R2 y RC2. La mejora llega en un caso a ser casi del 40%, sin embargo, estas mejoras elevadas se dan para unas soluciones iniciales no tan buenas como la solución inicial cuatro. Para esta última solución (la cuatro), el porcentaje de mejora se mueve entre el 5% y el 10% aproximadamente, un porcentaje que podríamos decir que es un poco bajo, sobre todo si se parte de una solución inicial lejana a la óptima.

Por norma general, aunque los distintos criterios de selección de soluciones por los agentes optimizadores tienen resultados similares, tras la experimentación llevada a cabo, el mejor criterio es “elección=aleatorio (1,3)” para todos los tipos de problema excepto las instancias RC1.

En resumen, la mejor opción es elegir de manera aleatoria entre las siguientes opciones; escoger la mejor solución disponible (elección = 2), escoger la segunda peor solución (elección = 3) y escoger una totalmente aleatoria (elección = 1). Una posible explicación de estos resultados es que al incluir la posibilidad de que el agente Solution Manager se quede con la segunda peor solución, se pueden evitar óptimos locales y la solución pueda mejorar más. De los tres criterios tenidos en cuenta, el peor ha sido escoger siempre la mejor opción.

A continuación, a modo recordatorio se exponen los distintos criterios utilizados:

- Elección = 1: escoger de manera aleatoria una solución proporcionada por los agentes optimizadores.
- Elección =2: escoger la mejor solución (menor valor objetivo) proporcionada por los agentes optimizadores.
- Elección = aleatorio (1,3) de manera aleatoria se escoge:
  - Elección = 1.
  - Elección = 2.
  - Elección = 3: escoger la segunda peor solución proporcionada por los agentes optimizadores.

## 6.4 Resultados y conclusiones de la solución final.

La siguiente tabla es una comparativa entre los resultados encontrados con un método exacto (Relajación Lagrangiana), obtenidos del documento de [33], y los resultados obtenidos de la experimentación con un sistema multiagente. Hay que tener en cuenta que la función objetivo era distinta en [33] y, por tanto, se ha tenido que añadir un valor de 100 um por cada vehículo o ruta a los valores de dicho documento.

Instancias	Método exacto, solución encontrada	Tiempo ejec	Resultado multiagentes(res1)	Tiempo ejec	Resultado Multiagentes/Resultado Método exacto
<b>C101.50</b>	862.40	0.50	1551.32	43.02	1.80
<b>C102.50</b>	861.40	1.50	1674.99	43.32	1.94
<b>C103.50</b>	861.40	12.40	1084.69	44.59	1.26
<b>RC101.50</b>	1650.02	1.70	3021.22	42.59	1.83
<b>RC102.50</b>	1419.90	1029.80	2887.41	44.34	2.03
<b>RC103.50</b>	1243.13	17.10	2247.36	43.57	1.81
<b>R201.50</b>	1388.43	9.20	2484.30	43.29	1.79
<b>R202.50A</b>	1192.74	1707.70	2191.96	43.56	1.84
<b>R205.50</b>	1166.60	55507.50	2165.25	43.34	1.86
<b>C201.50A</b>	660.20	1.20	1580.67	41.87	2.39
<b>C202.50A</b>	660.20	18.80	1445.12	44.05	2.19
<b>C203.50</b>	659.80	167.10	1233.92	44.42	1.87
<b>RC201.50</b>	1170.15	61.80	2708.30	43.53	2.31
<b>RC201.25</b>	656.65	0.60	1161.24	13.26	1.77
<b>RC202.25A</b>	590.41	6351.70	1014.02	13.43	1.72
<b>R101.50</b>	2243.37	0.70	3518.37	41.94	1.57
<b>R102.50</b>	2009.00	3.50	2736.56	43.07	1.36
<b>R103.50</b>	1665.95	16.20	2217.92	43.06	1.33
<b>Media</b>					1.82

Tabla 20 Comparación de las soluciones obtenidas con las óptimas encontradas.

En las siguientes ilustraciones se puede observar la evolución para la instancia C103.50. El círculo rojo corresponde al almacén, mientras que los amarillos representan los clientes. En azul se pueden observar los primeros clientes de cada ruta. Las líneas azules simulan la ruta que sigue el vehículo.

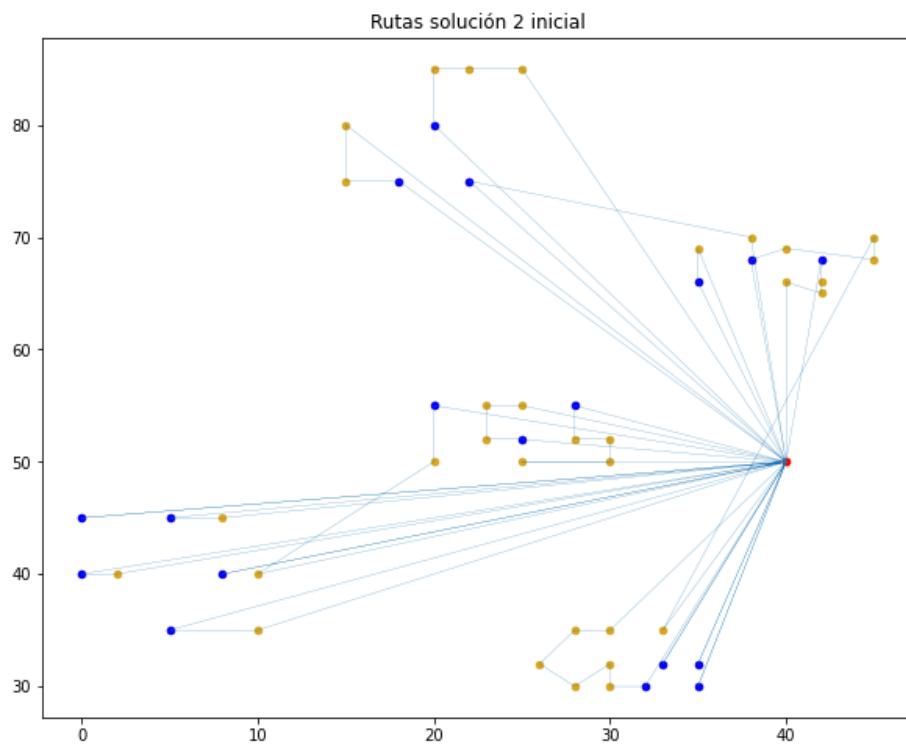


Ilustración 27 Representación gráfica de la solución inicial 2 para la instancia C103.50

Inicialmente esta solución tiene 18 rutas como se puede observar en la ilustración.

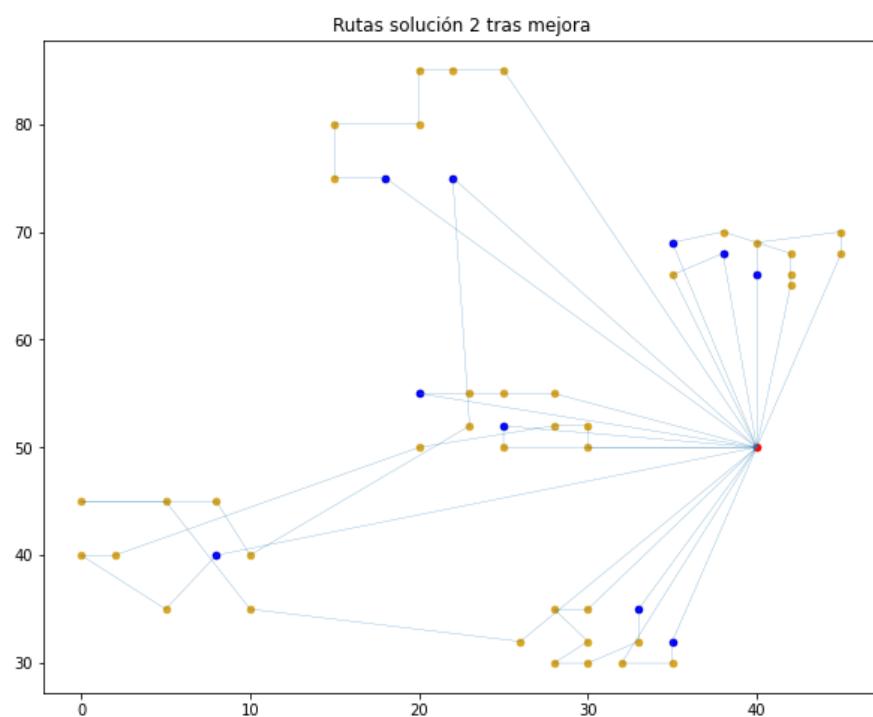


Ilustración 28 Representación gráfica de la solución final 2 para la instancia C103.50

Tras la mejora, el número de rutas ha sido disminuido de 18 a 10 rutas, lo que hace un total de 8 rutas ahorradas. Además de este ahorro, se pueden percibir ciertos cambios en las rutas, clientes pertenecientes a una ruta ahora se encuentran en otra. Este ejemplo muestra cómo el sistema de multiagentes cumple su función. Gracias a los distintos agentes pueden eliminarse rutas, intercambiar clientes de rutas diferentes e incluso cambiar el orden de los clientes de una misma ruta, opciones que no se pueden dar al utilizar un único algoritmo o agente.

En este caso, una solución encontrada en una experimentación para la instancia C103.50 ha sido la siguiente:

Valor de la función objetivo: 1038.73 um

Número de rutas: 6

Tiempo de ejecución: 745.54 segundos

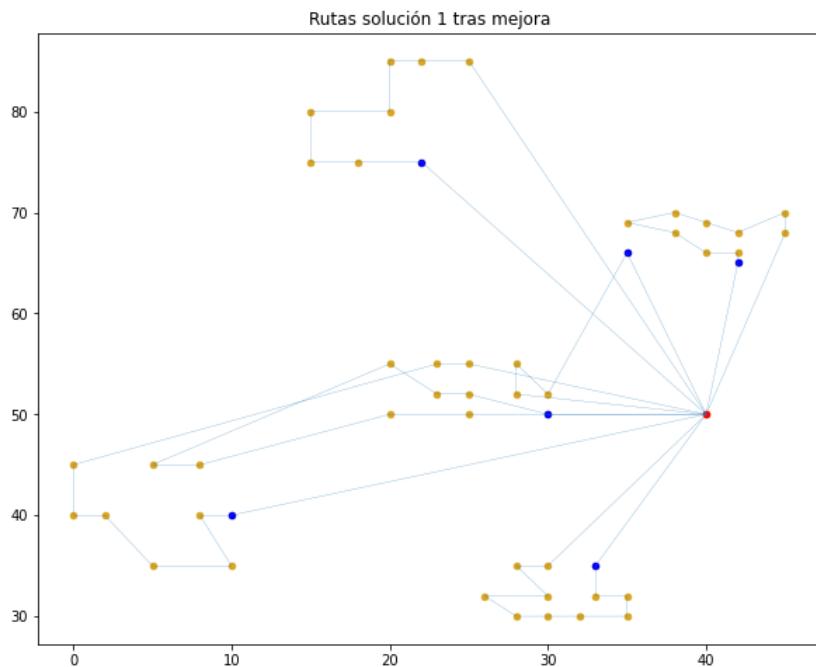


Ilustración 29 Representación gráfica de la solución final 1 para la instancia C103.50

Los resultados de la tabla 20 nos indican que el sistema de multiagentes nos proporciona una solución que aproximadamente duplica el coste de la solución óptima, además el tiempo de ejecución es en la mayoría de los casos superior. Sin embargo, para los problemas de tipo R1 y en la instancia C103.50 se observa que la calidad de las soluciones es bastante buena y se aproxima al óptimo.

El sistema de multiagentes requiere de un tiempo elevado, pero teniendo en cuenta de que los algoritmos utilizados como agentes no son algoritmos eficaces, son algoritmos que hemos realizado a partir de los conocimientos adquiridos en la carrera y en la realización del trabajo, se puede decir que el programa funciona correctamente y cumple sus funciones, c.q.d.

Además de esto, hay que tener en cuenta que el problema VRPTW es un problema difícil de resolver y que, para instancias con un número de clientes elevado, utilizar métodos como el empleado en [33] no proporciona una buena solución o directamente no resuelven el problema (se ha tenido que sustituir algunas instancias por otras al no disponer de la solución óptima). Sin embargo, el sistema de multiagentes es capaz de resolver ese tipo de instancias ya que se emplean heurísticas.

Finalmente, se puede decir que al menos hasta 50 clientes no es necesario utilizar un sistema multiagente. Sin embargo, para las instancias que no son tipo clustered comienza a ser necesario el empleo de al menos una

heurística o metaheurística, pues se puede observar en [33] que hay instancias que no están resueltas y otras en las que se emplea un tiempo elevado con los métodos exactos. También hay que destacar que los algoritmos utilizados son algoritmos que se han creado en base a otros que sí funcionan mejor, pero que han sido modificados para la realización del programa.

## 6.5 Propuestas de mejora para el sistema multiagente.

Para mejorar el funcionamiento del sistema multiagente se proponen los siguientes cambios:

- OA Vecinos, no varía.
- OA Insertion, añadir que, si no encuentra el algoritmo una solución, genere un número aleatorio nuevo hasta recorrer todos los clientes de la ruta.
- OA String Cross, sustituir por un algoritmo de búsqueda tabú similar.
- OA Intercambio, sustituir por un algoritmo de búsqueda tabú similar.
- OA Buscar solitario, sustituir por un algoritmo que, de una ruta aleatoria compruebe cliente por cliente de esa ruta si hay un cliente cercano (definir una distancia) perteneciente a otra ruta y tratar de insertarlo antes o después (comprobar cuál de las dos opciones sería mejor).

Después de realizar estos cambios, el tiempo de ejecución de cada agente aumentaría porque trataríamos con agentes más complejos. Sin embargo, el sistema multiagente podría mejorar y sin duda, sería mejor opción que utilizar un único algoritmo ya que el abanico de posibilidades que ofrece es mucho mayor. El sistema de multiagentes permite el intercambio de clientes entre diferentes rutas y el de una misma, además permite no caer en un óptimo local gracias a la comunicación entre los agentes optimizadores y Solution Manager (podemos con criterios hacer que se contemple la posibilidad de escoger una solución peor y así evitar óptimos locales, además de disponer de una población de soluciones inicial).

Otro cambio para realizar sería la sustitución del algoritmo de la solución inicial 3 por un algoritmo nuevo. Este algoritmo propone utilizar el algoritmo de la población inicial cuatro, pero en vez de buscar el cliente más cercano una vez añadido un primer cliente a la ruta, propone introducir en la ruta al cliente cuyo valor de la ecuación  $distancia + \frac{fecha\ de\ inicio\ de\ la\ entrega}{1000}$  sea menor. Este nuevo algoritmo no se usará en el programa, pero se ha hecho una prueba para ver si la solución inicial es mejor que el resto y se ha observado que en trece ocasiones ha sido así. Aunque no se utilice en el programa, si se probará en las instancias C201.50, C202.50 y RC201.50 para ver si se consigue una mejora respecto al resultado final, estos resultados se obtendrán siguiendo el criterio “elección=aleatorio (1,3)”. Con las tablas 21 y 22 se concluye que, para obtener una solución inicial, un método que pondere la distancia de un cliente a otro junto con la fecha inicial de entrega (el mínimo de la ventana temporal) es mejor en la mayoría de las ocasiones que métodos en los que se excluya a uno de los dos (distancia y tiempo).

	C101.50	C102.50	C103.50	RC101.50	RC102.50	RC103.50	R201.50	R202.50 <sup>a</sup>	R205.50
Inicial 1	2220.90	1997.34	1427.08	4440.99	4116.95	3008.03	3754.13	2961.83	3201.02
Inicial 2	3650.40	3353.34	2890.85	4552.80	3914.72	3204.26	4890.56	4120.89	3692.4845
Inicial nuevo algoritmo	1695.11	1900.14	2353.68	3775.99	3245.76	2711.01	2831.09	2640.00	2529.62
Inicial 4	1857.55	2474.18	1925.50	3538.66	3250.66	2905.55	3112.53	2787.14	2710.14

Tabla 21 Soluciones iniciales con nuevo algoritmo, parte 1

	C201.50A	C202.50A	C203.50	RC201.50	RC201.25	RC202.25A	R101.50	R102.50	R103.50
Inicial 1	3227.87	3155.24	2510.82	3756.96	2237.63	1556.21	4886.90	4093.94	3525.19
Inicial 2	3889.33	3030.95	2374.42	4063.63	2239.04	1884.39	5189.93	4330.43	3626.70
Inicial nuevo algoritmo	1612.65	2005.30	2137.29	3335.62	1200.69	1664.56	3901.60	3324.69	3231.53
Inicial 4	1754.561309	2461.328499	2258.81	2985.49	1364.01	1664.75	4036.71	3361.53	2968.47

Tabla 22 Soluciones iniciales con nuevo algoritmo, parte 2

Instancias	Mejor resultado	Tiempo ejec	Resultado multiagentes algoritmo 3 (res1)	Resultado Multiagentes/Resultado Método exacto	Resultado multiagentes nuevo algoritmo	Resultado Multiagentes/Resultado Método exacto	% Mejora sobre res1
C201.50A eleccion=aleatorio	660.20	1.20	1871.55	2.60	1449.83	2.20	22.53%
C202.50A eleccion=aleatorio	660.20	18.80	1529.55	2.32	1499.25	2.27	1.98%
RC201.50 eleccion=aleatorio	1,170.15	61.80	2708.30	2.31	3169.62	2.71	-17.03%

Tabla 23 Comparación resultados del algoritmo 3 con el nuevo algoritmo, todos con el criterio “elección=aleatorio (1,3)”

Instancias	Mejor resultado	Tiempo ejec	Mejor Resultado multiagentes (res1)	Resultado Multiagentes/Resultado Método exacto	Resultado multiagentes nuevo algoritmo	Resultado Multiagentes/Resultado Método exacto	% Mejora sobre res1
C201.50A eleccion	660.20	1.20	1580.67	2.39	1449.83	2.20	8.28%
C202.50A	660.20	18.80	1445.12	2.19	1499.25	2.27	-3.75%
RC201.50	1,170.15	61.80	2708.30	2.31	3169.62	2.71	-17.03%

Tabla 24 Comparación mejores resultados encontrados con el nuevo algoritmo, este último con el criterio “elección=aleatorio (1,3)”

Tras los resultados obtenidos de la tabla 23 y 24 se puede concluir que, conseguir una mejor solución inicial no implica obtener mejores resultados finales, al menos en las instancias en las que se ha hecho la comprobación. Por tanto, se propone para un futuro trabajo, realizar las mejoras de los agentes optimizadores mencionados anteriormente para comprobar si mejoran los resultados. Tras realizar los cambios mencionados, se podría probar de nuevo con el algoritmo propuesto para obtener una solución inicial.

## 7 CONCLUSIONES GENERALES

---

El primer paso para comenzar con la realización del sistema multiagente es conseguir una solución inicial. Para ello se han utilizado métodos constructivos con la restricción de número de vehículos relajado, es decir, no se ha puesto límite al número de vehículos o rutas de la solución. Tras conseguir los resultados para distintos criterios, se decidió probar a utilizar un criterio que incluyera la distancia y el tiempo de entrega de manera ponderada (se ha comprobado que el programa no mejora con este único cambio) con el cual se ha podido concluir que en el VRPTW es importante tener en cuenta ambas partes (distancia y tiempo).

El segundo paso es seleccionar los agentes que se van a utilizar en el sistema. Por un lado, están los agentes encargados de optimizar las soluciones y por otro, el agente encargado de proporcionar y extraer soluciones al resto. La elección de los agentes es muy importante y se ha comprobado que en el trabajo estos agentes hay que cambiarlos si se quieren obtener unos mejores resultados. En cuanto al intercambio de información, se ha demostrado que el peor criterio a seguir de los escogidos es el de seleccionar siempre la mejor solución que proporcionan los agentes, pues no evita óptimos locales.

Por último, una vez realizado el programa, hay que decidir el criterio de parada de este. En este trabajo se ha decidido realizar 300 interacciones entre el agente SolutionManager y los agentes optimizadores, esto equivale a unos 42 segundos aproximadamente de tiempo de ejecución para instancias de 50 clientes. Sin embargo, este criterio podría cambiarse por otro sin ningún problema, por ejemplo, que el valor de la última columna de la tabla 22 (Resultado multiagentes/Resultado método exacto) fuera menor o igual a cierto valor además de una condición que controle el tiempo por si nunca se llega a dicho objetivo (hay que tener en cuenta que para la realización del trabajo se han hecho 1620 experimentaciones y, por tanto, se dispone de poco tiempo para cada experimentación).

Con esto, queda diseñado un sistema multiagente para abordar el problema de rutado de vehículos con ventanas temporales con un almacén y cualquier cantidad de clientes. Este sistema se ha programado en lenguaje Python y se han realizado diversos experimentos con instancias de hasta 50 clientes. Se han propuesto además algunas mejoras sobre el algoritmo original.

# REFERENCIAS

---

- [1] BBVA, «The COVID-19 impact on Consumption in Real Time and High Definition A Big Data BBVA Research Project,» 2020.
- [2] D. L. Applegate, R. E. Bixby, V. Chvátal y W. J. Cook, *The Traveling Salesman Problem*, 2006, p. 12.
- [3] G.B.Dantzig y J.H. Ramser, «The truck dispatching problem,» *Management Science*, vol. 6, nº 1.
- [4] B. Golden, S. Raghavan y E. Wasil, de *The Vehicle Routing Problem: Latest Advances and New Challenges*, Springer, 2008, pp. 3-6.
- [5] R. Cordone y R. W. Calvo, «A Heuristic for the Vehicle Routing Problem,» *Journal of Heuristics*, pp. 107-129, 2001.
- [6] J. M. Wooldridge, *An introduction to multiagent systems*, 2 ed., John Wiley & Sons Ltd, 2009.
- [7] D. Barbucha y P. Jedrzejowicz, *An Agent-Based Approach to Vehicle Routing*, vol. 1, 2007.
- [8] D. Barbucha, I. Czarnowski, P. Jedrzejowicz, E. Ratajczak y I. Wierzbowska, JADE-based A-Team as a tool for implementig population-based algorithms, vol. 3, 2006.
- [9] L. B. Rocha, E. C. González y J. A. Orjuela, Una revisión al estado del arte del problema de ruteo de vehículos: Evolución histórica y métodos de solución., vol. 16, 2011, pp. 35-55.
- [10] E. M. Orozco, *Metaheurísticas para el problema de ruteo de vehículos con ventanas de tiempo*, 2017.
- [11] R. Martí, *Procedimientos Metaheurísticos en Optimización Combinatoria*.
- [12] M. M. Solomon, «Algorithms for the Vehicle Routing and Scheduling Problems with Time Windows Constraints,» 1984.
- [13] G. Clarke y J. Wright, «Scheduling of vehicles from a central depot to a number of delivery,» *Operations Research*, vol. 12, nº 4, p. 568–581, 1964.
- [14] S.Lin y B. W. Kernighan, «An Effective Heuristic Algorithm for the Traveling-Salesman Problem,» *Operations Research*, vol. 21, nº 2, pp. 498-516, 1973.
- [15] I. Or, *Traveling Salesman-Type Combinatorial Problems and their relation*, 1976.
- [16] M. Gendreau, A. Hertz y G. Laporte, «New Insertion and Postoptimization Procedures for the Traveling Salesman Problem,» *Operations Research*, vol. 40, nº 6, p. 1086, 1992.
- [17] P. M. Thompson y H. N. Psaraftis, «Cyclic Transfer Algorithms for Multivehicle Routing and Scheduling

Problems,» *Operations Research*, vol. 41, nº 5, pp. 935-946, 1993.

- [18] A. Olivera, *Heurísticas para Problemas de Ruteo de Vehículos*, Montevideo, 2004.
- [19] V. Breedam, «Improvement heuristics for the vehicle routing problem,» *European Journal of Operational Research*, vol. 86, nº 3, pp. 480-490, 1995.
- [20] U. P. d. Valencia. [En línea]. Available: <https://optimizacionheuristica.blogs.upv.es/2015/02/22/que-son-las-metaheuristicas/>.
- [21] I. Osman y J. Kelly, «Meta-Heuristics: Theory & Applications,» Kluwer Academic Publishers, 1996.
- [22] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1989.
- [23] S. Kirkpatrick, C. D. Gelatt y M. P. Vecchi, «Optimization by Simulated Annealing,» vol. 280, pp. 671-680, 1983.
- [24] M. Dorigo, *Optimization, Learning and Natural Algorithms*, Ph.D. Thesis, Politecnico di Milano, Italian, 1992.
- [25] F. Glover, «Future paths for integer programming and links to artificial intelligence,» *Computers & Operations Research*, vol. 13, nº 5, pp. 533-549, 1986.
- [26] G. v. Rossum, Interviewee, *The Making of Python*. [Entrevista]. 2003.
- [27] «Spade,» [En línea]. Available: <https://spade-mas.readthedocs.io/en/latest/readme.html>.
- [28] «Asyncio,» [En línea]. Available: <https://docs.python.org/3/library/asyncio.html>.
- [29] «Aima Python Code,» [En línea]. Available: <http://aima.cs.berkeley.edu/python/readme.html>.
- [30] «Mesa: Agent-based modeling in Python 3+,» [En línea]. Available: <https://mesa.readthedocs.io/en/master/>.
- [31] «VRP-REP,» [En línea]. Available: <http://www.vrp-rep.org/variants/item/vrptw.html>.
- [32] O. Bräysy y M. Gendreau, «Vehicle Routing Problem with Time Windows,» *TRANSPORTATION SCIENCE*, vol. 39, nº 1, pp. 104-118, 2005.
- [33] B. Kallehauge, J. Larsen y O. Madsen, «Lagrangian Duality Applied on Vehicle Routing with Time Windows Experimental Results,» 2001.



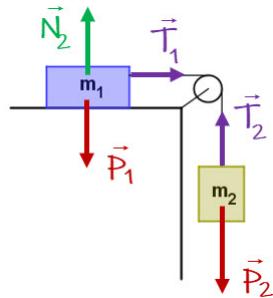
Nombre y Apellido:.....  
sección:

## EXAMEN I

**Problema 1.** Un vagón de 20 kg es jalado a nivel del suelo por una cuerda inclinada 30 grados sobre la horizontal. Una fuerza de fricción de 30 N se opone al movimiento. ¿Cuánto mide la fuerza de atracción si el vagón se mueve con a)una rapidez constante y b) una aceleración de  $0.40 \text{ m/s}^2$  ?

### Problema 2

Determine la expresión de la aceleración de los bloques de la figura, donde las fuerzas de fricción son despreciables. ¿Cuál es la tensión en la cuerda que los une?



### Problema 3

Calcule el trabajo realizado en contra de la gravedad por una bomba que descarga 600 litros de gasolina dentro de un tanque que se encuentra a 20 m por encima de la bomba. Un centímetro cúbico de gasolina tiene una masa de 0.82 gramos.

Trabajo Fin de Máster  
Máster Universitario en Ingeniería Industrial

Optimización de Rutas de Vehículos de Recogida de  
Basuras mediante MILP

Autor: Francisco de Paula Pastrana Alcántara

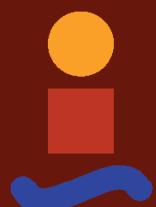
Tutor: María Rodríguez Palero

Dpto. Organización Industrial y Gestión de Empresas II

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2020





Trabajo Fin de Máster  
Máster Universitario en Ingeniería Industrial

# **Optimización de Rutas de Vehículos de Recogida de Basuras mediante MILP**

Autor:  
Francisco de Paula Pastrana Alcántara

Tutor:  
María Rodríguez Palero

Dpto. Organización Industrial y Gestión de Empresas II  
Escuela Técnica Superior de Ingeniería  
Universidad de Sevilla  
Sevilla, 2020



Trabajo Fin de Máster: Optimización de Rutas de Vehículos de Recogida de Basuras mediante MILP

Autor: Francisco de Paula Pastrana Alcántara

Tutor: María Rodríguez Palero

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal



*A mis abuelos*



# Agradecimientos

---

Me gustaría comenzar estas líneas agradeciendo a mi tutora, María, su dedicación, atención y ayuda para la elaboración de este proyecto. También dar las gracias a mis compañeras y amigas, Sara y Sandra, por las horas que hemos pasado juntas y por su apoyo durante esta etapa, sin ellas todo hubiera sido más difícil. No puedo olvidarme de mis amigos que me han acompañado desde que era pequeño y que sin duda forman parte de mi familia.

Este trabajo está dedicado a mis abuelos: Aurelia, Emilio y Tere, a los que tantas horas les he robado por tener “muchas cosas que hacer”, y a mi abuelo Paco, con quien no tuve la suerte de compartir mucho tiempo y sin embargo, siento tan cerca. A mis padres, Mayte y Paco, gracias por enseñarme lo que es el cariño y el amor, el dar sin esperar nada a cambio. Gracias por hacerme todo más llevadero, sin vosotros no habría sido capaz de terminar con esto, mi mayor suerte es ser vuestro hijo. A mi hermano Manu, la alegría de mi casa y del cual cada día estoy más orgulloso. Por último, a Inés, mi apoyo constante y mi compañera de viaje, gracias por ser la luz que me guía.

*Francisco de Paula Pastrana Alcántara*

*Sevilla, 2020*



# Resumen

---

En este trabajo se modela la recogida de residuos como un caso particular de la familia de problemas del *Vehicle Routing Problem* (VRP). Se desarrolla un modelo basado en *Mixed-Integer Linear Programming* (MILP) que se resolverá con el software CPLEX, programado en Python y ejecutado en Jupyter Notebook. Para el desarrollo del modelo se caracteriza el caso práctico objeto de estudio y se particularizan las formulaciones que previamente han demostrado su capacidad para resolver este tipo de problemas. El objeto sobre el que se aplica este modelo es la Provincia de Sevilla. En concreto, sobre las siete mancomunidades que la conforman. Se propone en cada mancomunidad la coordinación para la prestación del servicio integral de recogida de residuos. Es decir, todos los municipios adscritos a una de estas entidades tendrán una recogida conjunta. Se utilizarán datos reales de producción de residuos y entre otros resultados se ofrecerán las rutas recorridas por los vehículos obtenidas tras el proceso de optimización, graficadas sobre un mapa.



# **Abstract**

---

In this thesis, waste collection is modelled as a particular case of the Vehicle Routing Problem (VRP) family. A model based on Mixed-Integer Linear Programming (MILP) is developed and solved using CPLEX software. It is programmed in Python and executed in Jupyter Notebook. For the development of the model, the case of study is characterized and the formulations that have previously demonstrated their capacity to solve this type of problem are specified. The zone where this model is applied is the Province of Seville. Specifically, on the seven municipal associations that comprise it. The coordination for the waste collection service is proposed in each association. In other words, all the municipalities attached to one of these entities will have a joint collection. Real waste production data will be used and among other results the routes travelled by the vehicles obtained after the optimization process will be plotted on a map.



# Índice

---

<b>Agradecimientos</b>	i
<b>Resumen</b>	iii
<b>Abstract</b>	v
<b>Índice</b>	vii
<b>Índice de Tablas</b>	ix
<b>Índice de Figuras</b>	xi
<b>Lista de Abreviaturas</b>	xiii
<b>1 Introducción</b>	1
1.1 <i>Motivación</i>	1
1.2 <i>Objetivos del Proyecto</i>	1
1.3 <i>Metodología utilizada</i>	2
1.4 <i>Estructura del texto</i>	2
<b>2 Estado del Arte</b>	3
2.1 <i>Introducción histórica</i>	3
2.2 <i>Descripción del Problema</i>	4
2.2.1 Travelling Salesman Problem (TSP)	5
2.2.2 Vehicle Routing Problem (VRP)	7
2.3 <i>Métodos de Resolución</i>	8
2.3.1 Heurísticos	9
2.3.2 Metaheurísticos	11
2.3.3 Métodos exactos	13
<b>3 Presentación del Caso</b>	17
3.1 <i>Presentación del Plan de Residuos no Peligrosos de la Provincia de Sevilla</i>	17
3.2 <i>Modelo de gestión de residuos de la Provincia de Sevilla.</i>	17
3.2.1 Proceso de recogida de residuos	18
3.2.2 Unidades de Gestión de Residuos (UGRs) de la Provincia de Sevilla	19
3.2.3 Evolución hacia un Consorcio Provincial de Residuos	26
3.3 <i>Línea base para el desarrollo del Modelo</i>	28
<b>4 Desarrollo del Modelo</b>	33
4.1 <i>Hipótesis simplificativas y suposiciones</i>	33
4.2 <i>Primera aproximación</i>	34
4.2.1 Notación matemática	35
4.2.2 Función Objetivo	37
4.2.3 Gestión del flujo	37
4.2.4 Gestión de la demanda	38
4.2.5 Capacidad	39
4.2.6 Distancia	39
4.2.7 Tiempo	40
4.3 <i>Consolidación del Modelo</i>	41

4.3.1	Modelo matemático para una planta de destino	41
4.3.2	Modelo matemático para dos plantas de destino	45
<b>5</b>	<b>Aplicación del Modelo sobre la Provincia de Sevilla</b>	<b>49</b>
5.1	<i>Implementación en CPLEX/Python</i>	49
5.2	<i>Estructuras de Datos</i>	51
5.3	<i>Resultados</i>	53
5.3.1	UGR nº1 – Mancomunidad de los Alcores	53
5.3.2	UGR nº2 – Mancomunidad del Guadalquivir	55
5.3.3	UGR nº3 – Mancomunidad de Servicios de la Vega	58
5.3.4	UGR nº4 – Mancomunidad de Municipios Sierra Norte	60
5.3.5	UGR nº5 – Consorcio Estepa-Sierra Sur	62
5.3.6	UGR nº6 – Mancomunidad de Municipios Comarca de Écija	63
5.3.7	UGR nº7 – Mancomunidad Intermunicipal Campiña 2000	65
<b>6</b>	<b>Conclusiones</b>	<b>67</b>
6.1	<i>Valoración de resultados</i>	67
6.2	<i>Trabajos futuros</i>	68
<b>Referencias</b>		<b>69</b>
<b>Anexo I. Código</b>		<b>75</b>
<b>Anexo II. Estructuras de Datos</b>		<b>83</b>

# Índice de Tablas

---

Tabla 3.1. Manc. de los Alcores (UGR nº1) - Habitantes y residuos	21
Tabla 3.2. Manc. del Guadalquivir (UGR nº2) - Habitantes y residuos	22
Tabla 3.3. Manc. de Servicios de la Vega (UGR nº3) - Habitantes y residuos	23
Tabla 3.4. Manc. de Municipios de la Sierra Norte (UGR nº4) - Habitantes y residuos	24
Tabla 3.5. Cons. de Medio Ambiente Estepa-Sierra Sur (UGR nº5) - Habitantes y residuos	24
Tabla 3.6. Manc. de Municipios Comarca de Écija (UGR nº6) - Habitantes y residuos	25
Tabla 3.7. Manc. Intermunicipal Campiña 2000 (UGR nº7) - Habitantes y residuos	25
Tabla 4.1. Caracterización del modelo	35
Tabla 5.1. Ejemplo de código en Python	49
Tabla 5.2. Parámetros	50
Tabla 5.3. Variables de decisión	51
Tabla 5.4. Variables auxiliares	51
Tabla 5.5. Formato para Estructura de Datos	52
Tabla 5.6. Parámetros comunes para todas las UGRs.	52
Tabla 5.7. Formato para parámetros particulares	53
Tabla 5.8. UGR nº1 – Resultados	53
Tabla 5.9. UGR nº1 – Resultados (Continuación)	54
Tabla 5.10. UGR nº2 – Resultados	56
Tabla 5.11. UGR nº2 – Resultados (Continuación)	57
Tabla 5.12. UGR nº3 – Resultados	59
Tabla 5.13. UGR nº4 – Resultados	61
Tabla 5.14. UGR nº5 – Resultados	62
Tabla 5.15. UGR nº6 – Resultados	64
Tabla 5.16. UGR nº7 – Resultados	65
Tabla 0.1. UGR nº1 – Estructura de Datos	83
Tabla 0.2. UGR nº1 – Parámetros particulares	83
Tabla 0.3. UGR nº2 – Estructura de Datos	84
Tabla 0.4. UGR nº2 – Estructura de Datos (Continuación)	85
Tabla 0.5. UGR nº2 – Parámetros particulares	85
Tabla 0.6. UGR nº3 – Estructura de Datos	86
Tabla 0.7. UGR nº3 – Parámetros particulares	87

Tabla 0.8. UGR nº4 – Estructura de Datos	87
Tabla 0.9. UGR nº4 – Parámetros particulares	88
Tabla 0.10. UGR nº5 – Estructura de Datos	88
Tabla 0.11. UGR nº5 – Estructura de Datos (Continuación)	89
Tabla 0.12. UGR nº5 – Parámetros particulares	89
Tabla 0.13. UGR nº6 – Estructura de Datos	89
Tabla 0.14. UGR nº6 – Parámetros particulares	89
Tabla 0.15. UGR nº7 – Estructura de Datos	90
Tabla 0.16. UGR nº7 – Parámetros particulares	90

# Índice de Figuras

---

Figura 2.1. Clases de complejidad. (Caballero, 2017)	4
Figura 2.2. Clasificación de metodologías de resolución TSP y VRP. (Anbuudayasankar et al., 2014)	9
Figura 3.1. Distancia máxima hasta estación intermedia. (ECOEMBES, 2018)	19
Figura 3.2. División comarcal agraria de la Provincia de Sevilla. (Diputación de Sevilla, 2019)	20
Figura 3.3. Unidades de Gestión de Residuos (UGRs) propuestas. (Diputación de Sevilla, 2019)	25
Figura 3.4. Reparto de residuos por UGRs en la Provincia de Sevilla. (Diputación de Sevilla, 2019)	26
Figura 3.5. Mapa de densidad demográfica de la Provincia de Sevilla. (Diputación de Sevilla, 2019)	27
Figura 3.6. Distribución de municipios por UGRs. (Elaboración propia)	30
Figura 3.7. Distribución de instalaciones por UGRs. (Elaboración propia)	31
Figura 4.1. UGR ficticia. Simplificación del caso práctico. (Elaboración propia)	34
Figura 4.2. Modelo UGR con una planta de destino. (Elaboración propia)	42
Figura 4.3. Modelo UGR con dos plantas de destino. (Elaboración propia)	45
Figura 5.1. UGR nº1 – Gráfica con rutas	54
Figura 5.2. UGR nº1 – Mapa con rutas	55
Figura 5.3. UGR nº2 – Gráfica con rutas	58
Figura 5.4. UGR nº2 – Mapa con rutas	58
Figura 5.5. UGR nº3 – Gráfica con rutas	60
Figura 5.6. UGR nº3 – Mapa con rutas	60
Figura 5.7. UGR nº4 – Gráfica con rutas	61
Figura 5.8. UGR nº4 – Mapa con rutas	61
Figura 5.9. UGR nº5 – Gráfica con rutas	63
Figura 5.10. UGR nº5 – Mapa con rutas	63
Figura 5.11. UGR nº6 – Gráfica con rutas	64
Figura 5.12. UGR nº6 – Mapa con rutas	64
Figura 5.13. UGR nº7 – Gráfica con rutas	66
Figura 5.14. UGR nº7 – Mapa con rutas	66



# **Lista de Abreviaturas**

---

CVRP	Problema de Enrutamiento de Vehículos con Capacidades ( <i>Capacited Vehicle Routing Problem</i> )
INE	Instituto Nacional de Estadística
MILP	Programación Lineal Entera Mixta ( <i>Mixed-Integer Linear Programming</i> )
mTSP	Problema de Agente Viajero Múltiple ( <i>Multiple Travelling Salesman Problem</i> )
PPRNP	Plan Provincial de Residuos No Peligrosos de la Provincia de Sevilla (Diputación de Sevilla, 2019)
RSU	Residuos Sólidos Urbanos
TFM	Trabajo Fin de Máster
TSP	Problema del Agente Viajero ( <i>Travelling Salesman Problem</i> )
UGR	Unidad de Gestión de Residuos
VRP	Problema de Enrutamiento de Vehículos ( <i>Vehicle Routing Problem</i> )
<i>h</i>	horas
<i>km</i>	Kilómetros
<i>kg</i>	Kilogramos



# 1 INTRODUCCIÓN

---

Empieza en este capítulo la memoria del Trabajo Fin de Máster (de ahora en adelante, TFM). El título de este trabajo es “*Optimización de Rutas de Vehículos de Recogida de Basuras mediante MILP*”, donde este último acrónimo se refiere a la metodología de optimización utilizada, la Programación Lineal Entera Mixta, por las iniciales de sus siglas en inglés *Mixed-Integer Linear Program*. En esta primera sección, se proporciona una breve introducción a esta metodología, así como a la temática de la recogida de basuras. Se presenta también la Provincia de Sevilla como el sujeto sobre el que se aplica el modelo. Del mismo modo, se ofrece una visión general de como se ha estructurado el texto.

## 1.1 Motivación

El exponencial desarrollo económico y poblacional a nivel mundial en los últimos 80 años ha provocado un gran aumento en la producción de Residuos Sólidos Urbanos (RSU). Se estima que la generación de residuos se incremente todavía más en los años venideros. De manera que la gestión de estos residuos se postula como uno de los grandes problemas de la humanidad en el siglo XXI.

Aunque los esfuerzos van encaminados a una reducción masiva de la generación de residuos, es necesario gestionar los que de momento se siguen produciendo. Las medidas de prevención se enfocan en el origen de los residuos, intentando concienciar a la población en materia tanto de reutilización como de reciclaje; mientras que, dentro del ámbito de la gestión, además del tratamiento y recuperación de residuos, se encuentra la recogida de estos, cada vez más solicitada y con más restricciones de tipo ambiental, costes, etc. Es en esto en lo que se centra este TFM, en la resolución de un problema de recogida de RSU en un entorno concreto, la Provincia de Sevilla.

Recientemente se ha publicado el “*Plan Provincial de Residuos no Peligrosos de la Provincia de Sevilla (2020-2035)*” (Diputación de Sevilla, 2019). En este documento se define la estrategia a seguir en materia de residuos en la provincia. Propone la creación de nuevas infraestructuras, así como la adaptación de algunas de la ya construidas. También aborda la necesidad de un modelo de gestión centralizado que coordine las actuaciones de las distintas entidades responsables de los residuos que existen en la provincia.

## 1.2 Objetivos del Proyecto

La Provincia de Sevilla tiene un total de 106 municipios y organiza parte de la gestión de residuos en entidades supramunicipales conocidas como mancomunidades y consorcios. Estas entidades agrupan, entre otros, núcleos de población dispersos y con pocos habitantes. La aplicación del modelo que se discute en este texto constituye una propuesta para unificar la gestión de residuos en estos municipios, con el objetivo principal de aumentar la eficiencia del servicio mediante la reducción de costes.

El modelo matemático discutido tiene como función objetivo la reducción de los costes del servicio. En este caso se han asociado los costes directa y únicamente a las distancias establecidas entre los distintos municipios. Por tanto, la reducción de costes se plantea como una optimización de la distancia recorrida por las

flotas de vehículos utilizadas.

Actualmente, existe una referencia de este tipo de sistemas en la provincia, la Mancomunidad del Guadalquivir tiene implantado la recogida conjunta de RSU y se utilizará de referencia para hacerlo extensivo al resto. El problema se plantea de forma individual para cada una de las seis mancomunidades y el consorcio que conforman la Provincia de Sevilla.

### **1.3 Metodología utilizada**

El modelo de gestión conjunta para la recogida de RSU se va a plantear como un problema de enrutamiento de vehículos. Este problema es ampliamente conocido debido a su relevancia en el campo de la Optimización Combinatoria y a la dificultad que conlleva resolverlo. La resolución de este tipo de problemas ha sido discutida por multitud de autores, algunos de sus trabajos se revisarán más adelante.

Las metodologías de resolución pueden dividirse principalmente en tres grupos: metaheurísticos, heurísticos y exactos. Se repasarán las características más comunes de todos ellos. Las metodologías exactas son las menos estudiadas debido a que su capacidad de tratar problemas de gran tamaño es limitada. No obstante, el trabajo utiliza métodos exactos para resolver el problema puesto que el número máximo de nodos que tiene que tratar es de 33.

Para garantizar la obtención de una solución e intentar acotar al máximo el espacio de búsqueda de las soluciones, se han adaptado y particularizado los modelos encontrados en la literatura mediante una caracterización exhaustiva de la situación en la Provincia de Sevilla.

Para la resolución del caso práctico planteado, se utilizará el software de resolución de problemas de Programación Lineal Entera Mixta CPLEX (IBM ILOG, 2019). Aunque los cálculos se realizarán haciendo uso de las funcionalidades de CPLEX, el código se escribirá en Python (Python Software Foundation, 2018) y la ejecución se llevará a cabo en la aplicación Jupyter Notebook (Anaconda Software Distribution, 2018). Estas decisiones se justificarán a su debido tiempo.

Al terminar este trabajo, se presentará una propuesta con las rutas que tendrían que recorrer los vehículos para la aplicación del modelo de recogida conjunta. Dentro de esta propuesta se ofrecen más datos que caracterizan el modelo como la cantidad de basura recogida o la distancia recorrida entre otros. Para estos cálculos, se utilizará como base la información ofrecida por las entidades competentes.

### **1.4 Estructura del texto**

El texto se ha dividido en seis capítulos, empezando con el presente capítulo de introducción que concluye con este apartado. A continuación, se revisa el Estado del Arte de los problemas de enrutamiento de vehículos en el Capítulo 2, donde se introducen las formulaciones matemáticas más conocidas de estos problemas, como son el Travelling Salesman Problem (TSP) y el Vehicle Routing Problem (VRP). En esta sección se discuten también las principales metodologías de resolución para este tipo de problemas. En el Capítulo 3, se define el escenario sobre el que se aplicará el modelo, la Provincia de Sevilla y en particular su división supramunicipal en mancomunidades, analizando la situación y sentando las bases sobre las que se construirá el modelo presentado en el Capítulo 4. En dicho capítulo se traduce la información procedente del Capítulo 3 en parámetros y restricciones, presentando dos modelos definitivos, aplicables según la tipología de la mancomunidad. El Capítulo 5 comprende desde una breve explicación acerca del software utilizado hasta la presentación de los resultados obtenidos tras aplicar el modelo. Finalmente, en el Capítulo 6 se redacta una conclusión general considerando los puntos tratados en los capítulos anteriores.

## 2 ESTADO DEL ARTE

---

**E**n esta sección se revisa la literatura relacionada con la resolución del Problema de Enrutamiento de Vehículos, VRP por sus siglas en inglés (*Vehicle Routing Problem*). Se describe este problema empezando por una breve reseña histórica, así como las variantes que han surgido a lo largo de los años. De todas las formulaciones matemáticas que se han planteado, aquí se presentan únicamente aquellas que se utilizarán en futuros apartados para desarrollar el modelo matemático utilizado en este ensayo.

En el presente capítulo también se presentan los métodos que se han utilizado a lo largo de los años para resolver el VRP. Se comparan las distintas técnicas de resolución que existen para este tipo de problemas: heurísticas, metaheurísticas y exactas, profundizando en estas últimas. La Programación Lineal Entera Mixta (MILP) pertenece a esta familia y es la metodología elegida para resolver el caso presentado en la Sección 3.

### 2.1 Introducción histórica

En la práctica, en la ingeniería todo se basa en una continua toma de decisiones y es algo que tienen en común sus distintas especialidades. Particularmente, la rama de la logística se ve muy influenciada por la “correcta” elección entre las opciones disponibles. La logística comenzó siendo una ciencia militar que se ocupaba de la gestión de los recursos disponibles, decidiendo cómo se transportaban y hacia dónde. Con el paso del tiempo, este campo se ha hecho extensivo al ámbito industrial y comercial donde las tareas de transporte suponen un gran porcentaje del precio final del producto.

Sin embargo, la logística no solo se entiende en una sola dirección, donde el consumidor es el final de la cadena; sino que para cerrar el círculo existe un flujo de retorno que parte desde el consumidor, como es el caso de la recogida de residuos. Esto es lo que se conoce como Logística Inversa. El problema de la recogida de residuos se ha venido estudiando especialmente en los últimos 20 años, cuando se ha empezado a tomar conciencia de los problemas medioambientales que se derivan de una ineficiente gestión de los residuos. La aplicación de la Logística Inversa no se limita exclusivamente a la recogida de residuos; el desarrollo del comercio online se aplica cada vez más para satisfacer las devoluciones solicitadas por los clientes de los productos comprados por este canal. (Serrano, 2015)

Estos problemas de Logística Inversa pueden plantearse desde la definición clásica de la logística, el cliente tiene una demanda que en este caso es la recogida de los residuos y paga una tarifa por ello, por lo que recibe un servicio que satisface dicha demanda. Por esta razón, da igual el sentido del flujo de bienes o servicios, el denominador común es la resolución de un Problema de Enrutamiento de Vehículos.

La logística se plantea habitualmente desde un enfoque matemático, modelándose como un problema de asignación de recursos. Este es el caso de los Problemas de Enrutamiento de Vehículos, que se abordan desde distintas metodologías, ofreciendo soluciones aproximadas mediante algoritmos basados en heurísticas y metaheurísticas o soluciones óptimas exactas, haciendo uso de métodos exactos de resolución.

El grado de dificultad de este tipo problemas es NP-Hard según la clasificación planteada por la teoría de la complejidad computacional, donde los problemas se categorizan según el consumo de recursos computacionales. Según esta clasificación la clase de complejidad P engloba a todos aquellos problemas que

se pueden resolver por una máquina determinista en un tiempo dado por una función polinómica, lo que se conoce como tiempo polinómico. Por otro lado, tenemos el grado NP para los problemas resueltos en tiempo polinómico, pero en una máquina no determinista. El grado NP-Hard hace referencia a aquellos problemas que como mínimo son tan difíciles de resolver como un NP, pero no tienen que ser NP necesariamente. La resolución de problemas de clase NP se realiza mediante la comprobación de las soluciones factibles del problema en tiempo polinomial. La Figura 2.1 ofrece una representación esquemática de las clases de complejidad. Los problemas NP-Complete se consideran los más complejos y de demostrar que uno de ellos puede resolverse en tiempo polinomial, todos los NP podrían. Hoy en día no se ha encontrado ningún problema en el que se valide esta conjeta.

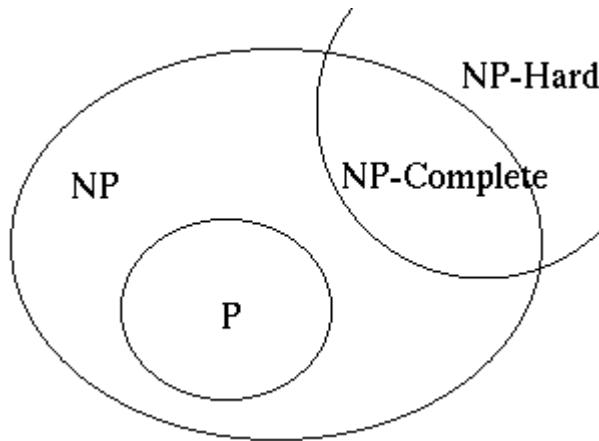


Figura 2.1. Clases de complejidad. (Caballero, 2017)

Los Problemas de Enrutamiento de Vehículos aparecieron por primera vez en 1959 formulados en un artículo firmado por George Dantzig y John Ramser, donde definen el *Truck Dispatching Problem*. El problema que planteaban era optimizar las rutas de una flota de camiones de gasolina entre una central y un número considerable de estaciones de servicio; mientras que el objetivo que establecían era minimizar la distancia total recorrida por toda la flota, problema que como se verá en las siguientes secciones, se parece mucho al que pretende dar solución este trabajo. (Dantzig & Ramser, 1959)

Las primeras aproximaciones realizadas, como por ejemplo la comentada anteriormente de Dantzig y Ramser, estudiaban estos problemas basándose en formulaciones de programación lineal, realizando sus cálculos a mano y apoyándose en los ordenadores que empezaban a aparecer, en lo que por aquel entonces denominaban “*Automatic digital computing machine*”. Multitud de problemas se han asemejado al planteamiento realizado por el *Truck Dispatching Problem*, como es el caso de la recogida de residuos. Resolver estos supuestos con esos medios resultaba tedioso y era un proceso que tomaba mucho tiempo. La evolución de las tecnologías, en concreto el desarrollo de la capacidad de cálculo de los equipos informáticos, ha permitido disminuir drásticamente los tiempos de computación, a la vez que el tamaño de los problemas aumentaba.

Del mismo modo, la aparición de nuevas técnicas también ha permitido avanzar en la resolución de casuísticas cada vez más complejas. La investigación se ha ido centrando en las técnicas heurísticas y metaheurísticas por su capacidad de abordar problemas de una envergadura que los métodos exactos no son capaces de resolver, es por ello por lo que hay más literatura disponible de las dos primeras que de esta última. Sin embargo, la resolución del caso aquí propuesto se realiza aplicando metodologías exactas. Esta decisión se justificará en este capítulo y se entenderá a medida que se caracterice el problema.

## 2.2 Descripción del Problema

Como se introduce en el punto anterior, la recogida de basuras puede modelarse como un Problema de Enrutamiento de Vehículos o VRP, como será referido a partir de este punto. Este problema consiste en el diseño y optimización de rutas de entrega o recogida desde un depósito hacia unos clientes ubicados en distintos puntos, cada uno con su demanda correspondiente. Estos modelos están sujetos a una serie de restricciones: uno o varios vehículos, la capacidad de estos, longitud máxima de las rutas, jornadas laborales, etc. Cada caso concreto se particulariza añadiendo restricciones que acerquen la formulación “básica” del

problema a la realidad. Estas diferencias en las restricciones han dado lugar al desarrollo de distintas tipologías de VRP: problemas con ventanas de tiempo, con preferencia de clientes, etc. Las distintas variantes se discutirán brevemente en el Apartado 2.2.2 de esta misma sección. (Laporte, 2006)

Existen una serie de elementos que son comunes a todos los problemas derivados del VRP y que por tanto tendrán que ser identificados en el caso práctico que se presenta en este proyecto. A continuación, siguiendo el planteamiento seguido por Gómez Cámara en su tesis (Gómez, 2010), se enumeran estos conceptos generales.

- **CLIENTES**

Los clientes son los puntos de destino de los vehículos, pueden ser de entrega o de recogida. Particularizando para el planteamiento de la recogida de residuos, se identificarían exclusivamente con puntos de recogida. Sin embargo, como se ha comentado anteriormente, se puede generalizar a que tienen asociada una demanda que debe ser satisfecha mediante la asignación de rutas que los visiten. Esta demanda puede ser para recibir o para entregar, pero no deja de ser una demanda.

- **DEPÓSITOS**

Se entiende por depósitos las localizaciones donde los vehículos se encuentran estacionados y también las instalaciones desde donde se distribuyen los productos demandados, o bien, donde se almacena lo que sea que se haya recogido a los clientes.

- **VEHÍCULOS**

Los vehículos son los encargados de transportar las mercancías entre los distintos puntos del problema. En problemas mono-objetivo, la optimización del VRP suele enfocarse a minimizar la distancia recorrida por los vehículos, puesto que el coste total suele depender mucho de este factor.

Cada vehículo tiene unas características que definen su capacidad de volumen a transportar. Además, suelen estar sujetos a otras limitaciones, como por ejemplo el tiempo que puede estar un conductor de servicio. Las flotas pueden estar compuestas por uno o por varios tipos de vehículos. Para el caso objeto de estudio, se expondrán las hipótesis relacionadas con la flota utilizada en el Capítulo 4.

- **RUTAS**

Las rutas son los caminos que recorren los vehículos y que conectan los nodos del problema. Este término es de gran importancia en la formulación de los problemas de enrutamiento de vehículos. Utilizando la notación utilizada por Gómez Cámara (Gómez, 2010), el planteamiento de estas cuestiones suele hacerse en forma de grafos:  $G = (V, E)$ , donde  $V$  es el conjunto de nodos que incluye tanto los depósitos como a los clientes, mientras que  $E$  son los arcos que se definen entre el punto  $i$  y el punto  $j$ , que pueden ser tanto depósitos como clientes. El conjunto de los arcos recorridos por los vehículos define las rutas elegidas como solución del problema.

## 2.2.1 Travelling Salesman Problem (TSP)

El Problema del Agente Viajero, en inglés *Travelling Salesman Problem* (TSP) es el caso más simple y general de enrutamiento de vehículos. Solo existe un vendedor que tiene que visitar a todos los clientes en una única ruta, minimizando los costes asociados con la distancia recorrida. Este problema se menciona en la primera aparición de los problemas de rutas de vehículos (Dantzig & Ramser, 1959), donde se establecen que los VRP son una generalización del TSP. A pesar de ser el caso más simple de VRP, la dificultad para resolver este problema también es muy alta, siendo su complejidad NP-Hard, compartiendo clasificación con el problema generalista.

Según la recopilación realizada en (Davendra, 2010), las primeras investigaciones sobre el TSP datan del siglo XIX realizadas por los matemáticos William R. Hamilton y Thomas P. Kirkman; mientras que las formulaciones matemáticas se empezaron a desarrollar en torno al 1930 en las universidades de Viena y Harvard. Los modelos encontrados en la literatura utilizan distintas notaciones para definir las variables y los elementos del problema, así como difieren en la forma de ordenar y presentar sus restricciones. A continuación, se establece la notación matemática que se va a utilizar en este apartado.

$c_{ij}$ : coste asociado al trayecto desde el nodo  $i$  al nodo  $j$ .

$x_{ij} = 1$  cuando se recorre el arco formado entre el nodo  $i$  y el nodo  $j$

$x_{ij} = 0$  cuando no se recorre el arco formado entre el nodo  $i$  y el nodo  $j$ .

$n$ : número de clientes (dimensión del problema)

La formulación en términos de programación entera que aquí se presenta esta sacada de (Gómez, 2010). Sin embargo, se ha adaptado la notación matemática a la comentada anteriormente para hacerla coincidir con la que se va a seguir a lo largo de este proyecto.

$$\min \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \quad (2.1)$$

$$s. a. \sum_{j=1}^n x_{ij} = 1 \quad \forall i \in \{1, \dots, n\}, i \neq j \quad (2.2)$$

$$\sum_{i=1}^n x_{ij} = 1 \quad \forall j \in \{1, \dots, n\}, j \neq i \quad (2.3)$$

$$u_i - u_j + n x_{ij} \leq n - 1 \quad \forall i, j \in \{2, \dots, n\} \quad (2.4)$$

$$1 \leq u_i \leq n - 1 \quad \forall i \in \{2, \dots, n\}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in E$$

En esta formulación, la función objetivo (2.1) busca el mínimo del producto entre el coste y la variable binaria, que establece si se recorre esa ruta o no. En esta expresión, el coste puede considerarse como el tiempo, la distancia recorrida, etc. Por otro lado, las restricciones (2.2) y (2.3) hacen que en cada nodo exista únicamente un solo arco de llegada y otro de salida, siempre y cuando este nodo pertenezca al óptimo. Finalmente, la restricción (2.4) se encarga de evitar que se formen subciclos entre nodos que no permitan encontrar una solución. La formulación de las restricciones para evitar subciclos tiene distintos planteamientos, el que se presenta aquí es el MTZ, de las iniciales de sus autores Miller, Tucker y Zemlin, las cuales definen el orden en que los nodos son visitados. (Benhida & Mir, 2018)

Dentro del TSP, también existen distintos planteamientos del problema. En el texto de Davendra, se ofrece la siguiente clasificación, siendo la última modalidad de la lista la que más se asemeja al VRP (Davendra, 2010). La formulación que se expone en el siguiente apartado 2.2.2 es igualmente válida para resolver ambos problemas, VRP y mTSP.

- Problema del agente viajero simétrico (sTSP): La particularidad de este problema es que la matriz de costes es simétrica, lo que implica que la distancia entre las ciudades es la misma independientemente del sentido.
- Problema del agente viajero asimétrico (aTSP): Es el caso contrario al anterior, la matriz de costes no es simétrica. No es lo mismo ir desde el nodo  $i$  al nodo  $j$ , que recorrer el mismo camino de  $j$  a  $i$ , de hecho, esa ruta podría incluso no ser posible.
- Problema del agente viajero múltiple (mTSP): Para un número dado de nodos  $n$ , se dispone de  $m$  agentes en un mismo depósito. Este problema consiste en encontrar para todos los agentes las rutas para que cada nodo restante sea visitado una única vez por un único viajante y el coste total sea mínimo.

El TSP es uno de los problemas más tradicionales en el campo de la Investigación Operativa y según se deriva de los textos (Davendra, 2010) y (Hoffman et al., 2016) la aplicabilidad de su formulación se extiende a tareas de programación de horarios en imprentas, asignación de turnos, programación de entrevistas, producción/fabricación en la industria, así como programación de misiones militares, secuencia de actividades de máquinas, análisis de la estructura de cristales, etc.

## 2.2.2 Vehicle Routing Problem (VRP)

Al contrario de lo que ocurre con otros problemas de Optimización Combinatoria, para el VRP no existe una definición aceptada universalmente. Esto se debe principalmente a que la variedad de restricciones que se encuentran en la práctica genera un alto número de variantes. No obstante, los esfuerzos se han concentrado en torno a una visión clásica de problema como es el problema de rutas de vehículos con capacidades, en inglés *Capacited Vehicle Routing Problem* (CVRP). Esta variante clásica es una de las más estudiadas. Además, su formulación sirve de punto de partida para adaptar situaciones reales más complejas, como ocurre con el caso práctico que se irá desarrollando a lo largo de este texto. (Laporte, 2006)

La definición clásica de un CVRP para un conjunto de  $n$  clientes se presenta como un grafo  $G = (V, A)$ , donde  $V = \{1, \dots, n\}$  es el conjunto de vértices del problema y  $A = \{(i, j) : i, j \in V, i \neq j\}$  es el conjunto de arcos. El vértice 1 se reserva para el depósito donde se encuentran los  $m$  vehículos cuya capacidad es  $Q$ . Por otro lado, cada cliente tiene asociada una demanda  $q_i \leq Q$ ; mientras que los arcos tienen un coste asociado  $c_{ij}$  que puede referirse a distancia recorrida, tiempo empleado o coste económico. (Laporte, 2006)

Existen distintas formas de plantear la formulación del CVRP, según como se aborde la resolución del problema. La que se presenta a continuación se corresponde con las Formulaciones de Flujo de Vehículos (*Vehicle Flow Formulations*) propuesta en (Toth & Vigo, 2000). Sin embargo, las restricciones relacionadas con la eliminación de subciclos se han adaptado siguiendo las investigaciones que se presentan en (Desrochers & Laporte, 1991).

$$\min \sum_{i=1}^n \sum_{j=1}^n c_{ij} \sum_{k=1}^m x_{ijk} \quad (2.5)$$

$$s.a. \quad \sum_{j=2}^n x_{1j} \leq m \quad (2.6)$$

$$\sum_{i=2}^n x_{i1} = \sum_{j=2}^n x_{1j} \quad (2.7)$$

$$u_i - u_j + Q x_{ij} \leq Q - q_i \quad \forall i, j \in \{2, \dots, n\}, i \neq j \quad (2.8)$$

$$q_i \leq u_i \leq Q \quad \forall i \in \{2, \dots, n\}$$

$$x_{ij} \in \{0,1\} \quad \forall (i, j) \in A$$

Al igual que ocurría en el TSP, la función objetivo (2.5) representa el coste total de la solución y este se obtiene multiplicando el coste de cada arco por la variable binaria asociada a dicho arco. En esta formulación, se introduce un nuevo sumatorio que se encarga de tener en cuenta el número de vehículos utilizados, de ahí que reciba el nombre de “Formulación de tres índices” en referencia a los índices  $i$ ,  $j$  y  $k$  que aparecen en el modelo. También relacionadas con los vehículos las restricciones (2.6) y (2.7). En concreto, la ecuación (2.6) planteada indica que  $m$  es la cantidad máxima de vehículos a utilizar, mientras que la (2.7) asegura que todos los camiones que salen regresan al depósito. Finalmente, en (2.8) se establecen las limitaciones de capacidad de los camiones, además de garantizar que no se forman subciclos.

Esta formulación básica o clásica es utilizada a modo de cimientos para construir modelos particulares que se adapten a casos reales con otro tipo de especificaciones. El objetivo o el resultado del problema también se adapta según lo que se quiera obtener: minimizar distancia, número de vehículos, costes asociados al uso de vehículos, etc. Esta familia de problemas que se desarrolla a partir del modelo del CVRP es muy extensa y engloba entre otros al caso práctico estudiado en este trabajo y cuyo modelo particular se discute en la Sección 4. Bajo estas líneas, se ofrece una recopilación de otras ramificaciones del VRP basándonos en el análisis detallado que hace de ellas Gómez Cámara en su tesis (Gómez, 2010).

- ***Multi Depot Vehicle Routing Problem (MDVRP)***

Este problema difiere del VRP clásico en que tiene más de un depósito, lo que significa que los vehículos están estacionados en varios puntos. Dentro de este grupo, según cómo se organicen las flotas de vehículos encontramos dos variantes: flotas fijas asociadas a cada estación, los vehículos están distribuidos entre los distintos depósitos y siempre salen y regresan al mismo; flotas fijas que parten de una estación y regresan a otras, se elimina la restricción del grupo anterior de tener que volver al depósito de partida.

- ***Periodic Vehicle Routing Problem (PVRP)***

Esta variante establece una periodicidad superior a un día. Por ejemplo, para un periodo fijado de  $T$  días, esto significa que los vehículos pueden visitar a los clientes cualquier día dentro de ese periodo. Se suele asignar a cada cliente una demanda y un calendario.

- ***Split Delivery Vehicle Routing Problem (SDVRP)***

Un SDVRP plantea la división de la cantidad demandada de un determinado bien por un cliente entre distintos camiones, con el objetivo de reducir los costes. Esta aproximación se utiliza especialmente en situaciones donde la demanda del cliente es mayor que la capacidad de los vehículos, lo que obliga a distribuir la carga a entregar entre varios camiones.

- ***Stochastic Vehicle Routing Problem (SVRP)***

La particularidad de esta modalidad reside en el carácter aleatorio de las variables y componentes de problema. La aleatoriedad puede estar presente en el número de clientes, la demanda o los tiempos del problema. Por lo general, estos problemas se abordan en dos fases bien diferenciadas. Durante la primera se resuelve el modelo con unos valores estimados de las componentes aleatorias, sin tener en cuenta su carácter estocástico; mientras que la segunda fase, se validan y corrigen las suposiciones realizadas durante la primera, a medida que se conocen más aspectos de estas variables aleatorias.

- ***Vehicle Routing Problem with Pick-up and Delivery (VRPPD)***

En este caso, los clientes actúan a la vez como receptores y emisores de carga. Una de las grandes problemáticas que plantean este tipo de problemas es el hecho de la reserva de espacio en los camiones para poder recoger mercancía. Se suelen plantear simplificaciones e hipótesis para reducir la complejidad del problemas: como que un mismo cliente no pueda recibir y recoger al mismo tiempo, solo puede hacer una de estas acciones; que no haya intercambio de mercancías entre los propios clientes, lo que se recoge se queda en el camión hasta el regreso a la base; o bien, se establece que no se puede empezar a recoger hasta que no se ha entregado toda la mercancía.

- ***Vehicle Routing Problem with Time Windows (VRPTW)***

El VRP con ventanas de tiempo asigna a cada cliente un espacio temporal en el que se pueden realizar las tareas de entrega o recogida de las mercancías. Esto añade un grado de complejidad extra cuando en vez de imponer la prohibición de acceder al cliente fuera de la ventana de tiempo definida, se establecen penalizaciones por acudir fuera del horario, haciendo que este factor entre en juego a la hora de la optimización de costes. Por ejemplo, si hay dos clientes que están muy separados del resto, seguramente sea conveniente pagar la penalización en uno de ellos para no tener que volver a incurrir en los costes asociados al desplazamiento hasta su posición.

## 2.3 Métodos de Resolución

Una vez presentados los problemas que se toman como referencia para el desarrollo del trabajo y sus formulaciones matemáticas, es el momento de plantear los métodos que se utilizan para resolverlos. En la literatura se pueden encontrar tres grupos importantes y bien diferenciados: heurísticos, metaheurísticos y exactos. Todos ellos se revisan en este punto, profundizando en el campo de los métodos exactos ya que son los utilizados en este texto. La clasificación realizada en (Anbuudayasankar et al., 2014) y que se presenta en la Figura 2.2 recoge algunos de los procedimientos más destacados de cada grupo y además incluye otros métodos como los interactivos e híbridos que no serán revisados aquí.

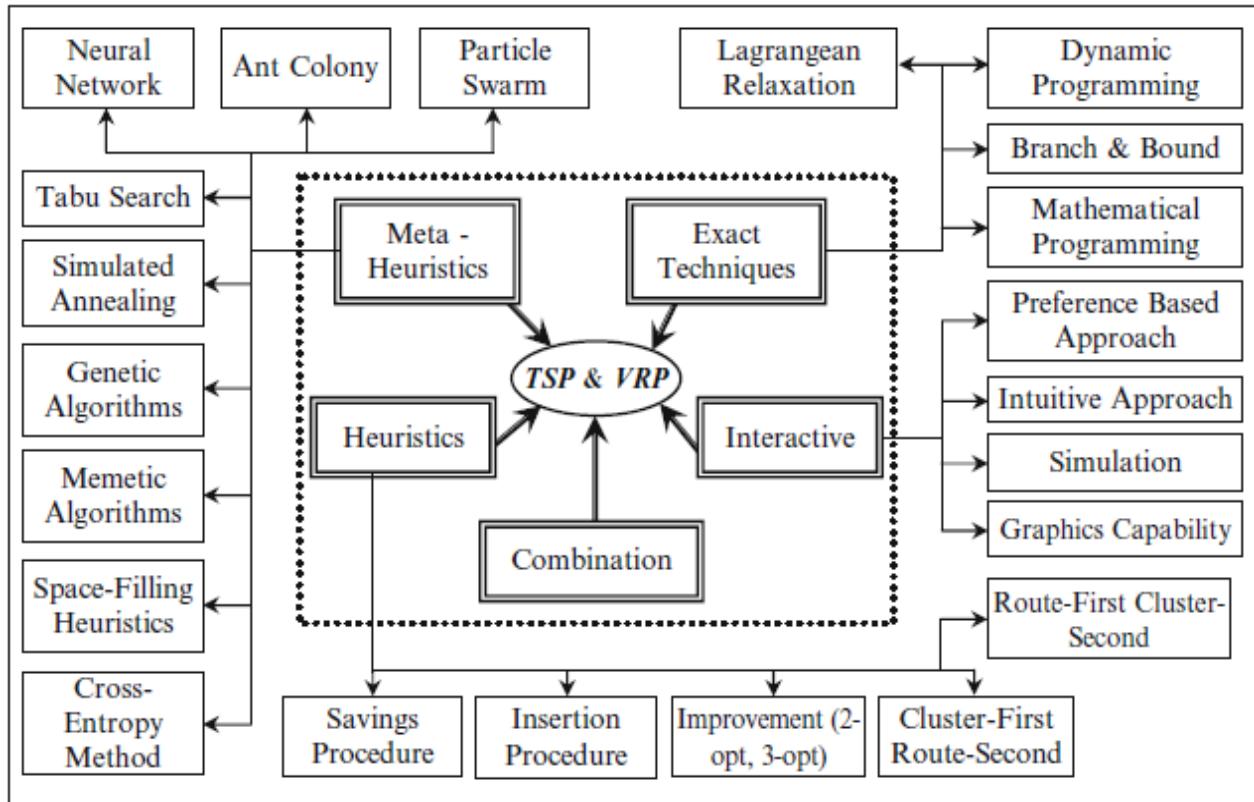


Figura 2.2. Clasificación de metodologías de resolución TSP y VRP. (Anbuudayasankar et al., 2014)

Como ya se ha comentado, la dificultad de este problema es NP-Hard, lo que significa que los esfuerzos para resolverlos crecen de forma exponencial cuando aumenta el número de puntos. Según la comparativa ofrecida en (Laporte, 2006), un TSP con cientos e incluso miles de nodos, puede resolverse mediante los algoritmos de *Branch-and-Cut*; mientras que los algoritmos de resolución exactos más sofisticados para VRP son capaces de resolver únicamente problemas de hasta 150 clientes. Por este motivo, los intentos de resolver el VRP se han enfocado en el desarrollo de métodos heurísticos y metaheurísticos que son mucho más flexibles que los exactos.

### 2.3.1 Heurísticos

Las metodologías heurísticas aplicadas a resolución de problemas pueden definirse como una aproximación intuitiva con la cual la estructura del problema se puede interpretar y analizar para obtener una solución razonable. Esto quiere decir que la solución que se obtiene efectivamente resuelve el problema, pero no tiene por qué ser la solución óptima global del problema. Algunas de las razones por las que se eligen soluciones mediante heurísticos son: es desconocido el planteamiento matemático del problema; incluso conociendo la formulación matemática, es computacionalmente imposible de encontrar una solución exacta; son fácilmente entendibles, etc.

A continuación, se ofrece una clasificación de métodos heurísticos, basándonos en los trabajos de (Laporte, 1992) y (Silver et al., 1980). En ocasiones es conveniente abordar el problema utilizando distintas aproximaciones de las que aquí se presentan para seleccionar aquella que mejor funciona. Los heurísticos que se comentan bajo estas líneas son generales para la resolución de todo tipo de problemas, siendo los métodos más aplicados en el VRP los métodos constructivos, por fases y de inserción, aunque estos últimos realmente son una variante de los constructivos.

- **Métodos de Descomposición**

En estos algoritmos, el problema principal se divide en problemas más pequeños que se resuelven por separado, siendo la solución de uno la entrada de otro. Dentro de este grupo, particularizado para el VRP tenemos los algoritmos por fases, que según el subproblema que se resuelva primero pueden ser de dos tipos:

- *Cluster first, route second:* Durante la primera fase se dividen los clientes en grupos (*clusters*) que serán recorridos dentro de una misma ruta, respetando las restricciones de capacidad de los vehículos. Para la segunda fase, se resuelve un TSP para cada *cluster*. Se han propuestos distintas formas de abordar la división en *clusters* como los algoritmos de barrido (Gillett & Miller, 1974) o bien resolviendo en la primera fase un problema de asignación generalizada, partiendo de un cliente aleatorio (Bramel & Simchi - Levi, 1995) y (Fisher & Ramchandran, 1981).
- *Route first, cluster second:* Al contrario que las propuestas del párrafo anterior, el TSP se calcula para todos los clientes en la primera fase sin tener en cuenta las restricciones de capacidad, que se consideraran en la segunda fase generando subrutas como propone (Beasley, 1983).

#### • Métodos de reducción

La idea aquí se basa en obtener las soluciones óptimas de varios casos numéricos que sean objeto de estudio. A partir de estas soluciones se buscan características comunes y se realiza la suposición de que son generalistas para todos los problemas. Con esto se consigue simplificar mucho la dificultad de estos problemas porque se está limitando el número de soluciones en base a unas pocas. Por ejemplo, una práctica común es tomar alguna de las variables como cero. Una vez obtenida la solución se verifica que realmente se cumplen las especificaciones. Sin embargo, este planteamiento presenta el inconveniente de desestimar soluciones que sean buenas. (White, 1969)

#### • Métodos de manipulación del modelo

El enfoque de este método es cambiar el planteamiento del modelo de manera que se facilite la obtención de una solución, para después tomarla como representativa del problema original. Lo que se hace es interpretar la solución de un modelo más sencillo para deducir la solución del problema global. En este caso, el número de soluciones puede aumentar o disminuir. Algunos ejemplos de modificación del modelo son: linealización de restricciones, relajación de algunas hipótesis que puedan ser muy restrictivas, cambiar las distribuciones probabilísticas por otras que sean más sencillas, etc. En (Geoffrion, 1976) se pueden encontrar simplificaciones aplicadas a las funciones dependientes de los costes.

#### • Métodos constructivos

La idea principal de un algoritmo constructivo es literalmente construir una solución. Se parte de una solución en blanco y mediante un proceso iterativo se va conformando una solución factible. Estos métodos suelen tener un enfoque determinista y en cada paso seleccionan la mejor opción en función de algún criterio. De esta manera, en estos algoritmos es necesario responder a tres cuestiones: por dónde se empieza, qué nodo es el siguiente en incluirse y en qué posición se incluye.

En esta categoría destacan los procedimientos de ahorros discutidos en (Clarke & Wright, 1964), en cuyo planteamiento se genera una nueva ruta a partir de dos rutas existentes, siempre y cuando la reorganización mejore la función objetivo y sea compatible con las restricciones del modelo. Desde que Clarke y Wright propusieron su algoritmo, se han estudiado numerosas mejoras en la elección de las rutas para combinar, como los ahorros en paralelo planteados por (Altinkemer & Gavish, 1991) o la propuesta de (Doyuran & Çatay, 2011) donde se supedita la creación de las nuevas rutas a que se combinan clientes con demandas grandes y pequeñas.

Otro enfoque constructivo son los conocidos como algoritmos de inserción, en los cuales se comienza con un TSP con un solo destino y a partir de este se van creando una serie de subrutas para los  $n$  clientes. Es decir, en cada iteración se inserta otro cliente en base a criterios como el más cercano o el más barato (Rosenkrantz et al., 1977). La principal ventaja de estas aproximaciones es que son fácilmente programables.

#### • Métodos de mejora local

En estos algoritmos se aplica un procedimiento de mejora partiendo de una solución factible inicial. Se establecen una serie de criterios para buscar en el entorno de esta solución de partida posibles soluciones que mejoren el resultado de la función objetivo. Si se encuentran en dicho entorno soluciones que mejoren, se evoluciona hacia esa nueva solución y se repite el proceso, por lo general hasta que no se pueda mejorar más.

El paso de una solución a otra se conoce como movimiento y el entorno donde se buscan candidatos,

vecindario. Existen diversos estudios sobre cómo generar los vecindarios, aquí se presentan algunos agrupados según sean intrarrutas o entrerrutas.

- Procedimientos intrarrutas: Las modificaciones se realizan dentro de una misma ruta. Uno de los más conocidos es el operador  $\lambda$ -Intercambio, en la que se elimina un número  $\lambda$  de elementos y se vuelven a crear arcos entre los  $\lambda$  segmentos que se generan, combinándolos todos y evaluando las nuevas rutas por si alguna mejora el resultado (Lin, 1965). Esta metodología fue mejorada en (Johnson & McGeoch, 2003), que busca la mejora entre los segmentos que se crean, sin tener que hacer una búsqueda completa. También se redujo esta búsqueda en (Renaud et al., 1996) con la versión 4-opt, donde se limita la exploración de los segmentos.
- Procedimientos entrerrutas: Se combinan varias rutas dando lugar a una nueva. En este campo se pueden destacar los operadores de Van Breedam, en los cuales se realizan intercambio de clientes entre rutas (Van Breedam, 1995).

También existen distintas formas de inserción según la estrategia elegida: se puede priorizar el encontrar por ejemplo la solución más cercana o barata que mejore la función objetivo, o bien evaluar cuál es la mejor de todas las posibles soluciones analizadas.

Uno de los principales problemas que presenta esta metodología es la aceptación de una solución local “mala”. Es decir, un óptimo local que se encuentre muy lejos del óptimo del problema y, aun así, se tome como la mejor, debido a que atendiendo a los criterios de selección no encuentra nada mejor.

### 2.3.2 Metaheurísticos

Las técnicas metaheurísticas son una de las herramientas más potentes para la resolución de problemas de Optimización Combinatoria. El término metaheurístico fue acuñado por Glover en el año 1986, quién posteriormente lo definiría en (Sørensen & Glover, 2013) como “*A metaheuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms. The term is also used to refer to a problem-specific implementation of a heuristic optimization algorithm according to the guidelines expressed in such a framework.*” [Un metaheurístico es un método algorítmico de alto nivel independiente del problema que proporciona un conjunto de pautas y estrategias para desarrollar algoritmos de optimización heurística. El término también se utiliza para referirse a la implementación en un problema específico de un algoritmo de optimización heurística de acuerdo con las directrices expresadas en dicho método]. Estas metodologías se han desarrollado significativamente durante los últimos 20 años debido a los grandes resultados que se han obtenido de su aplicación.

Del mismo modo que los heurísticos resuelven los problemas mediante procesos iterativos donde se evalúan posibles soluciones de acuerdo con unas restricciones; también comparten algunas características deseables como la sencillez, robustez, eficiencia, etc. Si se estableciera una jerarquía, los metaheurísticos se situarían en un escalón por encima de los heurísticos (Gómez, 2010).

Se han estudiado multitud de metaheurísticos para hacer frente a los distintos problemas combinatorios con los que se trabaja hoy en día, como pueden ser: Algoritmos Genéticos, Recocido Simulado, Búsqueda Tabú, Algoritmos Meméticos, *Greedy Randomize Adaptive Search Procedure* (GRASP), Colonia de Hormigas, etc. En su tesis, Gómez Cámara hace un repaso a los trabajos pioneros en cada una de las tipologías (Gómez, 2010). Sin embargo, para no hacer muy extensa esta sección, aquí se presentarán aquellos que son aplicables al VRP. Para ello se van a seguir las revisiones realizadas en (Laporte, 2006) y (Gómez, 2010), donde se distinguen tres tipos de metaheurísticos aplicados al VRP: búsqueda local, algoritmos basados en población y algoritmos basados en mecanismos de aprendizaje.

- **Búsqueda Local**

Estos algoritmos, al igual que sus homólogos heurísticos, empiezan desde una solución inicial y en cada iteración realizan un movimiento hacia otra solución que está en su vecindario, sustituyendo a la solución anterior. En este caso, lo que añaden los metaheurísticos son condiciones de búsqueda adicionales para cubrir más posibles soluciones. Por ejemplo, utilizan memoria para almacenar los resultados que ha ido encontrando en los movimientos y así saber hacia dónde tienen que dirigirse en las siguientes iteraciones. O bien, cuando no pueden mejorar en un vecindario, existen condiciones para saltar a otras regiones. (Laporte, 2006)

Algunos de los metaheurísticos que se incluyen dentro de esta familia son:

- *Tabu Search*

La Búsqueda Tabú fue presentada por Fred Glover en (Glover, 1986) como una forma de implementar la estrategia oscilante de asignación que el mismo había descrito años antes en (Glover, 1977). El término tabú se deriva de la funcionalidad de estos procedimientos de crear una lista, lista tabú, donde se almacenan durante un número determinado de iteraciones la información de los movimientos recientes.

El objetivo de estas listas no es prevenir que se repita el movimiento, sino evitar revertir el efecto de los movimientos realizados para llegar al óptimo actual. El “tabú” es realizar movimientos que generen bucles que van en contra de las mejoras realizadas. Sin embargo, para evitar que se prohíban movimientos atractivos se aplican algoritmos que cancelan el tabú: son los conocidos como criterios de aspiración. Uno de los más conocidos es el de permitir un movimiento clasificado como tabú, aunque empeore la función objetivo. Incluso se aceptan soluciones no factibles, violaciones de restricciones de capacidad y de recorrido máximo penalizando con factores ponderados la función objetivo, como se plantea en el algoritmo TabuRoute (Gendreau et al., 1994).

Otras funciones que suelen acompañar a estos métodos son las de diversificación o intensificación. En el algoritmo *Granular Tabu Search* (Toth & Vigo, 2003) se plantean estrategias de intensificación para profundizar en la búsqueda dentro de una zona que tenga buenas previsiones. Por otro lado, en el algoritmo *Unified Tabu Search* (Cordeau et al., 2001), además de permitir soluciones no factibles, se introducen técnicas de diversificación cuyo cometido es explorar otras zonas del espacio de búsqueda. Por último, si no se establece un criterio de finalización, el algoritmo podría estar iterando eternamente, para ello se suelen establecer límites de iteraciones.

- *Variable Neighbourhood Search*

Se definen una serie de vecindarios que serán explorados a lo largo del algoritmo. En estos algoritmos se comienza la exploración en un vecindario dado y, una vez se ha encontrado el óptimo local, se salta al siguiente vecindario. Si en el nuevo vecindario se encuentra una solución que mejore a la anterior, vuelve a iniciar el proceso desde el primer vecindario, si por el contrario no lo mejora, continúa con el siguiente vecindario. (Mladenović & Hansen, 1997)

En base a esta metodología se han desarrollado variantes como el *Very Large Scale Neighbourhood Search*, donde se establecen vecindarios de gran tamaño, lo que hace que se incremente mucho el esfuerzo computacional (Ergun, 2001). También similar a este último, el *Adaptive Large Neighbourhood Search* que combina las técnicas propias de la metodología de búsqueda en vecindarios con heurísticas de inserción y extracción de nodos, tratando problemas de gran tamaño. Además, este método es muy versátil puesto que se ha probado su efectividad resolviendo muchas de las variantes existentes del VRP. (Pisinger & Ropke, 2007)

- *Simulated annealing*

En castellano, se conoce como Recocido Simulado. Recibe este nombre debido al parecido que presenta con el proceso de fabricación del mismo nombre. En el recocido se calienta el material hasta altas temperaturas y después el enfriamiento se realiza de forma lenta y controlada, con el objetivo de obtener unas determinadas propiedades. Una de las ventajas de este procedimiento es la capacidad de escapar de los óptimos locales (Gómez, 2010).

En este caso la búsqueda local se realiza de forma aleatoria y no necesita de memoria para almacenar posiciones anteriores. En cada iteración se propone una modificación aleatoria, si el efecto que tiene sobre la función objetivo es positivo, se acepta. En caso contrario, la nueva solución se acepta en función de un parámetro de control denominado temperatura, aprovechando el símil con el proceso físico. Este parámetro es utilizado para controlar la exploración de las soluciones. Cuando la temperatura es muy alta (al principio del proceso) se aceptan las modificaciones, aunque puedan empeorar el coste; mientras que cuando las temperaturas son bajas, únicamente se permiten cambios que mejoren el óptimo actual. (Kirkpatrick et al., 1983)

A partir de los estudios sobre el Recocido Simulado han surgido modificaciones como el *Threshold Accepting* o Umbral de Aceptación, que se basa en la metodología que se acaba de explicar, pero eliminando el carácter aleatorio, se introduce un límite determinista que disminuye a la vez que lo hace la temperatura. Este límite es el que determina qué nuevas soluciones se aceptan en caso de que empeoren la función objetivo. De esta

manera, el algoritmo del Umbral de Aceptación acepta todas las nuevas configuraciones que “no sean mucho peores” que la antigua, mientras que el Recocido Simulado aceptaba soluciones peores únicamente para pequeñas probabilidades. (Dueck & Scheuer, 1990)

- **Algoritmos basados en Población**

El mecanismo básico utilizado por estas técnicas es la combinación de soluciones, es decir, se obtienen realizando combinaciones entre los parámetros que componen la solución. Se tratan a las soluciones como individuos y se modifican sus características para guiar la búsqueda. Algunos de los más conocidos son:

- *Genetic Algorithms*

Los Algoritmos Genéticos basan su planteamiento en la teoría Darwinista de la evolución. Son algoritmos evolutivos que utilizan los principios de selección natural. (Laporte, 2006) En cada iteración se tiene un conjunto de soluciones que se obtienen cruzando los parámetros de las soluciones anteriores para generar descendientes que combinen las características de sus predecesores. Además, para explorar un área mayor, se introducen mutaciones aleatorias que diversifican las zonas de búsqueda, estas mutaciones reemplazan a las características de los individuos menos prometedores. (Booker et al., 2005)

Un planteamiento híbrido se presenta en el algoritmo desarrollado por Prins, que toma la solución de un problema de VRP y elimina los delimitadores de ruta, transformando la solución en una válida para un TSP. Al final de este proceso, vuelve a reconstruir la solución del VRP con la esperanza de encontrar una mejora. (Prins, 2004)

- *Memetic Algorithm*

Su traducción al castellano es Algoritmo Memético y combinan las funcionalidades de los algoritmos evolutivos con la búsqueda por entornos. En este caso la información no pasa de padres a hijos “inalterable”, sino que cada solución adapta una característica determinada a conveniencia. Se elimina parte de la aleatoriedad de las mutaciones establecidas en el algoritmo genético. (Moscato & Cotta, 2003)

- *Adaptative Memory Procedure*

El Procedimiento de Memoria Adaptativa plantea una primera fase de ejecución de una búsqueda tabú, donde se almacenan las mejores soluciones. Una vez se obtiene el conjunto de candidatos, se le aplica la metodología desarrollada en los algoritmos genéticos, se combinan los candidatos entre sí con la finalidad de encontrar mejores soluciones (Rochat & Taillard, 1995). Otro planteamiento parecido aplicado directamente sobre un CVRP se puede encontrar en (Tarantilis & Kiranoudis, 2002).

- **Algoritmos basados en Mecanismos de Aprendizaje**

Dentro de esta categoría, se puede destacar principalmente el *Ant Colony Optimization*. Esta técnica probabilística pretende imitar el comportamiento de las colonias de hormigas en su búsqueda de alimento. Las hormigas se guían por la cantidad de feromonas que desprenden para establecer una ruta de abastecimiento.

Desde el nido parten individualmente varias hormigas y se supone que no volverán hasta encontrar alimento. De esta manera, se puede decir que la primera en regresar habrá encontrado el camino más corto, dejando el doble de feromonas en su ruta debido a que ha ido y ha vuelto. Entonces, tras esta primera búsqueda, el nivel de feromonas que hay en este camino es mayor que en los otros, por lo que otras hormigas seguirán el rastro de la primera, provocando a su vez que se incremente la intensidad de feromonas, haciendo que cada vez más hormigas sigan ese camino.

Traduciéndolo al terreno de la optimización, se generan soluciones aleatorias en cada ciclo que son exploradas y evaluadas según la cantidad de feromonas. Durante esta exploración, el algoritmo almacena los tramos que aparecen con frecuencia en soluciones buenas, con la intención de volver a utilizarlos en la exploración de la siguiente iteración (Dorigo & Stützle, 2004).

### 2.3.3 Métodos exactos

Las metodologías exactas devuelven soluciones óptimas exactas, a cambio de emplear tiempos de computación muy altos. De hecho, como se ha comentado en varias ocasiones, solo son capaces de lidiar con problemas que tienen un número limitado de puntos. En la literatura no se encuentra un consenso sobre el número máximo de nodos a tratar mediante técnicas exactas. Algunos autores apuntan en torno a unos 50

puntos (Machuca de Pina et al., 2018); mientras que otros señalan que 100 nodos es el límite para encontrar una solución en un tiempo relativamente aceptable (Baldacci et al., 2007). El algoritmo presentado por Fukasawa consigue resolver problemas con 135 nodos (Fukasawa et al., 2006).

Las técnicas utilizadas por esta familia de procedimientos se basan principalmente en una de estas tres metodologías: Búsqueda Directa en Árbol, Programación Dinámica o Programación Lineal Entera (Laporte & Nobert, 1987). La primera de ellas es la que ofrece menos potencial de cálculo y la que menos se ha desarrollado. Por otra parte, los algoritmos basados en programación dinámica han avanzado, pero tienen todavía áreas de mejora que poder explotar. Por último, la programación lineal entera basándose en los algoritmos de *Branch and Bound* o *Branch and Cut* ha demostrado resolver casos para un número considerable de puntos y de manera eficiente, siendo la metodología más exitosa hasta ahora (Baldacci et al., 2007; Laporte & Nobert, 1987). A continuación, se profundizará en estas tres tipologías mencionadas, prestando especial atención a las Programación Lineal Entera, por ser la metodología utilizada en este trabajo.

- *Direct Tree Search Methods*

Este tipo de técnicas reciben este nombre debido a la estructura en forma de árbol que toma la representación gráfica de la búsqueda de la solución óptima. En ellos se generan posibles rutas a medida que son necesarias. En cada paso o nivel del árbol se van generando bifurcaciones que dan lugar a nuevas soluciones posibles (Little et al., 1963). Aplicados al VRP, se puede decir que es una construcción secuencial de rutas mediante bifurcación y acotamiento.

La aplicación de estas metodologías conlleva en muchos casos realizar algunas relajaciones para poder calcular la solución como por ejemplo resolver un primer problema de asignación para tener una solución de partida, o la resolución del problema asociado de árbol de expansión más corto, obteniendo de esta manera un límite inferior.

Una de las estrategias de ramificación de árboles es la bifurcación en arcos, en la cual las ramas del árbol se crean incluyendo un nuevo arco en la solución o excluyéndolo de esta. La creación de las ramificaciones se produce cuando se da una de estas tres situaciones: la carga del vehículo ha excedido la capacidad, la distancia total supera la restricción de máximo recorrido, o bien la demanda de las ciudades que todavía no se contemplan en ninguna ruta, supera la capacidad de los vehículos que están sin utilizarse (Laporte & Nobert, 1987).

Por otro lado, la bifurcación en rutas establece que cada nivel del árbol corresponde con la creación de una nueva ruta para un vehículo. De manera que para una flota de unos  $m$  vehículos, como mucho el árbol tendrá  $m$  niveles. Para un determinado nodo  $i$ , se denominan  $j$  a los nodos que no están incluidos todavía en ninguna ruta. Para saber que nodo  $j$  irá después de  $i$ , se realiza una selección donde se van descartando candidatos en función de distintas consideraciones, como por ejemplo si el incluir  $j$  hace que se violen las restricciones, o si existe otro nodo  $j$  cuyo coste asociado sea menor. (Laporte & Nobert, 1987)

En los problemas de optimización, la eficiencia de cualquier algoritmo se mide en lo ajustado que sean los límites inferiores en el óptimo, para los problemas de minimización. En (Christofides et al., 1981a), se presenta un algoritmo de búsqueda en árbol incorporando límites inferiores obtenidos de la resolución del problema *Shortest Spanning k-degree Centre Tree*, que resuelve el m-TSP asociado para obtener dicho límite inferior; y también de la resolución del algoritmo *q-routes*, donde se generan rutas únicas que se resuelven como un TSP y que entre todas cubren el problema original.

- *Dynamic Programming*

Los procedimientos de Programación Dinámica se basan en descomponer los problemas en otros más simples, almacenando las soluciones una vez resueltos estos subproblemas. De esta forma, si el mismo subproblema se repite varias veces, se puede aprovechar la solución almacenada, ahorrando tiempo de computación. Se utilizan relajaciones en la formulación matemática del problema original para conseguir que los subproblemas se parezcan entre ellos (Laporte, 1992).

En la literatura es recurrente el uso de las *state-space relaxations*, que relaja las restricciones del problema creando un problema asociado cuyo espacio de búsqueda es menor, la resolución del problema asociado proporciona un límite inferior que se integra directamente en el modelo original (Christofides et al., 1981b). En esta misma línea de investigación, se ha estudiado el método *ngL-tour* que mejora el anterior proporcionando límites inferiores más ajustados para problemas con restricciones de ventanas de tiempo (Baldacci et al.,

2012).

El principal problema que se presenta en esta metodología es la conocida en inglés como *curse of dimensionality*, la maldición de la dimensión o simplemente el efecto Hughes. Este problema ocurre al estructurar datos, a medida que crece la dimensión del problema, el espacio de las soluciones factibles crece exponencialmente. En Programación Dinámica, esto puede ocurrir al descomponer en problemas más sencillos, el número de combinaciones posible puede volverse inabordable (Powell, 2011). Sin embargo, se han llegado a tratar problemas de enrutamiento de vehículos de hasta 50 nodos haciendo uso de esta técnica (Christofides, 1976).

- *Integer Linear Programming*

En este proyecto, tal y como se estableció en la Introducción, se va a resolver el problema de optimización de rutas de vehículos haciendo uso de la *Mixed-Integer Linear Programming*, o en castellano Programación Lineal Entera-Mixta, que pertenece a este subgrupo de métodos exactos. En estos problemas se parte de una formulación matemática en la que se tiene una serie de variables de decisión sujetas a unas restricciones y una función objetivo.

La formulación del VRP utilizada en este trabajo es la Formulación de Flujo de Vehículos, en particular la de 3 índices, como se presentó en el apartado 2.2.2. Para estas formulaciones es común la aplicación de las técnicas *Branch and Bound*, o las más modernas *Branch and Cut*, utilizadas en los resolutores comerciales como CPLEX, que ha sido el software con el que se ha trabajado.

Los procesos fundamentales en los que se basan los algoritmos *Branch and Bound* son principalmente dos: la búsqueda de la solución y las relajaciones lineales. En el primero de ellos se lleva a cabo una bifurcación (*Branch*), que consiste en fijar unos valores para alguna o algunas de las variables de decisión y resolver los subproblemas que se generan mediante estas suposiciones; mientras que las relajaciones lineales se relacionan con la acotación del espacio (*Bound*) y simplifican la resolución de los subproblemas. A medida que la búsqueda avanza se desechan aquellas ramificaciones cuyo valor de la función objetivo sea mayor que la mejor solución hasta el momento. (Gorostegui, 2011)

Otra forma de ver las técnicas *Branch and Bound* es como una estrategia de divide y vencerás, donde durante la fase de bifurcación se admiten relajaciones que generan subproblemas que se resuelven por separado. Estos problemas son realmente candidatos, porque en uno de ellos estará la solución óptima. De cada subproblema se pueden sacar cuatro resultados: encontrar una solución factible que mejore la que existe; que el subproblema esté vacío, lo que conlleva que se descarte o “pode” esa rama; otra opción es que no sea factible, en cuyo caso, se puede comprobar si es un límite inferior, chequeando si es menor o igual que el actual límite superior, establecido por la solución óptima actual; si esto tampoco ocurre se vuelve a dividir el problema, ampliando la lista de candidatos. Este proceso se repite hasta terminar con la lista de candidatos o lo que es lo mismo acabar con los subproblemas, quedando de este modo determinado el mejor valor de la función objetivo. (Ralphs, 2003)

Los primeros trabajos de esta técnica aunaban métodos resolutivos aplicados al CVRP y al TSP, tales como la relajación de las restricciones de eliminación de ciclos o la eliminación de rutas prohibidas. En (Laporte et al., 1987) se presenta un algoritmo paso por paso que resume esta metodología: en primer lugar, se obtiene una primera solución factible utilizando algún heuristicó; a partir de esta solución, se plantea la búsqueda de los nodos; se selecciona el nodo que tenga el mejor resultado del problema de asignación asociado; se plantea la bifurcación y el acotamiento tomando como parte de la solución el nodo previamente seleccionado; se vuelve a resolver el problema de asignación pero con la nueva información y finalmente se chequea la factibilidad de la solución.

Estas primeras implementaciones se encontraron con las limitaciones de espacio de almacenamiento de los ordenadores. De hecho, se limitaron a casos teóricos para testear la capacidad resolutiva de los algoritmos y a algunos casos prácticos a pequeña escala, como la distribución de gas planteada en (Radharamanan & Choi, 1986).

En la búsqueda de una mejora en la eficiencia del *Branch and Bound*, se introdujeron los planos de corte que reducían el tamaño del árbol de búsqueda, lo que se traduce en un ahorro considerable en el tiempo de computación. Surgieron de esta forma las metodologías *Branch and Cut*. Esta mejora teórica en las técnicas, acompañadas del desarrollo de la capacidad de computación de los equipos informáticos, han provocado que el *Branch and Cut* sea considerado la metodología exacta más interesante. (Baldacci et al., 2008)

La principal diferencia entre los algoritmos *Branch and Bound* y *Branch and Cut* reside en que la validez de las desigualdades que se generan en cada nodo del árbol de búsqueda es válida globalmente, de manera que se pueden reutilizar en otros nodos a posteriori. Las desigualdades más efectivas se almacenan en *cut pools*, ahorrando el tiempo de recalcular estos cortes en los nodos donde vuelva a ser necesario. (Ralphs, 2003)

Entre las desigualdades más efectivas, suelen encontrarse las restricciones de capacidad, también conocidas por su funcionalidad de eliminar la generación de subciclos. En (Augerat et al., 1995), se plantean algoritmos de separación para estas desigualdades, obteniendo límites inferiores mejores que los que se habían conseguido hasta el momento y llegando a resolver dos casos de 135 clientes.

En (Lysgaard et al., 2004) se revisan otras desigualdades distintas a las de capacidad, que también se traducen en planos de corte válidos como las desigualdades de capacidad enmarcadas o los planos de cortes de Gomory. Para cada una de ellas se describen los algoritmos de separación particulares de su clase. También se propone una estrategia de separación que distingue entre las distintas clases, para saber qué planos de corte aplicar y cuándo.

Otro enfoque se propone en (Baldacci et al., 2004), donde se introducen modificaciones en la formulación matemática del problema según el flujo de vehículos con dos índices. El método *Branch and Cut* presentado en este artículo propone resolver en cada nodo el problema relajado obtenido de la nueva formulación. Para calcular el límite inferior, en cada iteración se realizan a su vez dos procesos iterativos. Se establecen dos límites de iteraciones para identificar las restricciones violadas tanto de flujo como de capacidad.

Otra forma de afrontar estos problemas es mediante otras formulaciones, por ejemplo, la *Set Partitioning formulation* que asocia una variable binaria a cada ruta factible. En (Baldacci et al., 2008), plantean un método basado en esta formulación que establece un límite inferior al problema, encontrando una solución cercana al óptimo del problema dual gracias a las relajaciones lineales, que eliminan la obligatoriedad de algunas variables de ser números enteros.

Las combinaciones de distintas formulaciones del VRP también se utilizan para resolver estos problemas desde el planteamiento de la programación lineal. En concreto (Fukasawa et al., 2006) combina la *Vehicle Flow formulation* con la *Set Partitioning formulation*, vinculando las variables de decisión de ambas formulaciones. En este método, hace uso de las *Pricing and Cut generation techniques* que hacen uso de los procedimientos de *Branch and Cut* y *q-routes* comentados anteriormente. El subproblema de ahorro consiste en encontrar el menor coste reducido de las posibles rutas generadas, para después proceder con la bifurcación y el corte que se ha comentado en este mismo apartado para las técnicas de *Branch and Cut*.

# 3 PRESENTACIÓN DEL CASO

---

**A**ntes de desarrollar el modelo en el que se ha basado el estudio, en este capítulo se introducen el marco y el contexto sobre el que se aplicará la formulación propuesta en el Capítulo 4. Para ello, se ha tomado como referencia el *Plan Provincial de Residuos no Peligrosos de la Provincia de Sevilla (2020-2035)* (Diputación de Sevilla, 2019), en adelante PPRNP o simplemente el Plan. Este documento ofrece una visión general de la situación actual de la recogida de residuos municipales en la provincia, así como un plan estratégico con vistas al año 2035. Se utilizará la propuesta en materia de gestión de dicho plan como entrada al modelo utilizado en el presente trabajo.

## 3.1 Presentación del Plan de Residuos no Peligrosos de la Provincia de Sevilla

El PPRNP surge de la necesidad de revisar y actualizar las medidas y programas de prevención y gestión de residuos de la Provincia de Sevilla. Del mismo modo, lanza una propuesta para modernizar y adaptar las infraestructuras actuales, con vistas a cumplir las directrices europeas planteadas para el año 2035. También recopila datos de la producción de residuos municipales, la caracterización de estos y ofrece una previsión de cómo evolucionarán en los próximos años.

Este Plan se nutre de una serie de documentos y planes previos de carácter autonómico y estatal, tales como el Plan Director Provincial de Gestión de Residuos Sólidos Urbanos de Andalucía (1997-2002) (Junta de Andalucía, 1999) o el Plan Director Territorial de Residuos No Peligrosos de Andalucía (2010-2019) (Junta de Andalucía, 2010), que fue revisado en el año 2016 en relación con la publicación del Plan Estatal Marco de Gestión de Residuos (2016-2022) (Gobierno de España, 2015).

El PPRNP se plantea un horizonte temporal que comenzó con su aprobación en 2019 y se prevé que se desarrolle hasta el año 2035. Por esta razón se incluyen también planteamientos dirigidos a reducir la generación de residuos, garantizar la recogida selectiva, introduciendo la recogida de biorresiduo y mejorar el tratamiento de las distintas fracciones, aumentando el aprovechamiento de los residuos.

## 3.2 Modelo de gestión de residuos de la Provincia de Sevilla.

Como se ha comentado en el epígrafe anterior, el Plan plantea modificaciones en la gestión de los residuos municipales. En torno a este punto se desarrollará el modelo propuesto en la Sección 4. A continuación, tomando como partida la situación actual, se describirá la propuesta organizativa a futuros que presenta el Plan.

En primer lugar, la normativa vigente, Ley 5/2010, de 11 de Junio, de Autonomía Local de Andalucía, 2010, otorga las competencias en materia de recogida de residuos a los municipios, tal y como establece en el artículo 9 de competencias municipales: “*Los municipios andaluces tienen las siguientes competencias propias: (...) 6. Ordenación, gestión, prestación y control de los servicios de recogida y tratamiento de residuos sólidos urbanos o municipales, así como la planificación, programación y disciplina de la reducción de la producción de residuos urbanos o municipales*”. Sin embargo, dicha legislación contempla la agrupación

de municipios para la prestación de este tipo de servicios. Dentro de la Provincia de Sevilla, existen dos tipologías de agrupaciones supramunicipales: la mancomunidad y el consorcio.

Según la Real Academia Española, se define la mancomunidad como una “*corporación o entidad legalmente constituida por agrupación de municipios y provincias*”, mientras el consorcio en su tercera acepción del diccionario se describe como una “*agrupación de entidades para negocios importantes*”. Estas formaciones se rigen en base a unos estatutos que regulan sus competencias y establecen sus órganos de gobierno, aprobados en el acto de constitución de la entidad por todos los municipios que la integran, los cuales no tienen por qué ser colindantes. La diferencia entre estas dos figuras legales no tiene gran relevancia para el desarrollo del trabajo y en muchas ocasiones la elección viene motivada por factores geográficos e históricos (Font & Parrado Diez, 2000). De hecho, a lo largo del trabajo los términos mancomunidad y consorcio se han utilizado indistintamente.

Este modelo de gestión supramunicipal empezó a desarrollarse en la Provincia de Sevilla en el año 1982 con la constitución de la Mancomunidad de los Alcores. Desde entonces, se ha impulsado la creación de seis mancomunidades y un consorcio que acaparan las competencias sobre gestión de residuos de los distintos municipios que las integran, bajo el control y la supervisión de la Diputación de Sevilla. Surgieron para facilitar la prestación de servicios que un municipio individualmente no podría ofrecer, debido principalmente a motivos económicos. Estas entidades supramunicipales, reciben el nombre de Unidades de Gestión de Residuos (UGRs) y en el Plan están numeradas del 1 al 7 para facilitar la identificación de estas, numeración que se ha adoptado también en este texto.

Las Unidades de Gestión de Residuos (UGRs) son unidades territoriales de gestión, especializadas en la prestación de servicios relacionados con la recogida y el tratamiento de los residuos municipales. Como se ha comentado, son entidades supramunicipales con personalidad jurídica propia que agrupan a varios municipios y comparten el ejercicio de la competencia de la gestión sobre los residuos municipales. En los siguientes apartados, se presenta una visión general de la recogida de residuos, así como la distribución de los 106 municipios que conforman la Provincia de Sevilla, la mayoría de ellos agrupados en torno a estas entidades supramunicipales. Se definirán todas las UGRs presentes en la Provincia de Sevilla y las infraestructuras que tienen cada una.

### 3.2.1 Proceso de recogida de residuos

El procedimiento que se contempla en este trabajo es una recogida municipal, llevada a cabo en los distintos municipios mediante camiones diseñados para ello. Existen muchos tipos de camiones: carga trasera, carga lateral, neumática, por volteo, etc. Así como diferentes modelos de recogida: contenedores, neumática, soterrada, puerta a puerta, etc. Las características de cada uno de ellos los hacen más apropiados para una u otra localidad. Sin embargo, no se discutirá cuál es el vehículo más interesante, ni la modalidad de recogida más conveniente. Simplemente, en la Sección 4, se expondrán las hipótesis que se han tomado en este aspecto.

Para el desarrollo de esta actividad, las mancomunidades son titulares de una serie de infraestructuras, que pueden ser gestionadas directamente por ellos, o bien por empresas concesionarias privadas. Sin embargo, no todas las UGRs tienen infraestructuras propias, ni con las mismas capacidades. Esto depende fundamentalmente del tamaño de los municipios que integran las entidades y de su número. A continuación, se enumeran los principales tipos de instalaciones que intervienen en el proceso y cuál es su función:

- **Depósitos de vehículos:** Desde aquí parten los camiones hacia los municipios al inicio de la jornada y es donde se estacionan mientras no están funcionando. En estas instalaciones también se realizan tareas de mantenimiento y limpieza de estos vehículos.
- **Estaciones de transferencia:** Son puntos intermedios del proceso de recogida, aquí es donde los camiones descargan los residuos que han recogido en los municipios. En estas instalaciones se realizan tareas de clasificación para finalmente compactar los residuos y meterlos en unos contenedores especiales que serán recogidos por otros camiones para llevarlos a las plantas de tratamiento. El principal motivo para la creación de estas estaciones es reducir el desplazamiento de los camiones de recogida, lo que se traduce en un incremento de su vida útil y un considerable ahorro en costes. Algunas UGRs no hacen uso de estas estaciones de transferencia y llevan sus residuos directamente a la planta de tratamiento.
- **Plantas de tratamiento:** Es la última parada de los residuos desde que se recogieron en los contenedores de los municipios. Estas plantas pueden recibir los residuos desde las estaciones de

transferencia o directamente de la recogida municipal. En estas instalaciones los residuos son tratados para su eliminación, depositándolos en vertederos controlados.

Las instalaciones de transferencia y tratamiento están en funcionamiento en la actualidad y no es motivo de este TFM optimizar su emplazamiento, sino minimizar el recorrido que realizan los camiones en la recogida desde los municipios hasta el centro asignado, que según la UGR que sea puede ser una estación de transferencia o una planta de tratamiento. Con esta definición, también se excluye de los objetivos del trabajo el posible desplazamiento intermedio entre una estación de transferencia y la planta de tratamiento correspondiente.

La configuración de infraestructuras que se ha utilizado en este trabajo es la propuesta por el PPRNP. Por esta razón, es conveniente resaltar que el planteamiento realizado en el Plan ha intentado garantizar que la distancia máxima desde cualquier municipio a su destino ya sea estación de transferencia o planta de tratamiento, es de unos 40 km. La Figura 3.1 sacada del PPRNP y realizada por ECOEMBES, proporciona el coste que supone la recolección y la transferencia basándose en la distancia recorrida por los camiones y como se puede observar establece dicho valor máximo en unos 48 km, punto en el que coinciden ambas curvas. Aunque en esta gráfica se especifica que es para estaciones de transferencia, su validez es extensiva a las plantas de tratamiento que desempeñan a la vez el papel de estación intermedia y final.

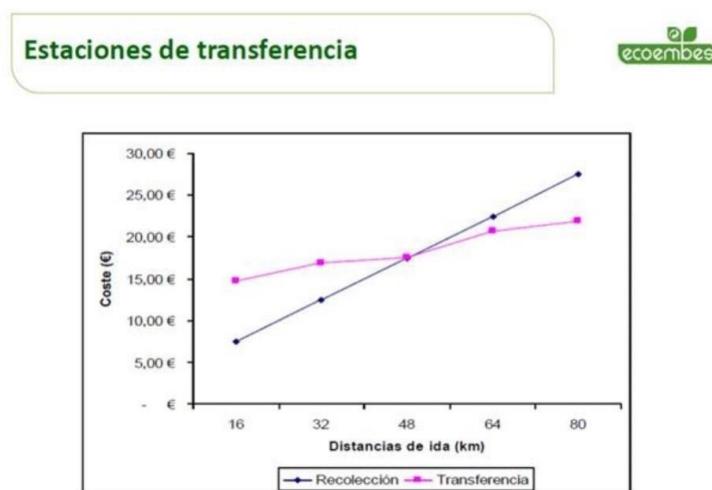


Figura 3.1. Distancia máxima hasta estación intermedia. (ECOEMBES, 2018)

### 3.2.2 Unidades de Gestión de Residuos (UGRs) de la Provincia de Sevilla

Toda la información que se resume en este apartado se corresponde con la propuesta a futuros del PPRNP. Aunque los datos que se manejan son actuales, se utilizan los supuestos presentados en dicho plan como línea base del trabajo. Esto es de especial relevancia en el reparto de los municipios entre las distintas UGRs. Actualmente, existen municipios que no pertenecen a ninguna de estas formaciones, así como municipios en entidades que no son las más “recomendables”, siguiendo los criterios que aquí se exponen. El Plan plantea un nuevo reparto de municipios que será el utilizado en la elaboración del presente texto.

Para entender cómo se organizan los municipios y los motivos que llevan a pertenecer a una u otra entidad, hay que conocer la distribución geográfica de la provincia y sus particularidades. Otro factor determinante han sido las decisiones políticas, puesto que en algunas ocasiones han provocado la salida de algunos municipios de determinadas UGRs y la incorporación a otras más “afines”.

Según los datos del Instituto Nacional de Estadística (INE), la Provincia de Sevilla comprende una superficie total de 14.036 km<sup>2</sup> y tiene una población de 1.942.389 personas. Lo que la convierte en la provincia más grande y poblada de toda Andalucía. Debido a su extensión, presenta varios tipos de climas y una gran diversidad orográfica, destacando la presencia del río Guadalquivir que hace de frontera natural y separa el territorio en dos partes. La provincia se divide en 9 comarcas, atendiendo a criterios territoriales y socioeconómicos. Sin embargo, en la Figura 3.2 se presenta una división comarcal agraria basada en criterios geográficos que divide el territorio en 7 comarcas, siendo esta división la que más ha influido en la formación

de las UGRs (Diputación de Sevilla, 2019).



Figura 3.2. División comarcal agraria de la Provincia de Sevilla. (Diputación de Sevilla, 2019)

La Provincia de Sevilla cuenta actualmente con un total de 106 municipios y 2 Entidades Locales Autónomas (ELAs). Sin embargo, en el Plan se presentan los datos de 105 municipios. Esto se debe a que los datos del municipio del Palmar de Troya (Utrera) que fue declarado como tal en 2018, están incluidos dentro de las estadísticas del municipio de Utrera y la producción de residuos en las ELAs se ha incluido dentro de los municipios a los que pertenecían: para Isla Redonda-La Aceñuela, se incluyen en el municipio de Écija, mientras que Las Marismillas se engloba dentro de la localidad de Las Cabezas de San Juan.

Como para el desarrollo de este trabajo lo que se necesita es una distribución de municipios, se ha decidido tomar como base la propuesta de división en UGRs que hace el plan, es decir, la implementación del modelo que aquí se presenta está supeditada a la reorganización de municipios propuesta por el PPRNP. A continuación, se detallan los municipios que quedan dentro de cada entidad una vez aplicada la mencionada reorganización, así como la producción de residuos y la población de cada municipio. Estos datos se han obtenido de las bases de datos del Instituto Nacional de Estadística del año 2019 y de los informes de producción de residuos recogidos en el *Plan Provincial de Residuos no Peligrosos de la Provincia de Sevilla (2020-2035)* (Diputación de Sevilla, 2019). En la Figura 3.3 se presenta esta propuesta de distribución por UGRs sobre un mapa, que como se puede apreciar comprende todo el territorio de la provincia, incluyendo todos los municipios que la integran.

**UGR nº1 – Mancomunidad de los Alcores**

Municipios	Habitantes	Residuos (ton/año)
Alcalá de Guadaira	75.279	30.458
Carmona	28.531	11.544
Coripe <sup>1</sup>	1.251	506
Dos Hermanas	133.968	54.203
El Coronil <sup>1</sup>	4.746	1.920
El Cuervo de Sevilla <sup>1</sup>	8.610	3.484
El Viso del Alcor	19.266	7.795
Las Cabezas de San Juan <sup>1</sup>	16.417	6.642
Lebrija <sup>1</sup>	27.524	11.136
Los Molares <sup>1</sup>	3.480	1.408
Los Palacios y Villafranca <sup>1</sup>	38.354	15.518
Mairena del Alcor	23.550	9.528
Montellano <sup>1</sup>	7.056	2.855
Sevilla	688.592	278.604
Utrera <sup>1</sup>	50.728	20.525

Tabla 3.1. Manc. de los Alcores (UGR nº1) - Habitantes y residuos

<sup>1</sup> Estos municipios no pertenecen actualmente a la mancomunidad. Forman parte de la propuesta de reorganización que plantea el PPRNP.

## UGR nº2 – Mancomunidad del Guadalquivir

Municipios	Habitantes	Residuos (ton/año)
Albaida del Aljarafe	3.197	1.381
Almensilla	6.080	2.626
Aznalcázar	4.586	1.981
Aznalcóllar	6.091	2.631
Benacazón	7.241	3.127
Bollullos de la Mitación	10.787	4.659
Bormujos	21.972	9.490
Camas <sup>1</sup>	27.509	11.881
Carrión de los Céspedes	2.544	1.099
Castilleja de Guzmán	2.821	1.218
Castilleja de la Cuesta <sup>1</sup>	17.418	7.523
Castilleja del Campo	629	272
Coria del Rio <sup>1</sup>	30.777	13.293
Espartinas	15.791	6.820
Gelves	10.184	4.398
Gines	13.420	5.796
Huévar del Aljarafe	3.015	1.302
Isla Mayor	5.839	2.522
La Puebla del Río	11.868	5.126
Mairena del Aljarafe	46.089	19.906
Olivares	9.394	4.057
Palomares del Río	8.767	3.786
Pilas	13.974	6.035
Salteras	5.530	2.388
San Juan de Aznalfarache <sup>1</sup>	21.416	9.250
Sanlúcar la Mayor	13.808	5.964
Santiponce	8.554	3.694
Tomares <sup>1</sup>	25.359	10.953
Umbrete	8.894	3.841
Valencina de la Concepción	7.751	3.348
Villamanrique de la Condesa	4.459	1.926
Villanueva del Ariscal	6.610	2.855

Tabla 3.2. Manc. del Guadalquivir (UGR nº2) - Habitantes y residuos

<sup>1</sup> Estos municipios no pertenecen actualmente a la mancomunidad. Forman parte de la propuesta de reorganización que plantea el PPRNP.

**UGR nº3 – Mancomunidad de Servicios de la Vega**

Municipios	Habitantes	Residuos (ton/año)
Alcalá del Río	12.029	6.415
Alcolea del Río	3.373	1.799
Brenes	12.471	6.651
Burguillos	6.716	3.582
Cantillana	10.684	5.698
Castilblanco de los Arroyos	4.864	2.594
El Castillo de las Guardas	1.443	770
El Garrobo <sup>1</sup>	790	421
El Madroño <sup>1</sup>	278	148
El Ronquillo <sup>1</sup>	1.364	727
Gerena	7.585	4.045
Guillena	12.788	6.820
La Algaba	16.374	8.732
La Rinconada	38.628	20.600
Lora del Río	18.662	9.952
Peñaflor	3.656	1.950
Tocina	9.501	5.067
Villanueva del Río y Minas	4.858	2.591
Villaverde del Río	7.818	4.169

Tabla 3.3. Manc. de Servicios de la Vega (UGR nº3) - Habitantes y residuos

<sup>1</sup> Estos municipios no pertenecen actualmente a la mancomunidad. Forman parte de la propuesta de reorganización que plantea el PPRNP.

#### **UGR nº4 – Mancomunidad de Municipios Sierra Norte**

Municipios	Habitantes	Residuos (ton/año)
Alanís	1.723	753
Almadén de la Plata	1.355	592
Cazalla de la Sierra	4.718	2.062
Constantina	5.896	2.577
El Pedroso	2.018	882
El Real de la Jara	1.503	657
Guadalcanal	2.627	1.148
La Puebla de los Infantes	2.991	1.307
Las Navas de la Concepción	1.559	681
San Nicolás del Puerto	596	260

Tabla 3.4. Manc. de Municipios de la Sierra Norte (UGR nº4) - Habitantes y residuos

#### **UGR nº5 – Consorcio de Medio Ambiente Estepa-Sierra Sur**

Municipios	Habitantes	Residuos (ton/año)
Aguadulce	2.020	724
Algamitas	1.255	450
Badolatosa	3.078	1.103
Casariche	5.456	1.954
El Rubio	3.408	1.221
El Saucejo	4.284	1.535
Estepa	12.505	4.479
Gilena	3.727	1.335
Herrera	6.461	2.314
La Roda de Andalucía	4.183	1.498
Lora de Estepa	18.662	6.685
Los Corrales	3.941	1.412
Marinaleda	2.627	941
Martín de la Jara	2.704	969
Pedrera	5.194	1.860
Pruna	2.603	932
Villanueva de San Juan	1.120	401

Tabla 3.5. Cons. de Medio Ambiente Estepa-Sierra Sur (UGR nº5) - Habitantes y residuos

**UGR nº6 – Mancomunidad de Municipios Comarca de Écija**

Municipios	Habitantes	Residuos (ton/año)
Cañada Rosal	3.320	1.393
Écija	39.873	16.735
Fuentes de Andalucía	7.111	2.984
La Campana	5.276	2.214
La Luisiana	4.576	1.921

Tabla 3.6. Manc. de Municipios Comarca de Écija (UGR nº6) - Habitantes y residuos

**UGR nº7 – Mancomunidad Intermunicipal Campiña 2000**

Municipios	Habitantes	Residuos (ton/año)
Arahal	19.526	8.238
La Lantuejuela	3.814	1.609
La Puebla de Cazalla	10.979	4.632
Marchena	19.457	8.209
Morón de la Frontera	27.627	11.656
Osuna	17.560	7.409
Paradas	6.908	2.914

Tabla 3.7. Manc. Intermunicipal Campiña 2000 (UGR nº7) - Habitantes y residuos

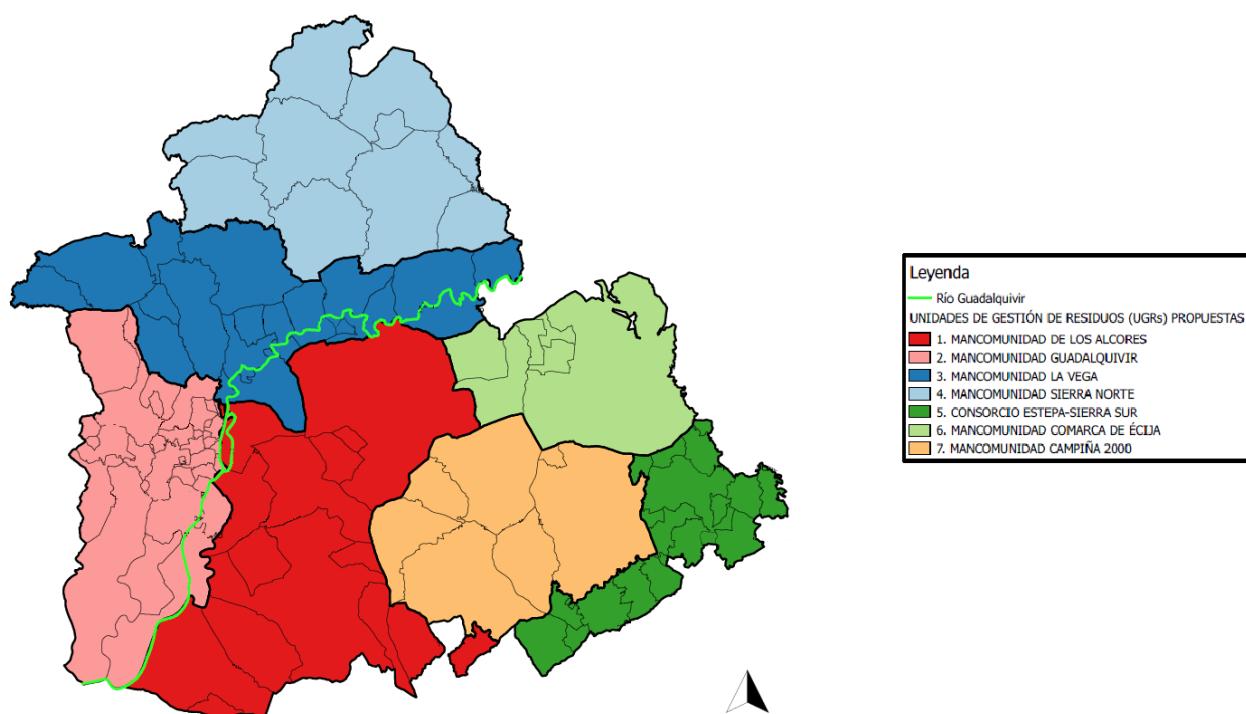


Figura 3.3. Unidades de Gestión de Residuos (UGRs) propuestas. (Diputación de Sevilla, 2019)

### 3.2.3 Evolución hacia un Consorcio Provincial de Residuos

La comunicación y coordinación entre las distintas entidades supramunicipales es prácticamente nula, identificándose en el PPRNP como una de las mayores debilidades en la Provincia. Una de las propuestas de este plan es terminar con esta falta de coordinación creando un órgano central, con el fin último de mejorar la eficiencia y la calidad del servicio, a la vez que se garantiza la estabilidad financiera del sistema, tal y como se puede leer en el Plan: “*Es ante esta realidad fragmentada y dispersa, sobre la que se ha de configurar el futuro modelo integral de gobernanza de la Provincia de Sevilla, dirigido hacia un modelo de base consorcial*” (Diputación de Sevilla, 2019). Se propone un Consorcio Provincial de Residuos, adscrito a la Diputación de Sevilla, sin que esto suponga la eliminación de las actuales UGRs, sino todo lo contrario, potenciar su uso y dotarlas de mayores recursos.

Dentro del proyecto de reestructuración y reunificación que se presenta en el PPRNP, este TFM propone que todas las entidades centralicen la recogida de la fracción resto, salvo algunas excepciones que se abordaran más adelante. Aunque bien podría hacerse extensivo al resto de residuos, se ha decidido utilizar este tipo de residuos debido a que supone el mayor porcentaje de la generación total. Actualmente, no todas las UGRs controlan la totalidad de la gestión de los residuos municipales, algunas simplemente se encargan de la recogida selectiva (vidrio, plásticos y papel). Otras se ocupan únicamente de la limpieza de la vía pública, mientras que otras realizan la gestión íntegra de todos estos servicios. Esto se define en los estatutos de cada agrupación y también estaría sujeto a cambio según lo expuesto en el Plan. En los siguientes párrafos, se especificarán y justificarán los criterios que se han seguido para incluir o no a un municipio dentro de la recogida conjunta de la fracción resto.

La Mancomunidad del Guadalquivir lleva años aplicando la recogida conjunta de la fracción resto. Por esta razón, se pretende imitar su modelo para el resto de UGRs. Sin embargo, no todos los municipios de la Mancomunidad del Guadalquivir hacen uso de este servicio. Por ejemplo, Mairena del Aljarafe perteneciente a esta mancomunidad realiza la recogida de la fracción resto por su cuenta, esto se debe a que su producción de residuos es lo suficientemente grande como para tener un sistema de recogida propio y que este sea rentable. Con total seguridad, en las otras UGRs habrá municipios que tengan una situación similar, por lo que se han establecido unos criterios máximos de producción y población según los cuales se decidirá si un municipio puede ofrecer este servicio por sí mismo o si es recomendable que se asocie a otras localidades para su prestación.

Para establecer estos criterios, es conveniente tener una visión global del reparto de la producción de residuos en toda la provincia. Atendiendo a los datos que ofrece la Figura 3.4, se observa como la Mancomunidad de los Alcores concentra la mayor parte de los residuos de toda la provincia, puesto que está compuesta por 3 de los municipios con mayor población: Sevilla, Dos Hermanas y Alcalá de Guadaíra. De esta gráfica también se puede extraer la conclusión de que la fracción resto es la de mayor volumen en todas las organizaciones, es por ello por lo que se decidió tomarla como representativa de la situación para el desarrollo del trabajo.

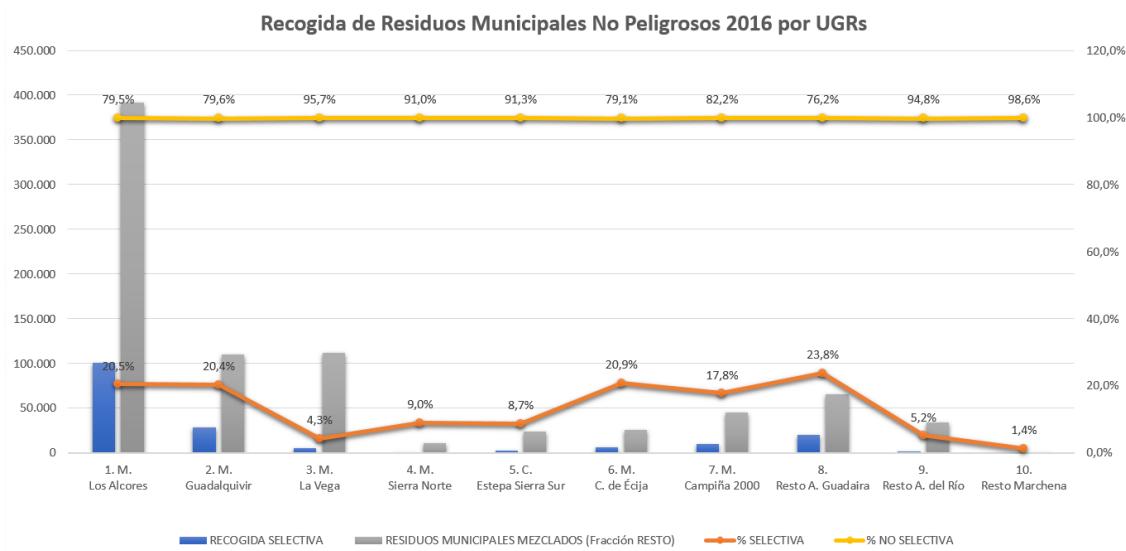


Figura 3.4. Reparto de residuos por UGRs en la Provincia de Sevilla. (Diputación de Sevilla, 2019)

También es necesario conocer la densidad demográfica de la provincia y el reparto poblacional entre las distintas UGRs. La Figura 3.5 es una infografía donde se clasifican los municipios según el número de habitantes. Sevilla cuenta con 17 municipios de más de 20.000 habitantes, 45 municipios con una población mayor de 5.000 personas y 43 municipios con menos de 5.000. Comparando este mapa y el ofrecido en la Figura 3.3, se ve como los municipios más poblados pertenecen a la Mancomunidad de los Alcores o a la Mancomunidad del Guadalquivir, datos que son coherentes con la información de producción de residuos comentados anteriormente. De hecho, según los datos facilitados por la Diputación de Sevilla en el Plan, la mancomunidad de los Alcores congrega a casi el 50% de la población servida, seguida por la Mancomunidad del Guadalquivir con el 13,2% del total (Diputación de Sevilla, 2019).

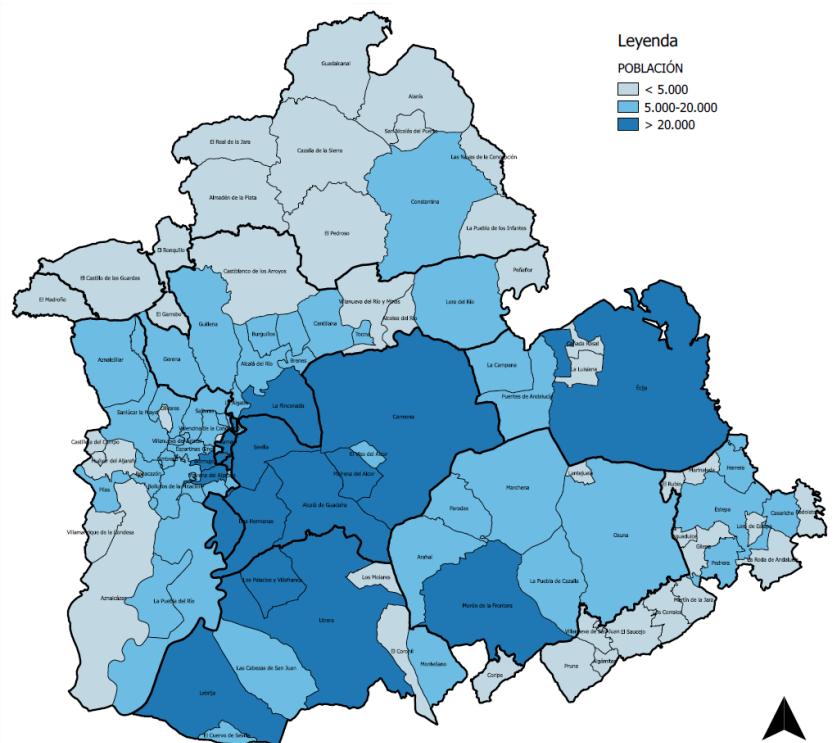


Figura 3.5. Mapa de densidad demográfica de la Provincia de Sevilla. (Diputación de Sevilla, 2019)

Considerando la información ofrecida en los párrafos anteriores y las tablas con los datos de producción y poblacionales del apartado 3.2.2, así como el ejemplo de la Mancomunidad del Guadalquivir, todo municipio que supere una generación de residuos de 15.000 ton/año o bien una población de 33.000 habitantes, se entiende que es capaz de autogestionarse y por consecuencia quedan fuera de los límites del modelo propuesto en este trabajo. Tampoco se han impuesto unas condiciones muy restrictivas en este sentido para manejar un número significativo de municipios en el estudio. En el siguiente listado se recogen todos los municipios excluidos:

- **UGR nº1 – Manc. de los Alcores**
  - Alcalá de Guadaíra
  - Dos Hermanas
  - Los Palacios y Villafranca
  - Sevilla
  - Utrera
- **UGR nº2 – Manc. del Guadalquivir**
  - Mairena del Aljarafe
- **UGR nº3 – Manc. de Servicios de la Vega**
  - La Rinconada
- **UGR nº6 – Manc. de Municipios de la Comarca de Écija**
  - Écija.

Toda la información presentada, incluyendo la exclusión de estos municipios tendría que quedar recogida en los estatutos de cada entidad, para que queden bien definidas sus competencias y las localidades que se benefician de este sistema.

### **3.3 Línea base para el desarrollo del Modelo**

Para el estudio que se plantea en este trabajo, no se respeta la distribución de los municipios entre las distintas plantas de destino que se hace en el Plan. Es decir, el Plan puede haber asignado una estación de transferencia para un municipio dado y según los criterios de la formulación matemática seguida en este estudio, es más eficiente que se lleve a otra. Se toma como referencia el reparto de los municipios entre las UGRs anteriormente presentado, con las exclusiones discutidas. Del mismo modo, en el siguiente listado se establecen las instalaciones que se han utilizado en el análisis, ya sean estaciones de transferencia o plantas de tratamiento.

#### **UGR nº1 – Mancomunidad de los Alcores**

- Planta de Tratamiento:
  - Montemarta-Cónica – Alcalá de Guadaira
- Estaciones de Transferencia:
  - ET de Lebrija - Lebrija

#### **UGR nº2 – Mancomunidad del Guadalquivir**

- Planta de Tratamiento:
  - Espartinas - Espartinas

#### **UGR nº3 – Mancomunidad de Servicios de la Vega**

- Planta de Tratamiento:
  - Centro de la Vega: Alcalá del Río
- Estación de Transferencia:
  - ET de Lora del Rio – Lora del Río

#### **UGR nº4 – Mancomunidad de Municipios Sierra Norte**

- Planta de Tratamiento:
  - Centro de la Vega – Alcalá del Río
- Estación de Transferencia:
  - ET de Constantina - Constantina

#### **UGR nº5 – Consorcio de Medio Ambiente Estepa-Sierra Sur**

- Planta de Tratamiento:
  - Mata Grande – Estepa
- Estación de Transferencia:
  - ET de El Saucejo – El Saucejo

#### **UGR nº6 – Mancomunidad de Municipios Comarca de Écija**

- Planta de Tratamiento:
  - Campiña 2000 - Marchena
- Estación de Transferencia:
  - ET de Écija - Écija

#### **UGR nº7 – Mancomunidad Intermunicipal Campiña 2000**

- Planta de Tratamiento:
  - Campiña 2000 - Marchena

Se vuelve a destacar que no entra dentro del alcance del proyecto el emplazamiento de las plantas de tratamiento y las estaciones de transferencia, se hace uso de las que propone el Plan y se toma su localización actual como dato. Dejando claro de este modo que el objetivo del modelo es minimizar la distancia recorrida por los camiones entre los municipios, atendiendo a la generación de residuos y a las distintas restricciones que se presentarán en el apartado correspondiente.

Para tener una visión esquemática de la estructura organizativa de la provincia y de la distribución de municipios sobre la que se va a aplicar el modelo matemático se presentan los gráficos de la Figura 3.6 y la Figura 3.7. Estos esquemas serán muy útiles para comprender durante la resolución del problema cómo se organizan los municipios y de qué instalaciones hacen uso en cada mancomunidad. Para evitar redundancia entre los distintos apartados, los mapas de cada UGR por separado especificando el emplazamiento de las instalaciones correspondientes se presentarán con las soluciones en el Capítulo 5.

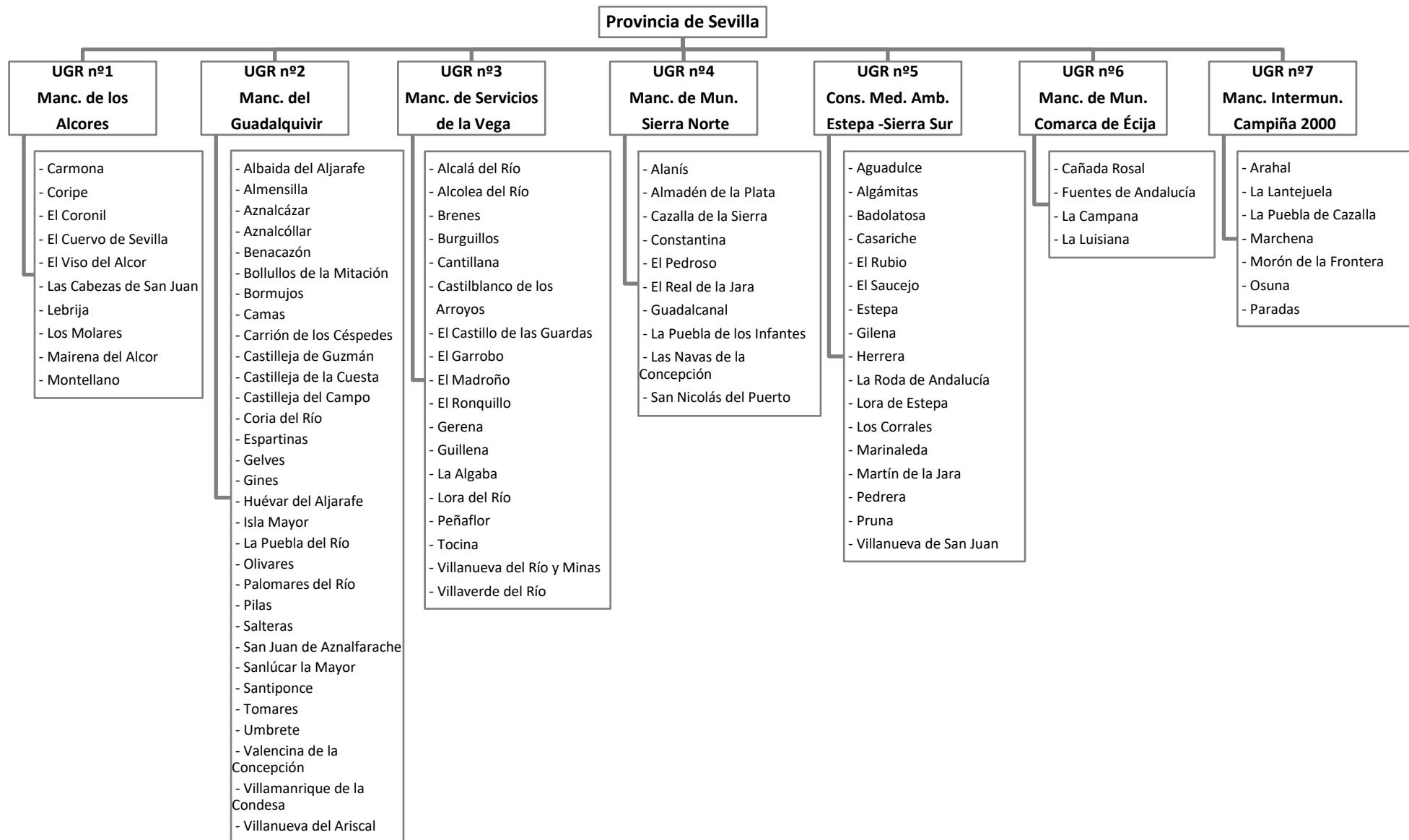


Figura 3.6. Distribución de municipios por UGRs. (Elaboración propia)

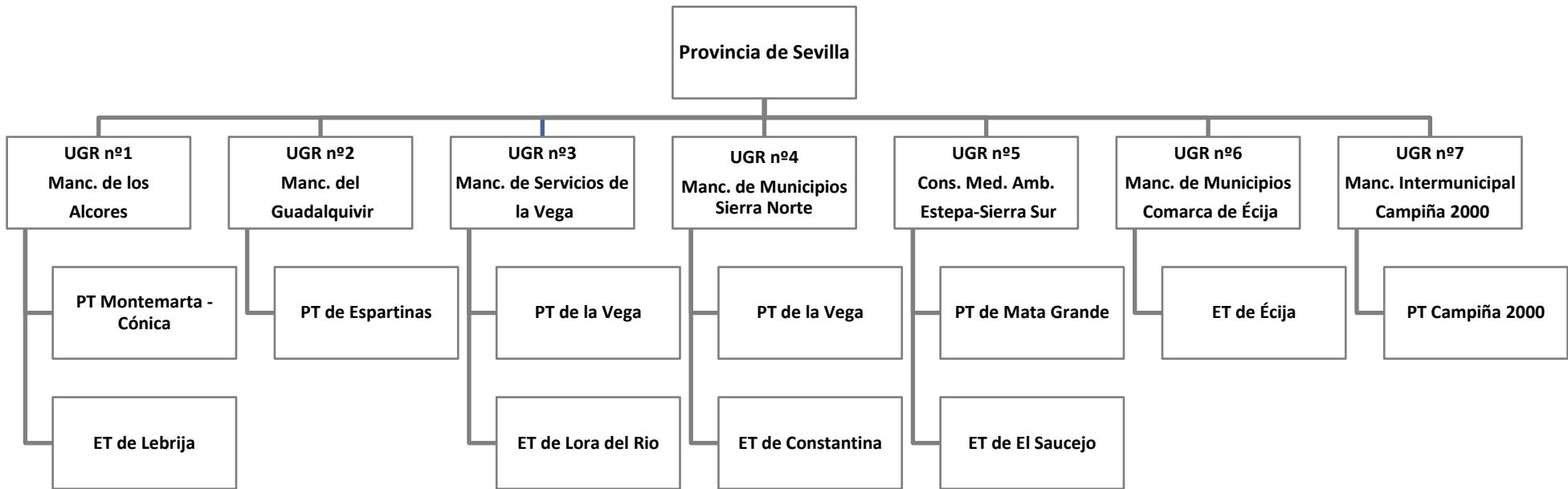


Figura 3.7. Distribución de instalaciones por UGRs. (Elaboración propia)



# 4 DESARROLLO DEL MODELO

---

Una vez conocido el caso objeto de estudio, es necesario realizar algunas hipótesis que terminan de caracterizar el problema. Estas simplificaciones se definen en el primer apartado de la presente sección. Una vez establecidos, ya se podría desarrollar el modelo matemático que se utilizará en la resolución del problema de la recogida de residuos urbanos en la Provincia de Sevilla. Además, antes de presentar el modelo definitivo que será implementado, se especifica la notación y la nomenclatura de los parámetros.

En el segundo apartado se traducen a restricciones matemáticas las características del problema. Partiendo de un caso simplificado se plantea una primera aproximación para posteriormente establecer cuáles serán los modelos definitivos aplicados a las mancomunidades. Se toman como referencia las definiciones clásicas del TSP y del VRP, cuya formulación se introdujo en la Sección 2, así como los casos similares que se han encontrado en la literatura.

## 4.1 Hipótesis simplificativas y suposiciones

Para caracterizar por completo el caso práctico, además de la información ofrecida en la Sección 3, es necesario plantear una serie de hipótesis y suposiciones que definirán finalmente el modelo matemático. En algunos casos se trata de datos que no se han podido encontrar debido a la falta de información al respecto, o bien porque no existen registros actualmente. A continuación, se establecen los parámetros que se van a fijar para cada UGR, las cifras exactas se especifican en la Sección 5.

- **Número de vehículos:** Puesto que la prestación conjunta del servicio de recogida de residuos sólidos urbanos solo se presta actualmente en la mancomunidad del Guadalquivir, para el resto de UGRs se asignará un tamaño de flota capaz de satisfacer la demanda de los municipios a los que prestan servicio. Esto será que la suma de las capacidades de todos los camiones es superior a la suma de la demanda de todos los clientes.
- **Flota homogénea:** Tal y como se ha comprobado en los trabajos de distintos autores, para no incrementar el peso computacional del modelo y facilitar la resolución del problema, se toma una flota homogénea, donde todos los vehículos tienen la misma capacidad (Baldacci et al., 2010; Dantzig & Ramser, 1959; Laporte, 2006). Tener una flota homogénea también conlleva que el modelo de recogida sea el mismo para todos los camiones, siendo la carga trasera la opción elegida. Finalmente, se supone que la flota es capaz de abarcar toda la demanda de la UGR a la que está asociada.
- **Localización de los depósitos:** En el caso de no haber encontrado información acerca de los depósitos de vehículos, se han tomado las sedes administrativas de las mancomunidades para este cometido.
- **Velocidades:** Se han establecido dos velocidades, una de tránsito intermunicipal y otra para los recorridos interiores en los municipios. Para la primera se han asignado 50 km/h y para la segunda 25 km/h.
- **Tiempos de operación:** El método de recogida que se ha tomado para todos los vehículos es el de carga trasera. Los tiempos que se manejan con estos camiones es de unos 50 segundos por contenedor recogido y de una media hora para descargar la basura en la planta de destino.
- **Distancia recorrida en el interior de los municipios:** No entra dentro de este trabajo optimizar el

recorrido interno de los municipios, existen herramientas muy potentes que permiten realizar estas tareas teniendo en cuenta factores poblacionales muy concretos, como son los Sistemas de Información Geográfica (SIG). Sin embargo, para contabilizar el tiempo empleado en el interior de las poblaciones se ha establecido una distancia que depende del perímetro del municipio y de un factor de concentración que se define más adelante en esta lista.

- **Número de contenedores:** De nuevo, se dispone de únicamente de los datos de la Mancomunidad del Guadalquivir. Se ha realizado una aproximación de la cantidad de contenedores en cada UGR partiendo del dato conocido, en base a la producción de residuos. Del mismo modo se han repartido los contenedores entre los municipios.
- **Factor de concentración:** Se trata de un coeficiente que va multiplicando al perímetro de cada municipio. Este parámetro se calcula en base a la concentración de contenedores en un municipio, cuanto menor es la ratio contenedores-km<sup>2</sup>, mayor es el coeficiente. De esta forma, se contabiliza una mayor distancia en los municipios donde los contenedores estén muy dispersos, que en aquellos donde los núcleos de población estén muy centralizados.
- **Distancias entre municipios:** Calculadas haciendo uso de las coordenadas geográficas de los municipios. Están definidas como líneas rectas que unen dos puntos, incluyendo una corrección asociada a la curvatura de la tierra gracias a una función de las herramientas de software utilizadas. Aunque es cierto que para las distancias que se van a manejar, esta corrección no influye mucho.

## 4.2 Primera aproximación

El primer paso en este apartado es relacionar el caso práctico presentado con situaciones similares en la literatura revisada. Para ello se va a definir una UGR ficticia muy simplificada con siete municipios, cada uno con su generación de residuos asociada, tres vehículos y un depósito que coincide con la planta de destino. La representación gráfica de este planteamiento puede verse en la Figura 4.1, en la cual los municipios se han representado por círculos azules y a cada vehículo se le ha asignado un color. A modo de recordatorio, la planta de destino podía ser una estación de transferencia o una planta de tratamiento.

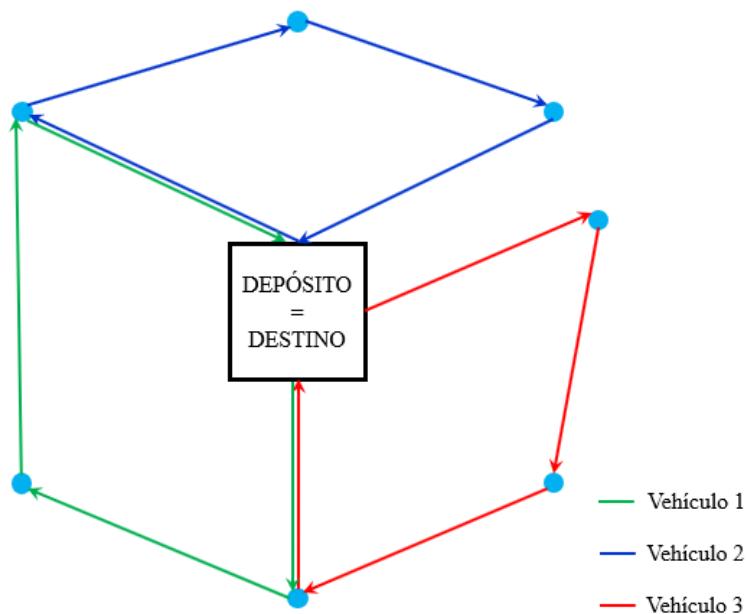


Figura 4.1. UGR ficticia. Simplificación del caso práctico. (Elaboración propia)

Como se puede apreciar en la representación, se generan tres rutas, cada una recorrida por un vehículo distinto. Esto hace que el problema sea fácilmente reconocible como un problema de enrutamiento de vehículos con capacidades, CVRP. Además, se observa que el problema puede aproximarse también como un mTSP que, como se vio anteriormente, es otro caso más del generalista VRP. Las formulaciones matemáticas del TSP y del VRP se presentaron en la Sección 2, siendo el modelo del VRP válido para resolver cualquiera de los problemas equivalentes mTSP o VRP. Una vez se han identificado los modelos que se pueden utilizar como punto de partida, es el momento de revisar qué consecuencias tiene en la formulación matemática lo expuesto

en la Presentación del Caso.

En la Tabla 4.1, se recopilan la caracterización extraída del análisis de la situación. En primer lugar, es necesario elegir qué será objeto del procedimiento de optimización. Como se ha mencionado anteriormente, se selecciona el coste para incluirlo en la función objetivo, relacionándolo directamente con la distancia recorrida por los vehículos, ya que se ha considerado que es el factor más influyente.

Continuando con los requerimientos, es necesario garantizar que se satisface la demanda, es decir que se recogen todos los residuos que se generan en cada municipio. Esta recogida está sujeta a las limitaciones impuestas por la capacidad de los vehículos, una vez que hayan alcanzado su capacidad máxima regresaran inmediatamente a la planta de destino que se le asigne. Por otra parte, se introduce un límite temporal asociado a la duración de una jornada laboral, como máximo los vehículos estarán circulando durante ocho horas. Esto no se trata como una ventana temporal o un intervalo de horas sino más bien como un límite superior. Por otro lado, habrá que definir cómo se introducen la distancia recorrida entre los municipios y dentro de estos en el modelo. Finalmente, todos los vehículos tienen que pasar por las estaciones de destino antes de regresar al depósito, nunca podrán volver a la base cargados de residuos.

Tipo	Caracterización el modelo
Objetivo	Minimizar coste (distancia)
Requerimientos	Continuidad de flujo Descarga en plantas de destino Recogida de todos los residuos Capacidad máxima de los vehículos Distancias intermunicipales Distancias intramunicipales Limitaciones de tiempo

Tabla 4.1. Caracterización del modelo

Tras caracterizar las especificaciones del caso práctico y a expensas de plasmar las restricciones en expresiones matemáticas, se puede dar una definición formal del problema en cuestión. Dada una UGR definida sobre un grafo  $G = (V, A)$  donde  $V = \{0, 1, \dots, n\}$  representa todos los nodos del problema y el conjunto  $A = \{(i, j) : i, j \in V, i \neq j\}$  define los arcos que se forman entre los nodos. El nodo 0 se reserva para la base de los vehículos y los clientes, en este caso municipios, están comprendidos entre  $i \in \{1, \dots, n\}$ . Existen  $m$  vehículos disponibles en la base, todos con la misma capacidad  $Q$ . Cada municipio tiene asociada una generación de residuos  $q_i$  entendida como la demanda de los clientes; mientras que las distancias están definidas entre los nodos y viene dada por la matriz simétrica de distancias  $c_{ij}$ , por lo que se cumple la condición de simetría  $c_{ij} = c_{ji}$ .

En los siguientes subapartados, se presentan los componentes de la formulación matemática del modelo empezando por definir la notación que se va a utilizar en las expresiones. En segundo lugar, se desarrolla la función objetivo con las consideraciones oportunas. Despues, se muestran los requerimientos divididos en cinco grupos según la característica o el parámetro al que se refieran: flujo de vehículos, demanda, capacidad, distancia y tiempo. Esta distribución puede llegar a ser algo difusa para algunos requerimientos puesto que las restricciones que se derivan de ellos afectan a elementos de varios grupos.

#### 4.2.1 Notación matemática

Se presenta la notación matemática seleccionada para definir las expresiones ofrecidas en esta sección. A menos que específicamente se indique lo contrario, esto será de aplicabilidad en lo que resta de trabajo a fin de facilitar la compresión del modelo.

$n:$	Número de municipios
$e:$	Número de plantas de destino
$m:$	Número de vehículos
$i, j:$	Índices asociados a los nodos del problema: depósito, municipios y plantas de destino
$k:$	Índice asociado a los vehículos
$c_{ij}:$	Matriz de distancias entre los nodos
$x_{ijk}:$	Variable binaria. Vale 1 cuando el arco $(i, j)$ es recorrido por el vehículo $k$ , y 0 en caso contrario
$p_j:$	Perímetro del municipio $j$
$f_j:$	Factor de concentración del municipio $j$
$u_i$	Variable auxiliar
$z_{ijk}$	Variable de linealización
$y_{jk}$	Cantidad de basura recogida en el municipio $j$ , por el vehículo $k$
$q_i :$	Producción de residuos del municipio $i$
$U:$	Límite superior de la producción de residuos
$c_{veh_k}:$	Carga del vehículo al final de la ruta
$d_{veh_k}:$	Distancia recorrida por el vehículo $k$ al final de la ruta
$d_{ext_k}:$	Distancia recorrida por el vehículo $k$ entre los municipios incluidos en la ruta
$d_{int_k}:$	Distancia recorrida por el vehículo $k$ en el interior de los municipios visitados en la ruta
$t_{ext_k}:$	Tiempo empleado por el vehículo $k$ entre los municipios incluidos en la ruta
$v_c:$	Velocidad en carretera
$t_{int_k}:$	Tiempo empleado por el vehículo $k$ en el interior de los municipios visitados en la ruta
$v_p:$	Velocidad en población
$t_{con_k}:$	Tiempo empleado por el vehículo $k$ en cargar los contenedores de los municipios visitados en la ruta
$n_c:$	Número de contenedores de la UGR
$q_t:$	Cantidad total de residuos generada

$t_r$ :	Tiempo de recogida por contenedor
$t_{vac_k}$ :	Tiempo empleado por el vehículo $k$ en vaciar la carga en la planta de destino
$t_v$ :	Tiempo de descarga de los vehículos
$t_{veh_k}$ :	Tiempo total empleado por el vehículo $k$ al final de la ruta
$e_1, e_2$	Plantas de destino
$Q$ :	Capacidad máxima de los vehículos

## 4.2.2 Función Objetivo

La función objetivo no sufre ningún cambio significativo respecto a la formulación original del VRP según la formulación del flujo de vehículos con tres índices. Puesto que en dicha expresión ya se consideraba el número de vehículos que interviene, a diferencia de lo que ocurría con el TSP.

$$\min \sum_{\substack{i=0 \\ i \neq j}}^n \sum_{j=0}^n c_{ij} \sum_{k=1}^m x_{ijk} \quad (4.1)$$

De hecho, simplemente quedaría identificar los términos de la expresión (4.1) con las características particulares del caso práctico. El coste  $c_{ij}$  en este caso se corresponde con la distancia recorrida por los vehículos, mientras que la variable binaria  $x_{ijk}$  toma el valor de la unidad cuando el arco comprendido entre el municipio  $i$  y el  $j$  es recorrido por el vehículo  $k$ , y cero en el caso contrario. La diferencia principal con la formulación presentada anteriormente reside en la asignación del índice 0 para el depósito.

## 4.2.3 Gestión del flujo

En este grupo se incluye todo lo relacionado con la organización del flujo de vehículos entre los nodos del problema. Es decir, aquellas restricciones que de alguna manera “dirigen” a los camiones en su recorrido. Se excluyen las restricciones que tienen en cuenta la producción de residuos, que se abordaran en el siguiente grupo. Para este conjunto de restricciones además de basarnos en los modelos discutidos en el Estado del Arte, se ha seguido el planteamiento expuesto en (Salvador, 2012).

Partiendo de un municipio cualquiera del problema, se va a discutir las implicaciones de que sea visitado en una ruta. Es propiedad del TSP que por cada municipio se pasa una y solo una vez, esta afirmación se relaja para el mTSP y para el VRP, incluyendo al camión en la ecuación. De manera que formulando de nuevo esa característica, se podría decir que, por cada municipio para un mismo vehículo, se pasa una y solo una vez. Dejando de este modo vía libre a otros vehículos para recoger la basura en caso de no haberlo hecho en una sola visita.

En el planteamiento clásico de los problemas de enrutamiento de vehículos se establece que desde un municipio  $i$ , se pueda ir a un único municipio  $j$ , para un determinado vehículo  $k$ . Y del mismo modo se plantea el complementario, a un municipio  $j$  solo se puede llegar desde un único municipio  $i$ , dado un vehículo  $k$ . Esto se traduce en las expresiones matemáticas (4.2) y (4.3), donde se ha sustituido la igualdad original de la formulación del TSP por el menor o igual. De esta manera no se obliga a todos los vehículos a pasar por todos los municipios, la obligación de acudir a los municipios vendrá impuesta por las restricciones que se plantean en el subapartado 4.2.4.

$$\sum_{j=0}^n x_{ijk} \leq 1 \quad \forall i \in \{0, \dots, n\}, i \neq j, \forall k \in \{1, \dots, m\} \quad (4.2)$$

$$\sum_{i=0}^n x_{ijk} \leq 1 \quad \forall j \in \{0, \dots, n\}, i \neq j, \forall k \in \{1, \dots, m\} \quad (4.3)$$

Tal y como planteaba la formulación del TSP revisada en la Sección 2, son necesarias las restricciones de eliminación de ciclos internos. Se respeta la formulación MTZ presentada en el capítulo mencionado, introduciendo la modificación correspondiente de la variable binaria, provocada por la inclusión de los vehículos mediante el índice  $k$ .

$$u_i - u_j + n x_{ijk} \leq n - 1 \quad \forall i, j \in \{1, \dots, n\}, i \neq j, \forall k \in \{1, \dots, m\} \quad (4.4)$$

Es necesario garantizar la continuidad del flujo, es decir, no puedo salir de un municipio si no he llegado hasta este previamente. La restricción (4.5) garantiza esta continuidad asegurando que los tramos dentro de una misma ruta son consecutivos. Para ello, se toma un municipio  $s$  y para un vehículo  $k$  determinado, se garantiza que si hay un arco  $(i, s)$  existe un único  $(s, j)$ .

$$\sum_{\substack{i=0 \\ i \neq s}}^n x_{isk} = \sum_{\substack{j=0 \\ j \neq s}}^n x_{sjk} \quad \forall s \in \{0, \dots, n\}, \forall k \in \{1, \dots, m\} \quad (4.5)$$

Otro de los requerimientos derivados de la caracterización del problema es el regreso al depósito, descargando previamente los residuos recogidos en la planta de destino. Sin embargo, en la simplificación del problema enunciada al comienzo de esta sección se estableció que la base y la planta coincidían. De manera que únicamente habría que garantizar la vuelta al depósito, lo cual se impone mediante la restricción (4.6). Todo vehículo  $k$  que salga de la base recorriendo el arco genérico  $(0, j)$  tiene que volver a ella  $(i, 0)$ .

$$\sum_{i=1}^n x_{0jk} = \sum_{j=1}^n x_{i0k} \quad \forall k \in \{1, \dots, m\} \quad (4.6)$$

#### 4.2.4 Gestión de la demanda

Cada municipio genera una cantidad de residuos sólidos urbanos conocida a priori. Como en todo problema de enrutamiento de vehículos, existe una demanda asociada a los clientes (municipios), en este caso el servicio demandado es la recogida de basura. Por tanto, es coherente identificar la generación de residuos con la demanda del problema, siendo la recogida total de estos residuos la condición que rige este conjunto de restricciones. El planteamiento de la gestión de la demanda que se formula a continuación deriva de los trabajos planteados en (Salvador, 2012) y (Baldacci et al., 2010).

Puesto que un vehículo  $k$  puede venir de recoger de otro municipio, es necesario conocer la cantidad que recoge en cada nodo  $j$ . Con dicho cometido se define una variable adicional  $y_{jk}$  que almacena esta información. Haciendo uso de esta nueva variable, la restricción (4.7) asegura la recogida de todos los residuos generados en todos los municipios.

$$\sum_{i=0}^n \sum_{k=1}^m x_{ijk} y_{jk} = q_j \quad \forall j \in \{1, \dots, n\}, i \neq j \quad (4.7)$$

El problema que presenta esta formulación de la gestión de la demanda es la multiplicación de una variable entera por una binaria, eliminando la linealidad del sistema. Para solventar este problema, se descompone la anterior (4.7) en un conjunto de cuatro restricciones dependientes de una nueva variable  $z_{ijk}$ .

$$\sum_{i=0}^n \sum_{k=1}^m z_{ijk} = q_j \quad \forall j \in \{1, \dots, n\}, i \neq j \quad (4.8)$$

$$z_{ijk} \leq U x_{ijk} \quad \begin{aligned} & \forall i \in \{0, \dots, n\}, \forall j \in \{1, \dots, n\}, \\ & \forall k \in \{1, \dots, m\} \end{aligned} \quad (4.9)$$

$$\sum_{i=0}^n z_{ijk} \leq y_{jk} \quad \forall j \in \{1, \dots, n\}, i \neq j, \forall k \in \{1, \dots, m\} \quad (4.10)$$

$$\sum_{i=0}^n z_{ijk} \geq y_{jk} - U \left( 1 - \sum_{i=0}^n x_{ijk} \right) \quad \forall j \in \{1, \dots, n\}, \forall k \in \{1, \dots, m\} \quad (4.11)$$

Además de las variables nuevas comentadas, en estas ecuaciones aparece también una cota superior de la demanda  $U$ . Lo que se consigue con este planteamiento es relacionar el movimiento de los camiones representado mediante la variable  $x_{ijk}$  con la recogida reflejada en las variables  $y_{jk}$  y  $z_{ijk}$ . El razonamiento seguido consiste en limitar la variable  $z_{ijk}$  inferior y superiormente por  $y_{jk}$  de manera que queden ambas determinadas, siendo equivalentes en planteamiento, pero con distintos índices. Este conjunto de restricciones se completa con la expresión (4.12) que tiene que cumplirse en última instancia.

$$\sum_{k=1}^m y_{jk} = q_j \quad \forall j \in \{1, \dots, n\} \quad (4.12)$$

Con el objetivo de disminuir la distancia recorrida es necesario garantizar que en todo municipio visitado se recoge basura, evitando de este modo viajes en vano. Este hecho está relacionado con la gestión del flujo, pero queda principalmente determinado por la gestión de la demanda a raíz de imponer la restricción (4.13), que permite esquivar situaciones donde la basura recogida sea nula ( $y_{jk} = 0$ ).

$$\sum_{i=0}^n x_{ijk} \leq y_{jk} \quad \forall j \in \{1, \dots, n\}, \forall k \in \{1, \dots, m\} \quad (4.13)$$

#### 4.2.5 Capacidad

Se exponen ahora las restricciones asociadas a la capacidad de los vehículos, la cual se apoya en la hipótesis de una flota homogénea planteada al comienzo de esta sección. En esta flota todos los vehículos tendrían una misma capacidad de valor  $Q$ . Para implementar esta limitación, se impone un límite superior  $Q$  en la declaración de la variable auxiliar  $c_{veh_k}$ , de esta manera se ahorra el coste computacional generado al incorporar una restricción adicional.

Esta variable auxiliar  $c_{veh_k}$  almacena las cantidades que recoge el vehículo  $k$ . Al concluir el proceso de optimización, el valor de esta se corresponderá con la cantidad total recogida (en kg) por dicho vehículo en todas las poblaciones que haya visitado. Su expresión matemática es muy sencilla y viene dada por:

$$c_{veh_k} = \sum_{j=1}^n y_{jk} \quad \forall k \in \{1, \dots, m\} \quad (4.14)$$

#### 4.2.6 Distancia

Las formulaciones que aquí se presentan influyen directamente sobre la función objetivo. En la expresión (4.1),  $c_{ij}$  se refiere a la distancia que existe entre el municipio  $i$  y el  $j$ . Sin embargo, tal y como se comentó en el apartado “Hipótesis simplificativas y suposiciones”, se considera también la distancia recorrida en el interior de las poblaciones. Por este motivo, la distancia total recorrida por un vehículo  $d_{veh_k}$  se puede descomponer en dos variables auxiliares  $d_{int_k}$  y  $d_{ext_k}$ , según la expresión (4.15).

$$d_{veh_k} = d_{ext_k} + d_{int_k} \quad \forall k \in \{1, \dots, m\} \quad (4.15)$$

$$d_{ext_k} = \sum_{i=0}^n \sum_{\substack{j=0 \\ i \neq j}}^n c_{ij} x_{ijk} \quad \forall k \in \{1, \dots, m\} \quad (4.16)$$

$$d_{int_k} = \sum_{i=0}^n \sum_{\substack{j=0 \\ i \neq j}}^n x_{ijk} p_j f_j \quad \forall k \in \{1, \dots, m\} \quad (4.17)$$

La expresión (4.16) que define a la variable auxiliar  $d_{ext_k}$ , es equivalente a la función objetivo original (4.1) y representa la distancia que tiene que cubrir los vehículos entre los municipios. Por otro lado,  $d_{int_k}$  referida al recorrido internos por los municipios visitados queda dada por (4.17). En esa ecuación aparecen dos parámetros además de la variable binaria. El primero de ellos  $p_j$ , referido al perímetro del municipio  $j$ , mientras que el segundo es el factor de concentración  $f_j$  descrito en el apartado 4.1. Este factor toma en valor de 1, 1.5 o 2 en función de la concentración de contenedores del municipio.

Esta reinterpretación modifica la formulación propuesta de la función objetivo, puesto que habría que considerar la distancia global y no únicamente la intermunicipal. La función objetivo presentada en (4.1), pasaría entonces a estar representada por (4.18), que si se desarrolla introduciendo los términos de las variables auxiliares  $d_{ext_k}$  y  $d_{int_k}$ , según su expresiones (4.16) y (4.17) respectivamente, quedaría como se muestra en (4.19).

$$\min \sum_{k=1}^m d_{veh_k} \quad (4.18)$$

$$\min \sum_{i=0}^n \sum_{j=0}^n \sum_{\substack{k=1 \\ i \neq j}}^m (c_{ij} + p_j f_j) x_{ijk} \quad (4.19)$$

#### 4.2.7 Tiempo

Analizando el desarrollo matemático presentado hasta ahora, se observa que el tiempo no está incluido en la función objetivo y no tiene ninguna restricción específica, salvo la ya comentada de la duración de una jornada laboral. Sin embargo, como en casi cualquier aplicación ingenieril es interesante conocer cuánto tiempo se emplea en realizar una determinada operación. Para ello, se utilizarán una serie de variables auxiliares que recogerán el tiempo empleado en cada tramo del recorrido. Las expresiones matemáticas que las definen se muestran a continuación:

$$t_{ext_k} = \frac{d_{ext_k}}{v_c} \quad \forall k \in \{1, \dots, m\} \quad (4.20)$$

$$t_{int_k} = \frac{d_{int_k}}{v_p} \quad \forall k \in \{1, \dots, m\} \quad (4.21)$$

$$t_{con_k} = \sum_{j=1}^n y_{jk} \frac{n_c}{q_t} t_r \quad \forall k \in \{1, \dots, m\} \quad (4.22)$$

$$t_{vac_k} = \sum_{i=0}^n x_{i0k} t_v \quad \forall k \in \{1, \dots, m\} \quad (4.23)$$

Es en estas expresiones donde se concentra el uso de la mayor parte de los parámetros introducidos en el modelo. Revisando (4.20) y (4.21), es inmediata la identificación de la relación con las distancias recorridas dentro y fuera de los municipios. La variable  $t_{ext_k}$  afectada por el parámetro  $v_c$  que representa la velocidad en carretera y que está establecida en 50 km/h, mientras que  $t_{int_k}$  a través del parámetro  $v_p$  fija la velocidad dentro de la población en 25 km/h.

Pasando a la igualdad establecida en (4.22), mediante la variable auxiliar  $t_{con_k}$  se contabiliza el tiempo que se tarda en recoger los contenedores. Para conocer cuántos contenedores se recogen, se hace uso del número de contenedores total de la mancomunidad ( $n_c$ ) y se pondera con la cantidad recogida  $y_{jk}$  por el vehículo  $k$ , dividido entre la cantidad total de residuos producida ( $q_t$ ). El valor obtenido de estas operaciones se multiplica por  $t_r$  que es el tiempo que tarda en recogerse un contenedor mediante el sistema de carga trasera, establecido en 50 segundos.

Por último, la expresión (4.23) registra el tiempo que tarda en descargar el camión una vez llega a la planta de destino (base en el caso simplificado). Todo camión que recorra un arco del tipo  $(i, 0)$  es obligado a descargar. El tiempo de esta operación es añadido a través del parámetro  $t_v$  que propone un tiempo de vaciado de media hora, contabilizando desde la entrada del camión en la zona o planta de descarga hasta que el camión sale de esta.

La suma de todas estas variables conforma el tiempo total empleado por un vehículo dado  $k$  y su valor se almacena en la variable auxiliar  $t_{veh_k}$  según la expresión (4.24). En la declaración de esta variable se establece un límite superior igual a las ocho horas de duración de la jornada laboral, evitando de esta manera crear más restricciones.

$$t_{veh_k} = t_{ext_k} + t_{int_k} + t_{con_k} + t_{vac_k} \quad \forall k \in \{1, \dots, m\} \quad (4.24)$$

## 4.3 Consolidación del Modelo

Partiendo de la formulación del apartado anterior, se hace extensivo el modelo matemático allí propuesto a las UGR reales que conforman la Provincia de Sevilla y son objeto del estudio. Se puede observar en los esquemas presentados en la Sección 3 (Figura 3.7) que las mancomunidades hacen uso de una o dos plantas de destino. Según las suposiciones realizadas para la UGR ficticia propuesta en el apartado anterior, el depósito y la planta de destino coincidían. Sin embargo, en el caso real esto no es así para la mayoría de las mancomunidades. De manera que serán necesarias dos adaptaciones del modelo matemático: una para el caso en que existe únicamente una planta de destino y otra para aquellas que tengan dos.

Con estas consideraciones, la definición del problema difiere sutilmente de la presentada anteriormente. Afectando principalmente al conjunto de los nodos para tener en cuenta los correspondiente a las plantas de destino. Se añadirán por tanto  $e$  plantas de destino, que según la UGR que sea tendrá una ( $e_1$ ) o dos ( $e_1, e_2$ ) estaciones. De manera que la definición se enunciaría como sigue, dada una UGR definida sobre un grafo  $G = (V, A)$  donde  $V = \{0, 1, \dots, n + e\}$  representa todos los nodos del problema y el conjunto de arcos que se forman entre los nodos dado por  $A = \{(i, j) : i, j \in V, i \neq j\}$ . El nodo 0 se reserva para la base de los vehículos, las plantas de destino se corresponden con los nodos  $i \in \{n + 1, \dots, e\}$  y los clientes, en este caso municipios, están comprendidos entre  $i \in \{1, \dots, n\}$ . Existen  $m$  vehículos disponibles en la base, todos con la misma capacidad  $Q$ . Cada municipio tiene asociada una generación de residuos  $q_i$  entendida como la demanda de los clientes; mientras que las distancias están definidas entre los nodos y viene dada por la matriz simétrica de distancias  $c_{ij}$ , por lo que se cumple la condición de simetría  $c_{ij} = c_{ji}$ .

### 4.3.1 Modelo matemático para una planta de destino

En la Figura 4.2 se ilustra un caso genérico para una UGR con una única planta de destino. Sin pérdida de generalidad, se considera que la planta de destino tiene una ubicación diferente al depósito donde se almacenan los vehículos. De esta forma, en el caso de que la planta y el depósito coincidieran realmente, la distancia sería 0 y los términos correspondientes a las restricciones asociadas quedarían anulados. Este modelo es aplicable a las siguientes UGRs:

- Mancomunidad del Guadalquivir
- Mancomunidad de Municipios Comarca de Écija
- Mancomunidad Intermunicipal Campiña 2000

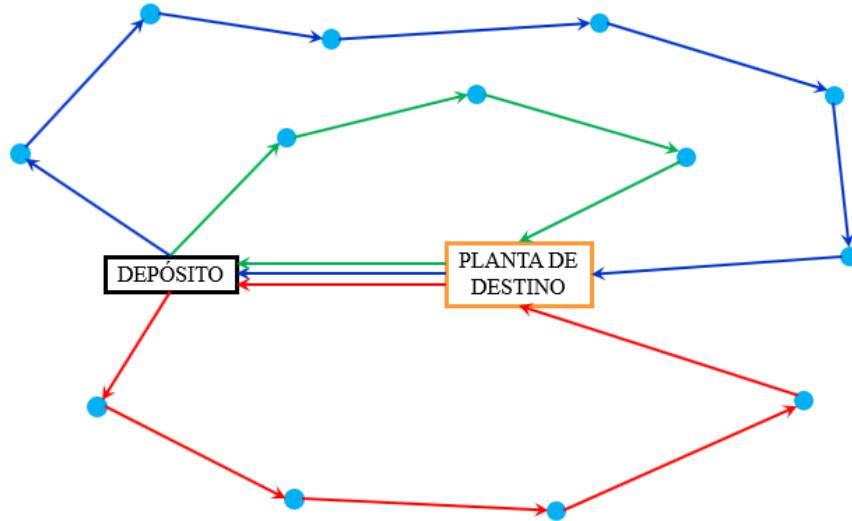


Figura 4.2. Modelo UGR con una planta de destino. (Elaboración propia)

Según las últimas modificaciones introducidas en la definición del problema, es necesario constituir las bases que regirán la gestión en la planta de destino. Con ese objetivo se añaden una serie de nuevas restricciones, las cuales pertenecen principalmente al grupo de gestión de flujo.

Por definición del problema, todo vehículo que salga del depósito recogerá basura, lo que obliga a todo vehículo a descargar en la planta de destino. Esta afirmación se puede reducir a que todo vehículo que sale de la base se dirige a la planta de destino, lo que se plasma en la restricción (4.25). Una vez el vehículo ha descargado, tiene que volver a la base, restricción impuesta según las expresiones (4.26) y (4.27). La primera que establece que todo camión que recorra el arco  $(i, e_1)$  desde cualquier municipio  $i$ , tiene que recorrer el arco  $(e_1, 0)$ ; mientras que la segunda es la garantía de que se va desde la planta al depósito una vez por vehículo.

$$\sum_{i=1}^n x_{ie_1k} = \sum_{j=1}^n x_{0jk} \quad \forall k \in \{1, \dots, m\} \quad (4.25)$$

$$\sum_{i=1}^n x_{ie_1k} = x_{e_10k} \quad \forall k \in \{1, \dots, m\} \quad (4.26)$$

$$x_{e_10k} = 1 \quad \forall k \in \{1, \dots, m\} \quad (4.27)$$

En línea con las restricciones anteriores se encuentra la (4.28), que limita el flujo entre la planta de destino y los municipios, según los arcos del tipo  $(e_1, j)$ . Desde la planta se puede ir solo a la base. Además, desde la base no se puede ir directamente a la estación, es decir el valor de la variable binaria para el arco  $(0, e_1)$  tiene que ser nulo, condición impuesta en (4.29).

$$\sum_{j=1}^n x_{e_1jk} = 0 \quad \forall k \in \{1, \dots, m\} \quad (4.28)$$

$$x_{0e_1k} = 0 \quad \forall k \in \{1, \dots, m\} \quad (4.29)$$

Para concluir con las expresiones asociadas a la gestión del flujo, en (4.30) se garantiza que todo vehículo en circulación debe tener un arco del tipo  $(i, e_1)$  que lo dirija en algún momento hacia la planta de destino.

$$\sum_{i=1}^n x_{ie_1k} \geq 1 \quad \forall k \in \{1, \dots, m\} \quad (4.30)$$

Por último, la expresión correspondiente a la variable auxiliar  $t\_vac_k$  (4.23) se ve afectada puesto que la descarga se realiza en la planta de destino en vez de en la base de los vehículos. Reemplazando los índices de la base por los de la planta, la nueva igualdad se representa en (4.32).

$$t\_vac_k = \sum_{i=0}^n x_{ie_1k} t_v \quad \forall k \in \{1, \dots, m\} \quad (4.31)$$

Tras haber detallado estas nuevas restricciones y a modo de resumen, se presenta un modelo completo que se aplicará a las mancomunidades del Guadalquivir, Écija y Campiña 2000. En este modelo definitivo se adapta el tratamiento de los índices utilizado en las restricciones discutidas en el apartado 4.2 para incorporar la planta de destino.

$$\min \sum_{i=0}^{n+e} \sum_{j=0}^{n+e} \sum_{\substack{k=1 \\ i \neq j}}^m (c_{ij} + p_j f_j) x_{ijk} \quad (4.32)$$

$$\sum_{j=0}^{n+e} x_{ijk} \leq 1 \quad \forall i \in \{0, \dots, n+e\}, i \neq j, \quad \forall k \in \{1, \dots, m\}$$

$$\sum_{i=0}^{n+e} x_{ijk} \leq 1 \quad \forall j \in \{0, \dots, n+e\}, i \neq j, \quad \forall k \in \{1, \dots, m\}$$

$$u_i - u_j + n x_{ijk} \leq n - 1 \quad \forall i, j \in \{1, \dots, n\}, i \neq j, \quad \forall k \in \{1, \dots, m\} \quad (4.35)$$

$$\sum_{\substack{i=0 \\ i \neq s}}^n x_{isk} = \sum_{\substack{j=1 \\ j \neq s}}^{n+e} x_{sjk} \quad \forall s \in \{1, \dots, n\}, \forall k \in \{1, \dots, m\} \quad (4.36)$$

$$\sum_{i=0}^n \sum_{k=1}^m z_{ijk} = q_j \quad \forall j \in \{1, \dots, n\}, i \neq j \quad (4.37)$$

$$z_{ijk} \leq U x_{ijk} \quad \forall i \in \{0, \dots, n\}, \forall j \in \{1, \dots, n\}, \forall k \in \{1, \dots, m\} \quad (4.38)$$

$$\sum_{i=0}^n z_{ijk} \leq y_{jk} \quad \forall j \in \{1, \dots, n\}, i \neq j, \forall k \in \{1, \dots, m\} \quad (4.39)$$

$$\sum_{i=0}^n z_{ijk} \geq y_{jk} - U \left( 1 - \sum_{i=0}^n x_{ijk} \right) \quad \forall j \in \{1, \dots, n\}, \forall k \in \{1, \dots, m\} \quad (4.40)$$

$$\sum_{k=1}^m y_{jk} = q_j \quad \forall j \in \{1, \dots, n\} \quad (4.41)$$

$$\sum_{i=0}^n x_{ijk} \leq y_{jk} \quad \forall j \in \{1, \dots, n\}, \forall k \in \{1, \dots, m\} \quad (4.42)$$

$$c\_veh_k = \sum_{j=1}^n y_{jk} \quad \forall k \in \{1, \dots, m\} \quad (4.43)$$

$$d\_ext_k = \sum_{i=0}^{n+e} \sum_{\substack{j=0 \\ i \neq j}}^{n+e} c_{ij} x_{ijk} \quad \forall k \in \{1, \dots, m\} \quad (4.44)$$

$$d\_int_k = \sum_{i=0}^n \sum_{\substack{j=0 \\ i \neq j}}^n x_{ijk} p_j f_j \quad \forall k \in \{1, \dots, m\} \quad (4.45)$$

$$t\_ext_k = \frac{d\_ext_k}{v_c} \quad \forall k \in \{1, \dots, m\} \quad (4.46)$$

$$t\_int_k = \frac{d\_int_k}{v_p} \quad \forall k \in \{1, \dots, m\} \quad (4.47)$$

$$t\_con_k = \sum_{j=1}^n y_{jk} \frac{n_c}{q_t} t_r \quad \forall k \in \{1, \dots, m\} \quad (4.48)$$

$$t\_vac_k = \sum_{i=0}^n x_{ie_1k} t_v \quad \forall k \in \{1, \dots, m\} \quad (4.49)$$

$$t\_veh_k = t\_ext_k + t\_int_k + t\_con_k + t\_vac_k \quad \forall k \in \{1, \dots, m\} \quad (4.50)$$

$$\sum_{i=1}^n x_{ie_1k} = \sum_{j=1}^n x_{0jk} \quad \forall k \in \{1, \dots, m\} \quad (4.51)$$

$$\sum_{i=1}^n x_{ie_1k} = x_{e_10k} \quad \forall k \in \{1, \dots, m\} \quad (4.52)$$

$$x_{e_10k} = 1 \quad \forall k \in \{1, \dots, m\} \quad (4.53)$$

$$\sum_{j=1}^n x_{e_1jk} = 0 \quad \forall k \in \{1, \dots, m\} \quad (4.54)$$

$$x_{0e_1k} = 0 \quad \forall k \in \{1, \dots, m\} \quad (4.55)$$

$$\sum_{i=1}^n x_{ie_1k} \geq 1 \quad \forall k \in \{1, \dots, m\} \quad (4.56)$$

$$q_i \leq u_i \leq Q \quad \forall i \in \{1, \dots, n\}$$

$$x_{ijk} \in \{0,1\} \quad \forall (i,j,k)$$

$$z_{ijk}, y_{jk}, c\_veh_k, d\_ext_k, d\_int_k, t\_ext_k, t\_int_k, t\_con_k, t\_vac_k, t\_veh_k \in \mathbb{R}$$

### 4.3.2 Modelo matemático para dos plantas de destino

Las modificaciones necesarias para modelar la casuística representada en la Figura 4.3 son similares a las planteadas para una única planta de destino. Las diferencias entre ambos casos están asociadas a las restricciones de gestión del flujo que abordan los posibles conflictos que se pueden producir entre las dos plantas de destino. Las UGRs que se estudian bajo este modelo son:

- Mancomunidad de los Alcores
- Mancomunidad de Servicios de la Vega
- Mancomunidad de Municipios Sierra Norte
- Consorcio Medio Ambiente Estepa-Sierra Sur

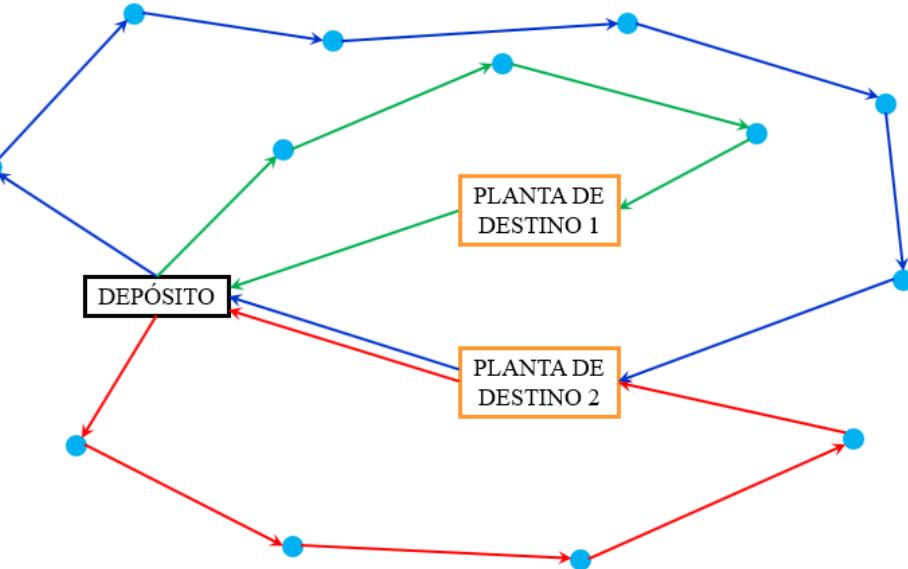


Figura 4.3. Modelo UGR con dos plantas de destino. (Elaboración propia)

Los cambios por introducir en la formulación de las restricciones expuestas en el apartado anterior son básicamente adaptaciones que amplían su aplicabilidad, incluyendo la estación adicional. En esta ocasión, en la primera planta viene dada por  $e_1$  y la segunda por  $e_2$ . A continuación, reescribimos las expresiones anteriores para dos plantas de destino.

$$\sum_{i=1}^n (x_{ie_1k} + x_{ie_2k}) = \sum_{j=1}^n x_{0jk} \quad \forall k \in \{1, \dots, m\} \quad (4.57)$$

$$\sum_{i=1}^n x_{iek} = x_{e0k} \quad \forall k \in \{1, \dots, m\}, \forall e \in \{e_1, e_2\} \quad (4.58)$$

$$x_{e_1 0k} + x_{e_2 0k} = 1 \quad \forall k \in \{1, \dots, m\} \quad (4.59)$$

$$\sum_{j=1}^n x_{ejk} = 0 \quad \forall k \in \{1, \dots, m\}, \forall e \in \{e_1, e_2\} \quad (4.60)$$

$$x_{0ek} = 0 \quad \forall k \in \{1, \dots, m\}, \forall e \in \{e_1, e_2\} \quad (4.61)$$

$$\sum_{i=1}^n x_{iek} \geq 1 \quad \forall k \in \{1, \dots, m\}, \forall e \in \{e_1, e_2\} \quad (4.62)$$

$$t\_vac_k = \sum_{i=0}^n (x_{ie_1 k} + x_{ie_2 k}) t_v \quad \forall k \in \{1, \dots, m\} \quad (4.63)$$

Como se puede comprobar, las restricciones (4.58), (4.60), (4.61) y (4.62) simplemente han extendido su aplicabilidad a la incorporación de la nueva planta de destino tal y como se mencionaba anteriormente. Por otro lado, las restricciones (4.57) y (4.59) introducen un nuevo término referido a esta nueva planta. En concreto en la expresión (4.57) se establece que todo vehículo que salga del depósito tiene que pasar por alguna de las dos plantas; mientras que (4.59) garantiza que cualquier vehículo  $k$  vaya a una u otra planta. La igualdad (4.63) referida al tiempo de vaciado suma también el componente de la nueva planta.

Finalmente, para dejar claro que modelo se aplica en las mancomunidades de los Alcores, la Vega y Sierra Norte, y en el consorcio Estepa-Sierra Sur, se recopilan todas las restricciones juntas en un modelo definitivo para dos plantas de destino.

$$\min \sum_{i=0}^{n+e} \sum_{j=0}^{n+e} \sum_{\substack{k=1 \\ i \neq j}}^m (c_{ij} + p_j f_j) x_{ijk} \quad (4.64)$$

$$\sum_{j=0}^{n+e} x_{ijk} \leq 1 \quad \forall i \in \{0, \dots, n+e\}, i \neq j, \quad (4.65)$$

$$\forall k \in \{1, \dots, m\}$$

$$\sum_{i=0}^{n+e} x_{ijk} \leq 1 \quad \forall j \in \{0, \dots, n+e\}, j \neq i, \quad (4.66)$$

$$\forall k \in \{1, \dots, m\}$$

$$u_i - u_j + n x_{ijk} \leq n - 1 \quad \forall i, j \in \{1, \dots, n\}, i \neq j, \quad (4.67)$$

$$\forall k \in \{1, \dots, m\}$$

$$\sum_{\substack{i=0 \\ i \neq s}}^n x_{isk} = \sum_{\substack{j=1 \\ j \neq s}}^{n+e} x_{sjk} \quad \forall s \in \{1, \dots, n\}, \forall k \in \{1, \dots, m\} \quad (4.68)$$

$$\sum_{i=0}^n \sum_{k=1}^m z_{ijk} = q_j \quad \forall j \in \{1, \dots, n\}, i \neq j \quad (4.69)$$

$$z_{ijk} \leq U x_{ijk} \quad \begin{aligned} & \forall i \in \{0, \dots, n\}, \forall j \in \{1, \dots, n\}, \\ & \forall k \in \{1, \dots, m\} \end{aligned} \quad (4.70)$$

$$\sum_{i=0}^n z_{ijk} \leq y_{jk} \quad \forall j \in \{1, \dots, n\}, i \neq j, \forall k \in \{1, \dots, m\} \quad (4.71)$$

$$\sum_{i=0}^n z_{ijk} \geq y_{jk} - U \left( 1 - \sum_{i=0}^n x_{ijk} \right) \quad \forall j \in \{1, \dots, n\}, \forall k \in \{1, \dots, m\} \quad (4.72)$$

$$\sum_{k=1}^m y_{jk} = q_j \quad \forall j \in \{1, \dots, n\} \quad (4.73)$$

$$\sum_{i=0}^n x_{ijk} \leq y_{jk} \quad \forall j \in \{1, \dots, n\}, \forall k \in \{1, \dots, m\} \quad (4.74)$$

$$c\_veh_k = \sum_{j=1}^n y_{jk} \quad \forall k \in \{1, \dots, m\} \quad (4.75)$$

$$d\_ext_k = \sum_{i=0}^{n+e} \sum_{\substack{j=0 \\ i \neq j}}^{n+e} c_{ij} x_{ijk} \quad \forall k \in \{1, \dots, m\} \quad (4.76)$$

$$d\_int_k = \sum_{i=0}^n \sum_{\substack{j=0 \\ i \neq j}}^n x_{ijk} p_j f_j \quad \forall k \in \{1, \dots, m\} \quad (4.77)$$

$$t\_ext_k = \frac{d\_ext_k}{v_c} \quad \forall k \in \{1, \dots, m\} \quad (4.78)$$

$$t\_int_k = \frac{d\_int_k}{v_p} \quad \forall k \in \{1, \dots, m\} \quad (4.79)$$

$$t\_con_k = \sum_{j=1}^n y_{jk} \frac{n_c}{q_t} t_r \quad \forall k \in \{1, \dots, m\} \quad (4.80)$$

$$t\_vac_k = \sum_{i=0}^n (x_{ie_1 k} + x_{ie_2 k}) t_v \quad \forall k \in \{1, \dots, m\} \quad (4.81)$$

$$t\_veh_k = t\_ext_k + t\_int_k + t\_con_k + t\_vac_k \quad \forall k \in \{1, \dots, m\} \quad (4.82)$$

$$\sum_{i=1}^n (x_{ie_1 k} + x_{ie_2 k}) = \sum_{j=1}^n x_{0jk} \quad \forall k \in \{1, \dots, m\} \quad (4.83)$$

$$\sum_{i=1}^n x_{iek} = x_{e0k} \quad \forall k \in \{1, \dots, m\}, \forall e \in \{e_1, e_2\} \quad (4.84)$$

$$x_{e_10k} + x_{e_20k} = 1 \quad \forall k \in \{1, \dots, m\} \quad (4.85)$$

$$\sum_{j=1}^n x_{ejk} = 0 \quad \forall k \in \{1, \dots, m\}, \forall e \in \{e_1, e_2\} \quad (4.86)$$

$$x_{0ek} = 0 \quad \forall k \in \{1, \dots, m\}, \forall e \in \{e_1, e_2\} \quad (4.87)$$

$$\sum_{i=1}^n x_{iek} \geq 1 \quad \forall k \in \{1, \dots, m\}, \forall e \in \{e_1, e_2\} \quad (4.88)$$

$$q_i \leq u_i \leq Q \quad \forall i \in \{1, \dots, n\}$$

$$x_{ijk} \in \{0,1\} \quad \forall (i,j,k)$$

$$z_{ijk}, y_{jk}, c\_veh_k, d\_ext_k, d\_int_k, t\_ext_k, t\_int_k, t\_con_k, t\_vac_k, t\_veh_k \in \mathbb{R}$$

# 5 APLICACIÓN DEL MODELO SOBRE LA PROVINCIA DE SEVILLA

---

Utilizando las formulaciones que se acaban de presentar en la sección anterior, ha llegado el momento de implementar estos modelos en las herramientas de software elegidas. El código se ha escrito en Python (Python Software Foundation, 2018) y ejecutado en Jupyter Notebook (Anaconda Software Distribution, 2018), haciendo uso del software de resolución de programación lineal CPLEX (IBM ILOG, 2019). En primer lugar, se discute la implementación del modelo en estas herramientas. En la segunda parte de este capítulo, se presenta el formato de los datos utilizados y cómo se han tratado para la resolución del problema. Finalmente, este capítulo concluye presentando los resultados obtenidos tras ejecutar la optimización desarrollada.

## 5.1 Implementación en CPLEX/Python

Existen distintos resolutores de problemas de programación lineal, cada uno con sus ventajas e inconvenientes. Tras un primer repaso a la literatura relacionada con el VRP, se decidió optar por el solver CPLEX puesto que aparecía en bastantes artículos académicos. Posteriormente, se descubrió la posibilidad de utilizar el lenguaje de programación Python junto con CPLEX, aprovechando las ventajas de ambas herramientas. Esto es posible gracias a la librería que comparten estos entornos. La ejecución de los programas se ha realizado en Jupyter Notebook.

La codificación completa para cada uno de los modelos discutidos en la Sección 4 se puede encontrar en el Anexo II. No obstante, se presenta una restricción con algunos elementos representativos junto a su equivalencia en el lenguaje de programación, se han utilizado colores para identificar las distintas partes de la expresión.

Para facilitar la compresión del ejemplo, se proporciona el contenido de algunos elementos que aparecen en el código: `index_depot=[0,...,n]`; `index_est=[1,...,n+e]` y `index_municipios=[1,...,n]`.

---

Modelo matemático	$\sum_{\substack{i=0 \\ i \neq s}}^n x_{isk} = \sum_{\substack{j=1 \\ j \neq s}}^{n+e} x_{sjk} \quad \forall s \in \{1, \dots, n\}, \forall k \in \{1, \dots, m\}$
-------------------	--

Python	<pre>mdl.add_constraints(mdl.sum(x[i,s,k] for i in index_depot if i!=s) == mdl.sum(x[s,j,k] for j in index_est if j!=s) for s in index_municipios for k in range(1,m+1))</pre>
--------	--

Tabla 5.1. Ejemplo de código en Python

Uno de los mayores atractivos de Python reside en la legibilidad de su código. Como se puede comprobar en la restricción ofrecida a modo de ejemplo. La manera de redactar las restricciones es bastante literal, permitiendo una traducción a código muy intuitiva. Para la codificación de las restricciones se han seguido las lecciones y explicaciones de (Correa, 2019) y (Díaz, 2017). A continuación, para comprender la simpleza de la programación en Python se detallan el significado de algunos elementos:

- `add_constraints`: Se utiliza para añadir múltiples restricciones con la misma estructura al modelo.
- `sum`: Se introduce un sumatorio sobre los índices que se detallan en los paréntesis que lo delimitan.
- `for`: Bucle en el que se apoya el sumatorio y la creación de restricciones.
- `Range`: Función que, en este caso, genera una serie numérica de enteros desde 1 hasta m.

En las siguientes tablas, se ofrece la equivalencia que han recibido los distintos elementos que componen el modelo en el programa de Python, se recuerda también el significado de cada una. Se ha intentado mantener el orden de aparición en el modelo, a fin de facilitar la búsqueda de cada elemento en caso de que sea necesario. Se ha distinguido en tres grupos: en la Tabla 5.2 se enumeran los parámetros, en la Tabla 5.3 las variables de decisión y por último, en la Tabla 5.4 se detallan las variables auxiliares.

Notación	Código	Tipo	Descripción
$n$	<code>n</code>	Entero	Número de municipios
$e$	<code>e</code>	Entero	Número de plantas de destino
$m$	<code>m</code>	Entero	Número de vehículos
$p_j$	<code>p[j]</code>	Decimal	Perímetro del municipio $j$ , en km
$f_j$	<code>fcon[j]</code>	Decimal	Factor de concentración del municipio $j$
$q_i$	<code>q[i]</code>	Decimal	Producción de residuos del municipio $i$ , en kg
$U$	<code>U</code>	Entero	Límite superior de la producción de residuos, en kg
$v_c$	<code>vc</code>	Entero	Velocidad en carretera, en km/h
$v_p$	<code>vp</code>	Entero	Velocidad en población, en km/h
$n_c$	<code>cont</code>	Entero	Número de contenedores de la UGR
$q_t$	<code>q_total</code>	Decimal	Cantidad total de residuos generada, en kg
$t_r$	<code>Tr</code>	Decimal	Tiempo de recogida por contenedor, en h
$t_v$	<code>Tv</code>	Decimal	Tiempo de descarga de los vehículos, en h
$Q$	<code>Q</code>	Entero	Capacidad máxima de los vehículos, en kg

Tabla 5.2. Parámetros

Notación	Código	Tipo	Descripción
$x_{ijk}$	$x[i, j, k]$	Binaria	Toma el valor 1 cuando el arco $(i, j)$ es recorrido por el vehículo $k$ , y 0 en caso contrario
$u_i$	$u[i]$	Entera	Variable auxiliar para eliminar subciclos
$z_{ijk}$	$z[i, j, k]$	Decimal	Variable de linealización para restricción (4.7)
$y_{jk}$	$y[j, k]$	Decimal	Cantidad de basura recogida (en kg) en el municipio $j$ , por el vehículo $k$

Tabla 5.3. Variables de decisión

Notación	Código	Tipo	Descripción
$c_{veh_k}$	$c\_veh[k]$	Decimal	Carga del vehículo al final de la ruta, en kg
$d_{veh_k}$	$d\_veh[k]$	Decimal	Distancia recorrida por el vehículo $k$ al final de la ruta, en km
$d_{ext_k}$	$d\_ext[k]$	Decimal	Distancia recorrida por el vehículo $k$ entre los municipios incluidos en la ruta, en km
$d_{int_k}$	$d\_int[k]$	Decimal	Distancia recorrida por el vehículo $k$ en el interior de los municipios visitados en la ruta, en km
$t_{veh_k}$	$t\_veh[k]$	Decimal	Tiempo total empleado por el vehículo $k$ al final de la ruta, en h
$t_{ext_k}$	$t\_ext[k]$	Decimal	Tiempo empleado por el vehículo $k$ entre los municipios incluidos en la ruta, en h
$t_{int_k}$	$t\_int[k]$	Decimal	Tiempo empleado por el vehículo $k$ en el interior de los municipios visitados en la ruta, en h
$t_{con_k}$	$t\_con[k]$	Decimal	Tiempo empleado por el vehículo $k$ en cargar los contendores de los municipios visitados en la ruta, en h
$t_{vac_k}$	$t\_vac[k]$	Decimal	Tiempo empleado por el vehículo $k$ en vaciar la carga en la planta de destino, en h

Tabla 5.4. Variables auxiliares

Como se observa en estas tablas, se combinan variables y parámetros de tipo entero y decimal. Lo que hace que sea un problema de Programación Lineal Entera-Mixta (MILP, por su nombre en inglés *Mixed-Integer Linear Program*). CPLEX aborda la resolución de este tipo de problemas MILP haciendo uso de las técnicas *Branch and Cut* para la exploración del espacio de las soluciones. La optimización planteada por CPLEX comprende encontrar una sucesión de soluciones factibles cada vez mejores, mientras que a la vez trabaja para probar que no existe ninguna solución mejor. (IBM Corp., 2017)

## 5.2 Estructuras de Datos

Antes de realizar los cálculos, es necesario tratar la información y estructurarla de manera que puedan ser procesados como datos de entrada por los programas utilizados. El formato empleado para organizar esta información se presenta en la Tabla 5.5. Estas estructuras de datos se localizan en hojas de Excel, una para cada UGR y son leídas directamente al ejecutar el programa proporcionando la dirección de memoria en la que ubican. En el Anexo I, se especifican todos los datos utilizados para cada mancomunidad. A continuación, se detalla la información que se proporciona en cada columna:

- (a) Municipio: Esta columna se utiliza para identificar a los clientes y será utilizada en la representación gráfica de la solución. La primera fila en todas las estructuras se corresponde con el depósito y no se especifica nombre.
- (b) Producción diaria de residuos: Se ha dividido entre 365 días la información presentada en el apartado 3.2.2, procedente de (Diputación de Sevilla, 2019).
- (c) Latitud, Longitud: Coordenadas geográficas obtenidas desde *Google Maps*. (Google, 2020)
- (d) Factor de concentración: Parámetro que varía entre los valores 1, 1.5 y 2. Este factor se calcula externamente en una hoja de Excel según se ha explicado en apartados anteriores. No se define ni para el depósito ni para las plantas de destino.
- (e) Perímetro: Obtenido a través de la superficie total de cada municipio, considerando el municipio como un círculo. Información obtenida en (Instituto Nacional de Estadística, 2020).

Nº	Municipio (a)	Prod. Residuos (kg/día) (b)	Latitud (c)	Longitud (c)	Factor de concentración (d)	Perímetro (km) (e)
0						
1						
...						
$n$						
...						
$n + e$						

Tabla 5.5. Formato para Estructura de Datos

Por otro lado, los parámetros que se presentan bajo estas líneas se introducen directamente durante la ejecución del programa. En la Tabla 5.6 se recogen los valores de aquellos parámetros comunes a todas las mancomunidades; mientras que en el A se proporciona el valor de los parámetros que son particulares para cada UGR, utilizando el formato que presentado en la Tabla 5.7.

Parámetro	Descripción	Valor asignado
Q	Capacidad máxima de los vehículos	14.000 kg
U	Límite superior de la producción de residuos	1.000.000 kg
Vc	Velocidad en carretera	50 km/h
Vp	Velocidad en población	25 km/h
Tr	Tiempo de recogida por contenedor	0.015 h
Tv	Tiempo de descarga de los vehículos	0.5 h

Tabla 5.6. Parámetros comunes para todas las UGRs.

Parámetro	Descripción	Valor asignado
m	Número de vehículos	
cont	Número de contenedores	

Tabla 5.7. Formato para parámetros particulares

### 5.3 Resultados

En este apartado se revisan los resultados obtenidos tras el proceso de optimización. Puesto que cada mancomunidad se resuelve independiente al resto, se van a presentar los resultados por separado. No obstante, el formato en el que se proporcionan las soluciones es el mismo. Para cada UGR se tiene: una gráfica donde se representan los municipios y las rutas, un mapa interactivo también con los municipios y rutas; y por último una tabla de Excel donde se proporcionan los valores de las variables más representativas del problema.

#### 5.3.1 UGR nº1 – Mancomunidad de los Alcores

La Mancomunidad de los Alcores genera 155.673 kg de residuos al día. Para satisfacer esta demanda se utiliza una flota de 13 camiones de basura que dan servicio a los 10 municipios que integran la agrupación. La Tabla 5.8 muestra el valor que toman las variables auxiliares en el óptimo. Por otro lado, la Figura 5.1 y la Figura 5.2 ofrecen la representación gráfica del resultado. Siguiendo las rutas que se presentan en esas figuras, el valor de la función objetivo es de **1243,8 km** recorridos por la flota de esta mancomunidad. Las rutas que toma cada vehículo se recogen en el siguiente listado:

- **Ruta Vehículo 1:** DEPOT, Lebrija, ET de Lebrija, DEPOT
- **Ruta Vehículo 2:** DEPOT, Carmona, PT Montemarta-Cónica, DEPOT
- **Ruta Vehículo 3:** DEPOT, Mairena del Alcor, PT Montemarta-Cónica, DEPOT
- **Ruta Vehículo 4:** DEPOT, Los Molares, El Coronil, PT Montemarta-Cónica, DEPOT
- **Ruta Vehículo 5:** DEPOT, Mairena del Alcor, PT Montemarta-Cónica, DEPOT
- **Ruta Vehículo 6:** DEPOT, El Viso del Alcor, PT Montemarta-Cónica, DEPOT
- **Ruta Vehículo 7:** DEPOT, Lebrija, ET de Lebrija, DEPOT
- **Ruta Vehículo 8:** DEPOT, Carmona, El Viso del Alcor, PT Montemarta-Cónica, DEPOT
- **Ruta Vehículo 9:** DEPOT, Las Cabezas de San Juan, PT Montemarta-Cónica, DEPOT
- **Ruta Vehículo 10:** DEPOT, Carmona, PT Montemarta-Cónica, DEPOT
- **Ruta Vehículo 11:** DEPOT, Coripe, Montellano, PT Montemarta-Cónica, DEPOT
- **Ruta Vehículo 12:** DEPOT, Las Cabezas de San Juan, Lebrija, ET Lebrija, DEPOT
- **Ruta Vehículo 13:** DEPOT, El Cuervo de Sevilla, PT Montemarta-Cónica, DEPOT

Vehículo	c_veh (kg)	d_veh (km)	t_veh (h)	d_ext (km)	d_int (km)	t_ext (h)	t_int (h)	t_con (h)	t_vac (h)
1	12843	115,5	8,0	103,5	12	2,1	0,48	5	0,5
2	13730	90,4	8,0	70,4	20	1,4	0,8	5,3	0,5
3	14000	52,4	7,1	43,4	9	0,87	0,36	5,4	0,5
4	9119	92,1	6,2	75,6	16,5	1,5	0,66	3,5	0,5

Tabla 5.8. UGR nº1 – Resultados

Vehículo	c_veh (kg)	d_veh (km)	t_veh (h)	d_ext (km)	d_int (km)	t_ext (h)	t_int (h)	t_con (h)	t_vac (h)
5	12106	52,4	6,4	43,4	9	0,87	0,36	4,7	0,5
6	14000	57,4	7,2	48,4	9	0,97	0,36	5,4	0,5
7	12848	115,4	8,0	103,4	12	2,1	0,48	5,0	0,5
8	11523	99,5	7,5	70,5	29	1,4	1,2	4,4	0,5
9	14000	91,9	7,9	81,4	10,5	1,6	0,42	5,4	0,5
10	13730	90,4	8,0	70,4	20	1,4	0,8	5,3	0,5
11	9209	132,3	7,0	116,8	15,5	2,3	0,62	3,5	0,5
12	9019	129,2	7,0	106,7	22,5	2,1	0,9	3,5	0,5
13	9545	124,9	6,9	114,4	10,5	2,3	0,42	3,7	0,5

Tabla 5.9. UGR nº1 – Resultados (Continuación)

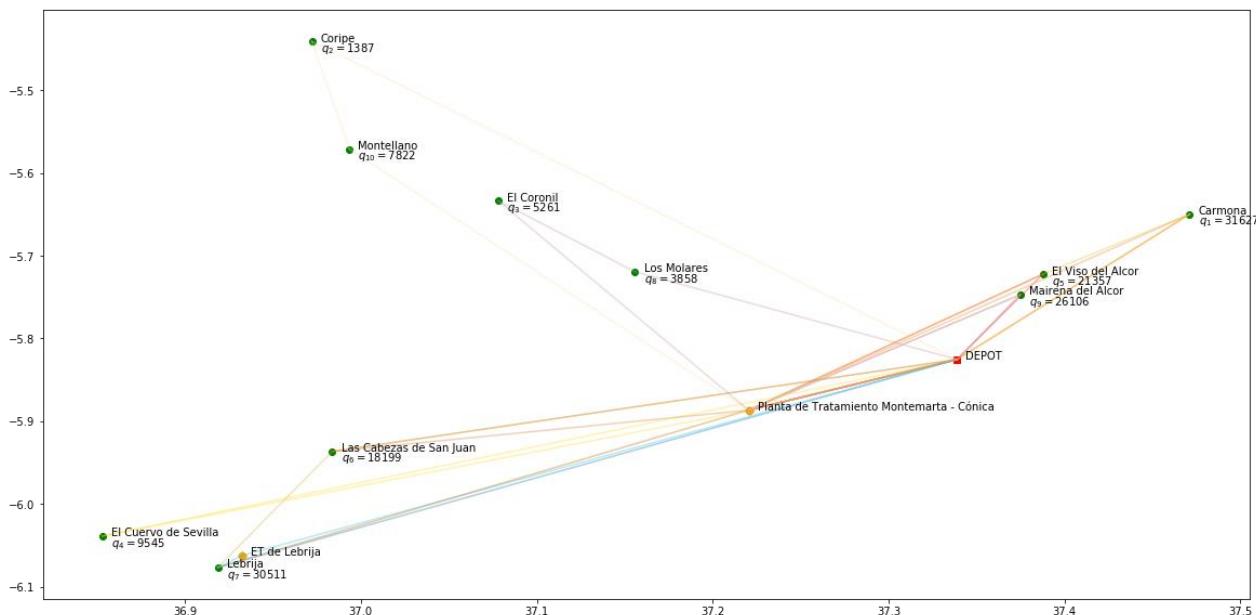


Figura 5.1. UGR nº1 – Gráfica con rutas

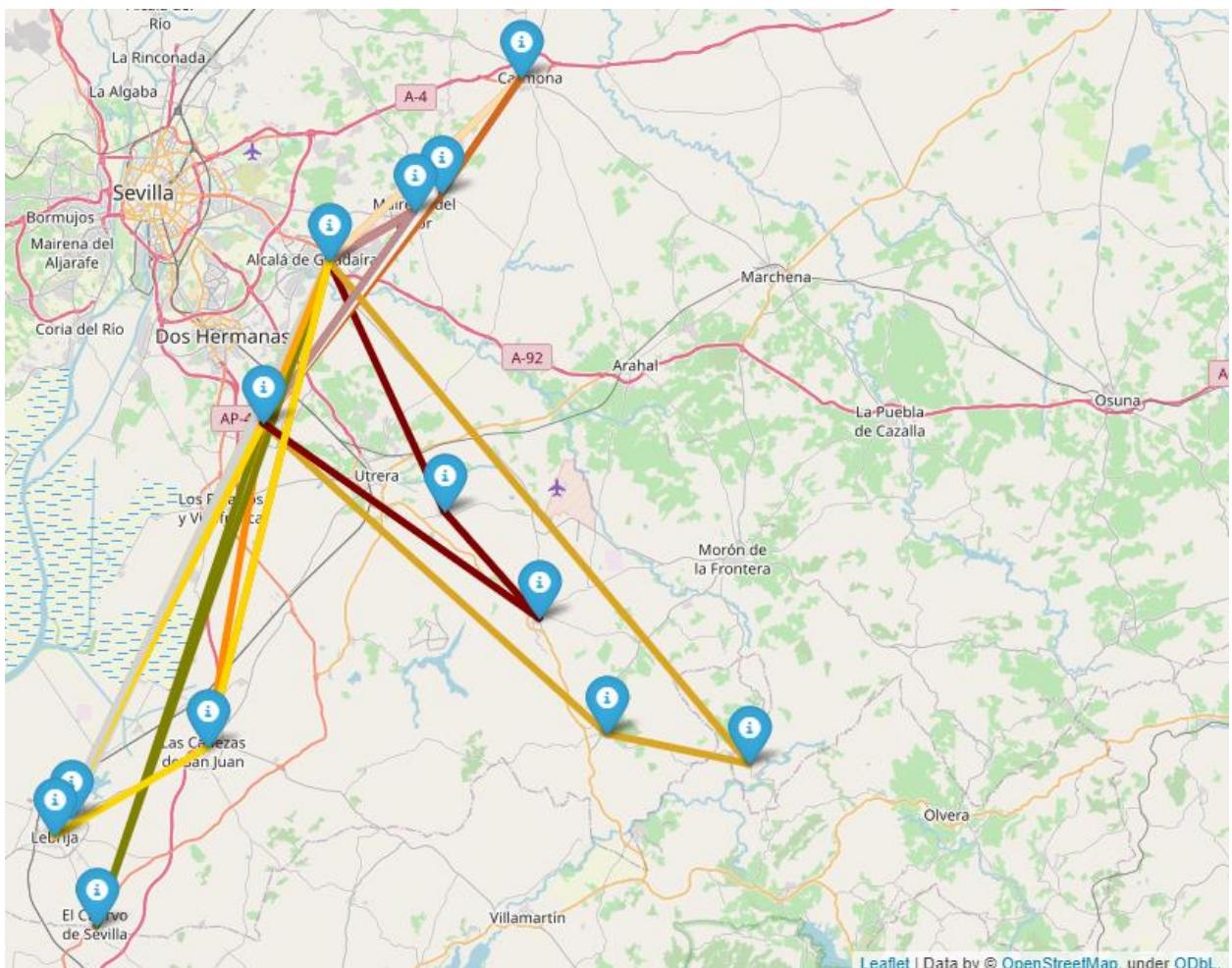


Figura 5.2. UGR nº1 – Mapa con rutas

### 5.3.2 UGR nº2 – Mancomunidad del Guadalquivir

En esta UGR el sistema de recogida conjunta de basuras ya está en funcionamiento. Sin embargo, debido a las modificaciones propuestas en el *Plan Provincial de Residuos No Peligrosos de la Provincia de Sevilla* (Diputación de Sevilla, 2019) se han adaptado los cálculos para el total de los 31 municipios propuestos. La flota utilizada es de 32 vehículos. Los resultados asociados a cada vehículo se detallan en la Tabla 5.8. Por otro lado, la solución óptima obtenida es de un total de **1297,5 km**, que corresponde con la suma de las distancias de todas las rutas que se presentan en la Figura 5.1 y la Figura 5.2. Estos trayectos se detallan a continuación:

- **Ruta Vehículo 1:** DEPOT, Gelves, San Juan de Aznalfarache, ET de Espartinas, DEPOT
- **Ruta Vehículo 2:** DEPOT, Castilleja de la Cuesta, Camas, ET de Espartinas, DEPOT
- **Ruta Vehículo 3:** DEPOT, Castilleja de la Cuesta, Castilleja de Guzmán, ET de Espartinas, DEPOT
- **Ruta Vehículo 4:** DEPOT, Huévar del Aljarafe, Pilas, Castilleja del Campo, Carrión de los Céspedes, ET de Espartinas, DEPOT
- **Ruta Vehículo 5:** DEPOT, Bollullos de la Mitación, ET de Espartinas, DEPOT
- **Ruta Vehículo 6:** DEPOT, Valencina de la Concepción, ET de Espartinas, DEPOT
- **Ruta Vehículo 7:** DEPOT, La Puebla del Río, Almensilla, ET de Espartinas, DEPOT
- **Ruta Vehículo 8:** DEPOT, Huévar del Aljarafe, Pilas, Espartinas, ET de Espartinas, DEPOT
- **Ruta Vehículo 9:** DEPOT, Almensilla, Gelves, ET de Espartinas, DEPOT
- **Ruta Vehículo 10:** DEPOT, Coria del Río, Gelves, ET de Espartinas, DEPOT
- **Ruta Vehículo 11:** DEPOT, Tomares, ET de Espartinas, DEPOT
- **Ruta Vehículo 12:** DEPOT, Albaida del Aljarafe, Aznalcóllar, Espartinas, ET de Espartinas, DEPOT
- **Ruta Vehículo 13:** DEPOT, Gines, ET de Espartinas, DEPOT

- **Ruta Vehículo 14:** DEPOT, Coria del Rio, ET de Espartinas, DEPOT
- **Ruta Vehículo 15:** DEPOT, Camas, ET de Espartinas, DEPOT
- **Ruta Vehículo 16:** DEPOT, Olivares, ET de Espartinas, DEPOT
- **Ruta Vehículo 17:** DEPOT, Espartinas, ET de Espartinas, DEPOT
- **Ruta Vehículo 18:** DEPOT, Benacazón, ET de Espartinas, DEPOT
- **Ruta Vehículo 19:** DEPOT, Camas, ET de Espartinas, DEPOT
- **Ruta Vehículo 20:** DEPOT, Palomares del Río, La Puebla del Río, Coria del Río, ET de Espartinas, DEPOT
- **Ruta Vehículo 21:** DEPOT, Sanlúcar la Mayor, ET de Espartinas, DEPOT
- **Ruta Vehículo 22:** DEPOT, Bormujos, ET de Espartinas, DEPOT
- **Ruta Vehículo 23:** DEPOT, Sanlúcar la Mayor, ET de Espartinas, DEPOT
- **Ruta Vehículo 24:** DEPOT, Bormujos, Gines, ET de Espartinas, DEPOT
- **Ruta Vehículo 25:** DEPOT, Villanueva del Ariscal, ET de Espartinas, DEPOT
- **Ruta Vehículo 26:** DEPOT, Tomares, Camas, Santiponce, Valencina de la Concepción, ET de Espartinas, DEPOT
- **Ruta Vehículo 27:** DEPOT, Umbrete, ET de Espartinas, DEPOT
- **Ruta Vehículo 28:** DEPOT, Tomares, San Juan de Aznalfarache, ET de Espartinas, DEPOT
- **Ruta Vehículo 29:** DEPOT, Aznalcázar, Villamanrique de la Condesa, ET de Espartinas, DEPOT
- **Ruta Vehículo 30:** DEPOT, Salteras, ET de Espartinas, DEPOT
- **Ruta Vehículo 31:** DEPOT, San Juan de Aznalfarache, ET de Espartinas, DEPOT
- **Ruta Vehículo 32:** DEPOT, Isla Mayor, Coria del Río, ET de Espartinas, DEPOT

Vehículo	c_veh (kg)	d_veh (km)	t_veh (h)	d_ext (km)	d_int (km)	t_ext (h)	t_int (h)	t_con (h)	t_vac (h)
1	14000	41,2	6,9	33,2	8	0,66	0,32	5,4	0,5
2	14000	40	6,9	29	11	0,58	0,44	5,4	0,5
3	14000	35,1	6,8	27,1	8	0,5	0,32	5,4	0,5
4	11537	74,4	6,8	53,9	20,5	1,1	0,82	4,4	0,5
5	12765	30,5	6,2	21,5	9	0,43	0,36	4,9	0,5
6	9171	27,9	4,7	21,9	6	0,44	0,24	3,5	0,5
7	14000	52,9	7,2	40,9	12	0,82	0,48	5,4	0,5
8	14000	59,6	7,4	42,1	17,5	0,84	0,7	5,4	0,5
9	14000	43,4	6,9	36,4	7	0,73	0,28	5,4	0,5
10	14000	48,6	7,0	39,6	9	0,79	0,36	5,4	0,5
11	14000	36,1	6,8	27,1	9	0,54	0,36	5,4	0,5
12	14000	67,1	7,6	50,1	17	1	0,68	5,4	0,5

Tabla 5.10. UGR nº2 – Resultados

Vehículo	c_veh (kg)	d_veh (km)	t_veh (h)	d_ext (km)	d_int (km)	t_ext (h)	t_int (h)	t_con (h)	t_vac (h)
13	14000	27,9	6,6	20,9	7	0,42	0,28	5,4	0,5
14	14000	42,9	6,9	36,9	6	0,74	0,24	5,4	0,5
15	14000	33,7	6,7	28,7	5	0,57	0,2	5,4	0,5
16	11116	25,6	5,4	18,1	7,5	0,36	0,3	4,3	0,5
17	14000	20,4	6,4	16,4	4	0,33	0,16	5,4	0,5
18	8569	27,8	4,5	20,3	7,5	0,41	0,3	3,3	0,5
19	14000	33,7	6,7	28,7	5	0,57	0,2	5,4	0,5
20	14000	62,7	7,6	41,7	21	0,83	0,84	5,4	0,5
21	2339	25,2	2,1	17,7	7,5	0,35	0,3	0,9	0,5
22	14000	31,2	6,7	23,2	8	0,46	0,32	5,4	0,5
23	14000	25,2	6,5	17,7	7,5	0,35	0,3	5,4	0,5
24	13880	38,4	6,9	23,4	15	0,47	0,6	5,3	0,5
25	7822	20,8	4	15,8	5	0,32	0,2	3,0	0,5
26	14000	58,8	7,6	34,8	24	0,70	0,96	5,4	0,5
27	10525	20,8	5,1	16,8	4	0,3	0,16	4,1	0,5
28	14000	44,2	7,1	30,2	14	0,60	0,56	5,4	0,5
29	10704	63,2	6,1	50,7	12,5	1,01	0,5	4,1	0,5
30	6544	22,9	3,6	18,4	4,5	0,37	0,18	2,5	0,5
31	14000	34,9	6,7	29,9	5	0,60	0,2	5,4	0,5
32	12967	80,4	7,3	68,4	12	1,4	0,48	5	0,5

Tabla 5.11. UGR nº2 – Resultados (Continuación)

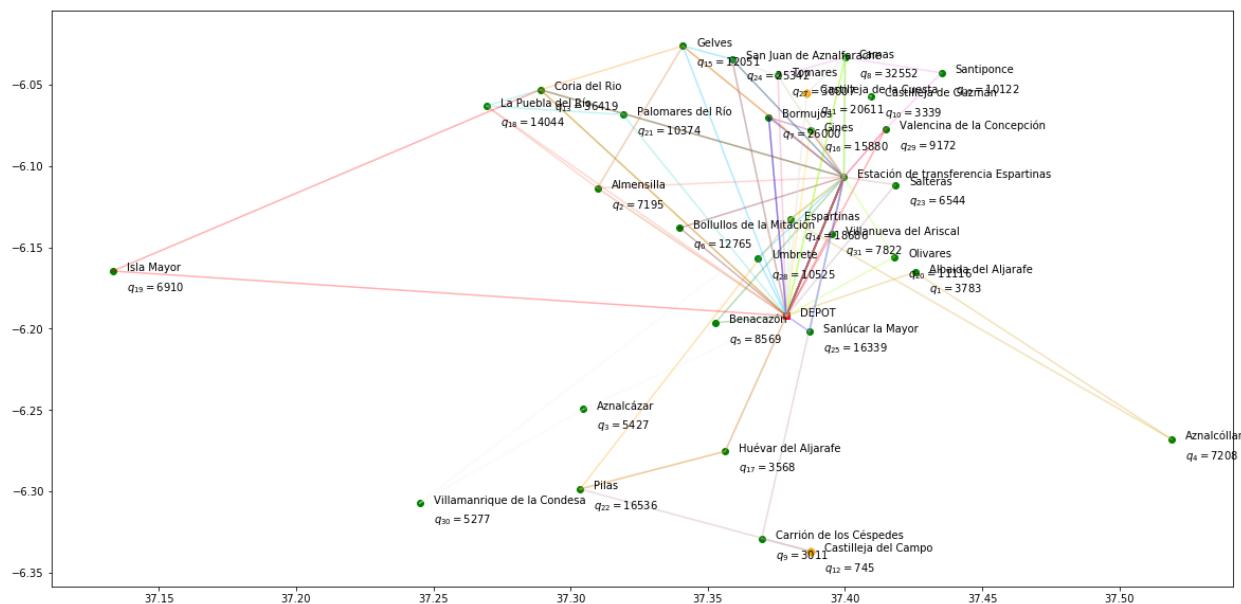
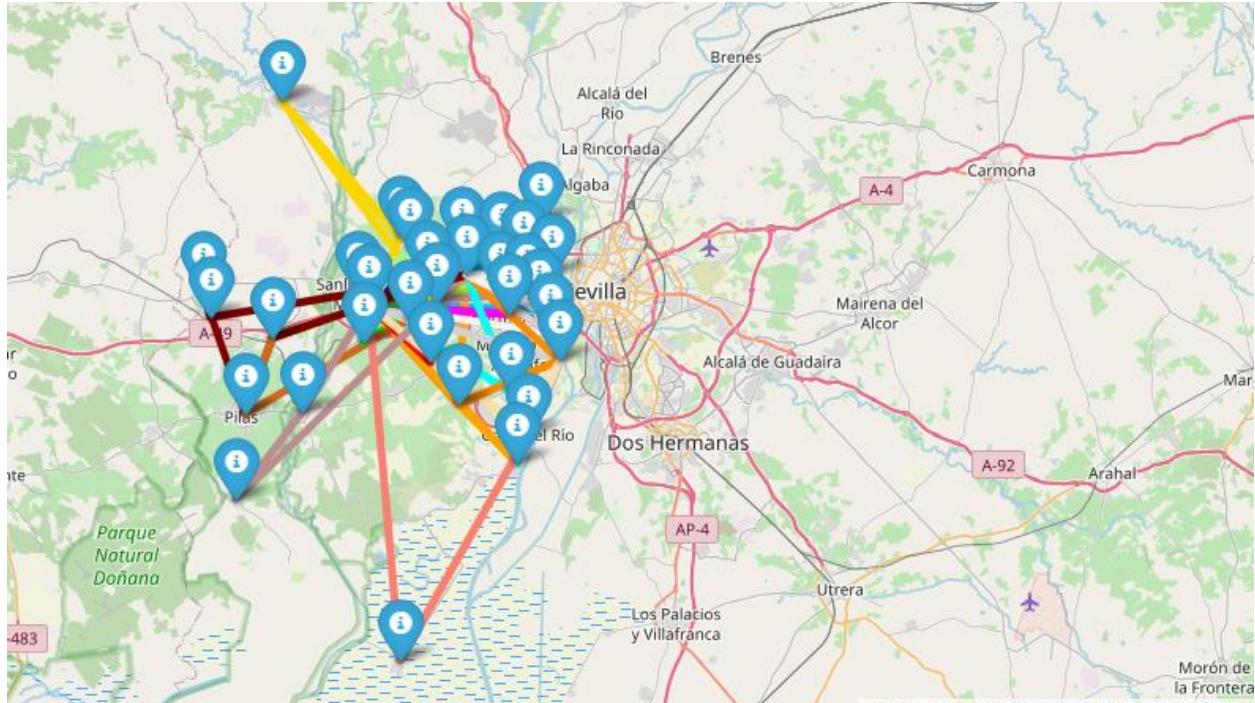


Figura 5.3. UGR nº2 – Gráfica con rutas



### 5.3.3 UGR nº3 – Mancomunidad de Servicios de la Vega

La Mancomunidad de Servicios de la Vega está compuesta por 18 municipios y el resultado obtenido tras el proceso de optimización es de **863.4 km** de recorrido para la flota completa., la cual cuenta con 16 camiones de recogida. Las rutas utilizadas se listan bajo este párrafo, mientras que la información que se deriva de la resolución del modelo se puede encontrar en la Tabla 5.12 y en los planos mostrados en la Figura 5.5 y la Figura 5.6.

- **Ruta Vehículo 1:** DEPOT, Burguillos, PT Alcalá del Río (La Vega), DEPOT
- **Ruta Vehículo 2:** DEPOT, La Algaba, PT Alcalá del Río (La Vega), DEPOT
- **Ruta Vehículo 3:** DEPOT, Lora del Río, ET Lora del Río, DEPOT
- **Ruta Vehículo 4:** DEPOT, Castilblanco de los Arroyos, PT Alcalá del Río (La Vega), DEPOT
- **Ruta Vehículo 5:** DEPOT, Alcalá del Rio, La Algaba, PT Alcalá del Río (La Vega), DEPOT

- **Ruta Vehículo 6:** DEPOT, Lora del Río, ET Lora del Río, DEPOT
- **Ruta Vehículo 7:** DEPOT, Tocina, PT Alcalá del Río (La Vega), DEPOT
- **Ruta Vehículo 8:** DEPOT, Gerena, PT Alcalá del Río (La Vega), DEPOT
- **Ruta Vehículo 9:** DEPOT, Brenes, Villanueva del Río y Minas, Cantillana, PT Alcalá del Río (La Vega), DEPOT
- **Ruta Vehículo 10:** DEPOT, Villaverde del Río, PT Alcalá del Río (La Vega), DEPOT
- **Ruta Vehículo 11:** DEPOT, Cantillana, PT Alcalá del Río (La Vega), DEPOT
- **Ruta Vehículo 12:** DEPOT, Alcalá del Río, PT Alcalá del Río (La Vega), DEPOT
- **Ruta Vehículo 13:** DEPOT, El Ronquillo, El Castillo de las Guardas, El Madroño, El Garrobo, Guillena, PT Alcalá del Río (La Vega), DEPOT
- **Ruta Vehículo 14:** DEPOT, Alcolea del Río, Peñaflor, PT Alcalá del Río (La Vega), DEPOT
- **Ruta Vehículo 15:** DEPOT, Brenes, PT Alcalá del Río (La Vega), DEPOT
- **Ruta Vehículo 16:** DEPOT, Guillena, PT Alcalá del Río (La Vega), DEPOT

Vehículo	c_veh (kg)	d_veh (km)	t_veh (h)	d_ext (km)	d_int (km)	t_ext (h)	t_int (h)	t_con (h)	t_vac (h)
1	9813	14	4,7	8	6	0,16	0,24	3,8	0,5
2	14000	33,6	6,7	29,6	4	0,59	0,16	5,4	0,5
3	13267	94,2	7,6	86,7	7,5	1,7	0,3	5,1	0,5
4	7107	24	3,8	18	6	0,36	0,24	2,7	0,5
5	14000	39,1	6,9	30,6	8,5	0,61	0,34	5,4	0,5
6	14000	94,1	7,9	86,6	7,5	1,73	0,3	5,4	0,5
7	13882	53	7,0	49	4	0,98	0,16	5,4	0,5
8	11083	35,2	5,6	29,2	6	0,58	0,24	4,3	0,5
9	14000	75,6	7,7	59,6	16	1,2	0,64	5,4	0,5
10	11423	30,4	5,6	24,4	6	0,49	0,24	4,4	0,5
11	14000	39,4	6,8	33,4	6	0,67	0,24	5,4	0,5
12	13500	22,5	6,2	18	4,5	0,36	0,18	5,2	0,5
13	10349	126,8	7,5	102,8	24	2,06	0,96	4,0	0,5
14	10271	130,1	7,2	121,1	9	2,42	0,36	4,0	0,5
15	12917	31	6,2	27	4	0,54	0,16	5,0	0,5
16	14000	20,4	6,4	14,4	6	0,29	0,24	5,4	0,5

Tabla 5.12. UGR nº3 – Resultados

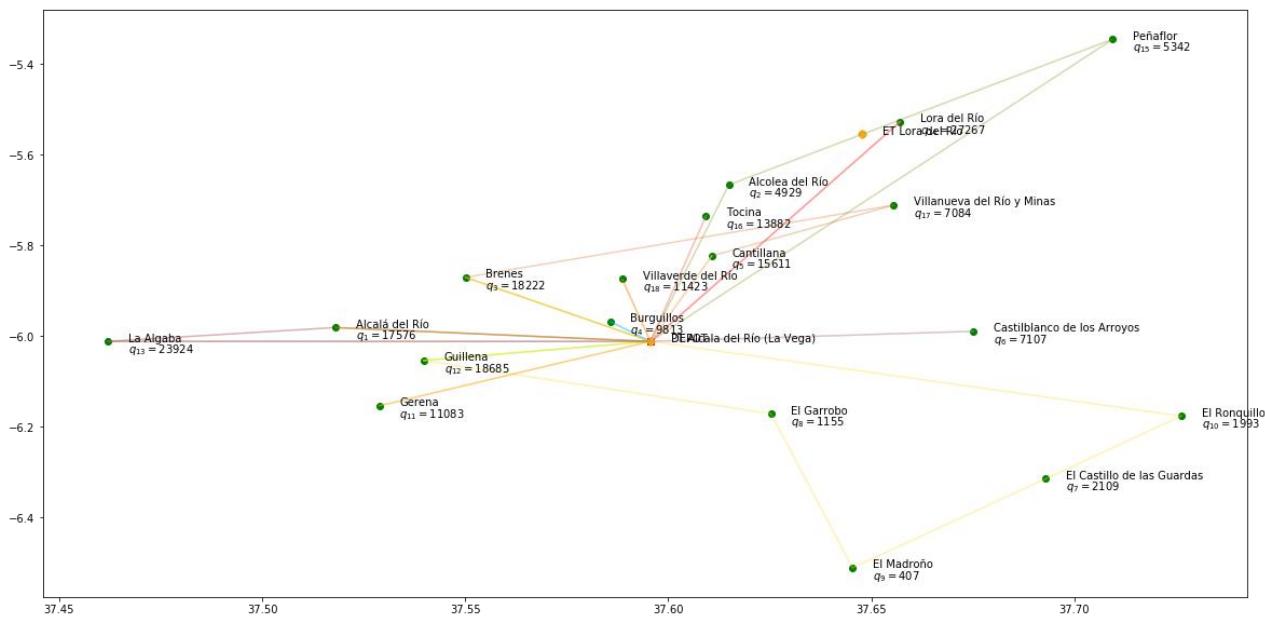


Figura 5.5. UGR nº3 – Gráfica con rutas

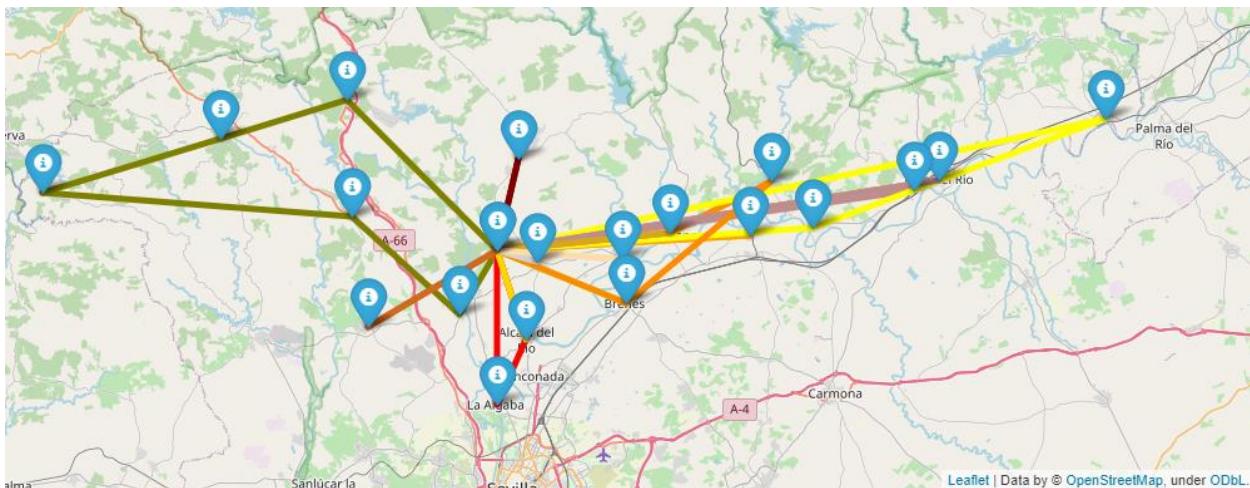


Figura 5.6. UGR nº3 – Mapa con rutas

### 5.3.4 UGR nº4 – Mancomunidad de Municipios Sierra Norte

Se establece una flota de tres vehículos para esta UGR, siendo una de las flotas más pequeñas de todo el proyecto. El valor mínimo que se encuentra para la función objetivo es de **329,8 km** recorridos por el conjunto de los vehículos. Estos recorridos se representan gráficamente en Figura 5.7 y Figura 5.8, mientras que los resultados numéricos se recogen en la Tabla 5.13. Los pueblos visitados en cada ruta se muestran en la siguiente lista, donde aparecen ordenados por orden de llegada.

- **Ruta Vehículo 1:** DEPOT, San Nicolas del Puerto, Las Navas de la Concepción, La Puebla de los Infantes, Constantina, ET de Constantina, DEPOT
- **Ruta Vehículo 2:** DEPOT, Cazalla de la Sierra, El Pedroso, ET de Constantina, DEPOT
- **Ruta Vehículo 3:** DEPOT, Alanís, Guadalcanal, El Real de la Jara, Almadén de la Plata, PT de la Vega, DEPOT

Vehículo	c_veh (kg)	d_veh (km)	t_veh (h)	d_ext (km)	d_int (km)	t_ext (h)	t_int (h)	t_con (h)	t_vac (h)
1	13223	110,5	5,1	86,5	24	1,7	0,96	13223	0,5
2	8066	52,9	3	38,9	14	0,78	0,56	8066	
3	8632	166,4	5,56	142,4	24	2,8	0,96	8632	0,5

Tabla 5.13. UGR nº4 – Resultados

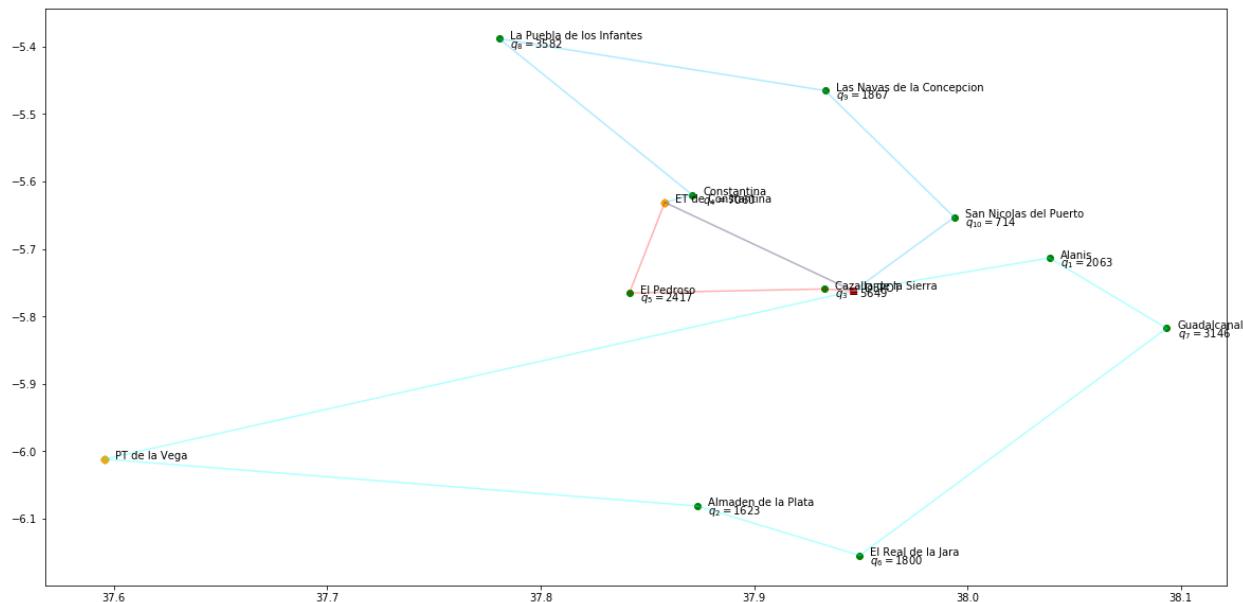


Figura 5.7. UGR nº4 – Gráfica con rutas

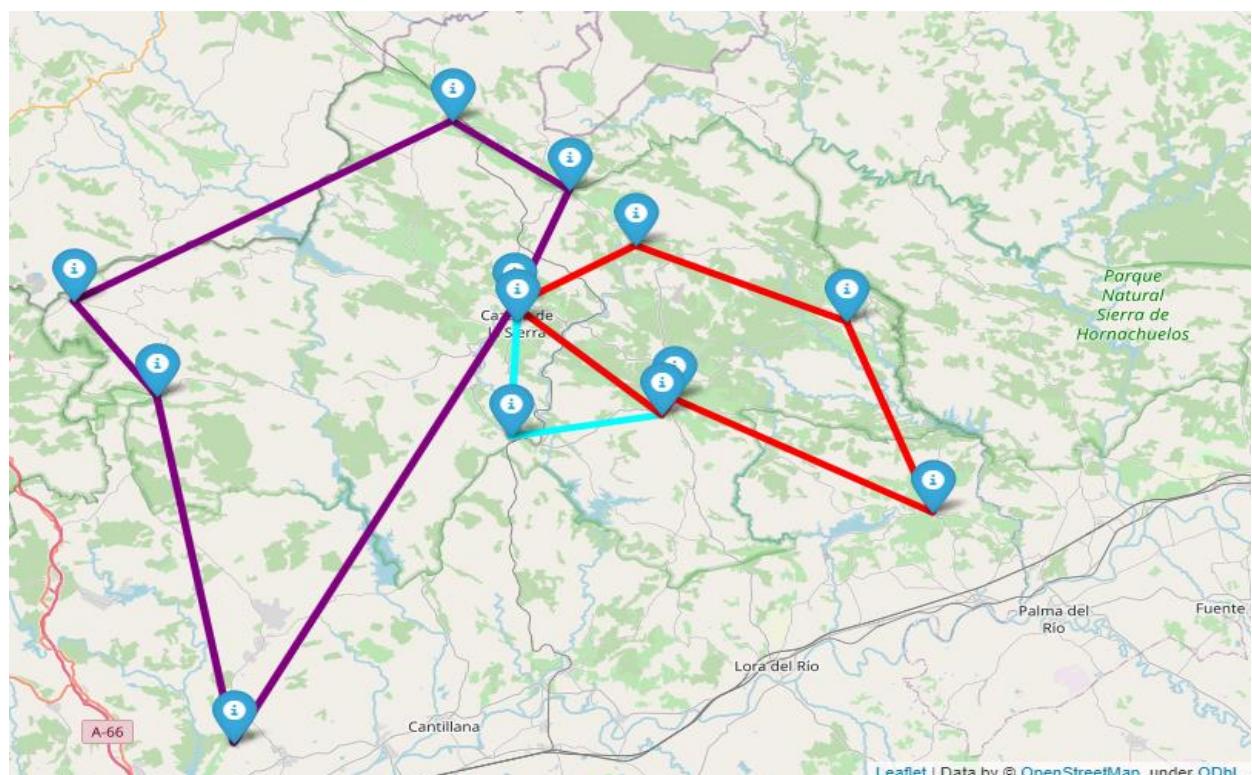


Figura 5.8. UGR nº4 – Mapa con rutas

### 5.3.5 UGR nº5 – Consorcio Estepa-Sierra Sur

El consorcio Estepa-Sierra Sur ofrece el servicio de recogida de RSU a 17 municipios sevillanos. Las rutas que toman los siete vehículos de los que disponen se representan en la Figura 5.8 y en la Figura 5.9. Estos recorridos se recogen también en el listado que se puede leer bajo estas líneas, detallando los municipios visitados por cada vehículo. Para esta configuración de rutas, el total de kilómetros recorridos es de **381,9 km**. El resto de datos obtenidos tras aplicar la optimización aparecen en la Tabla 5.14.

- **Ruta Vehículo 1:** DEPOT, Gilena, Pedrera, Lora de Estepa, PT de Mata Grande, DEPOT
- **Ruta Vehículo 2:** DEPOT, Herrera, PT de Mata Grande, DEPOT
- **Ruta Vehículo 3:** DEPOT, Martin de la Jara, Los Corrales, El Rubio, Marinaleda, PT de Mata Grande, DEPOT
- **Ruta Vehículo 4:** DEPOT, Aguadulce, El Saucejo, Villanueva de San Juan, Pruna, Algamitas, ET de El Saucejo, DEPOT
- **Ruta Vehículo 5:** DEPOT, Estepa, PT de Mata Grande, DEPOT
- **Ruta Vehículo 6:** DEPOT, La Roda de Andalucía, Badolatosa, Casariche, PT de Mata Grande, DEPOT
- **Ruta Vehículo 7:** DEPOT, Lora de Estepa, PT de Mata Grande, DEPOT

Vehículo	c_veh (kg)	d_veh (km)	t_veh (h)	d_ext (km)	d_int (km)	t_ext (h)	t_int (h)	t_con (h)	t_vac (h)
1	14000	44,6	7,1	30,6	14	0,61	0,56	5,4	0,5
2	6341	22,3	3,5	14,8	7,5	0,30	0,3	2,4	0,5
3	12446	90,7	7,5	69,7	21	1,4	0,84	4,8	0,5
4	11075	119,1	7,6	97,6	21,5	2,0	0,86	4,3	0,5
5	12273	17,1	5,7	11,1	6	0,22	0,24	4,7	0,5
6	12482	67	7,0	52	15	1,0	0,6	4,8	0,5
7	13071	21,1	6,0	19,1	2	0,38	0,08	5,	0,5

Tabla 5.14. UGR nº5 – Resultados

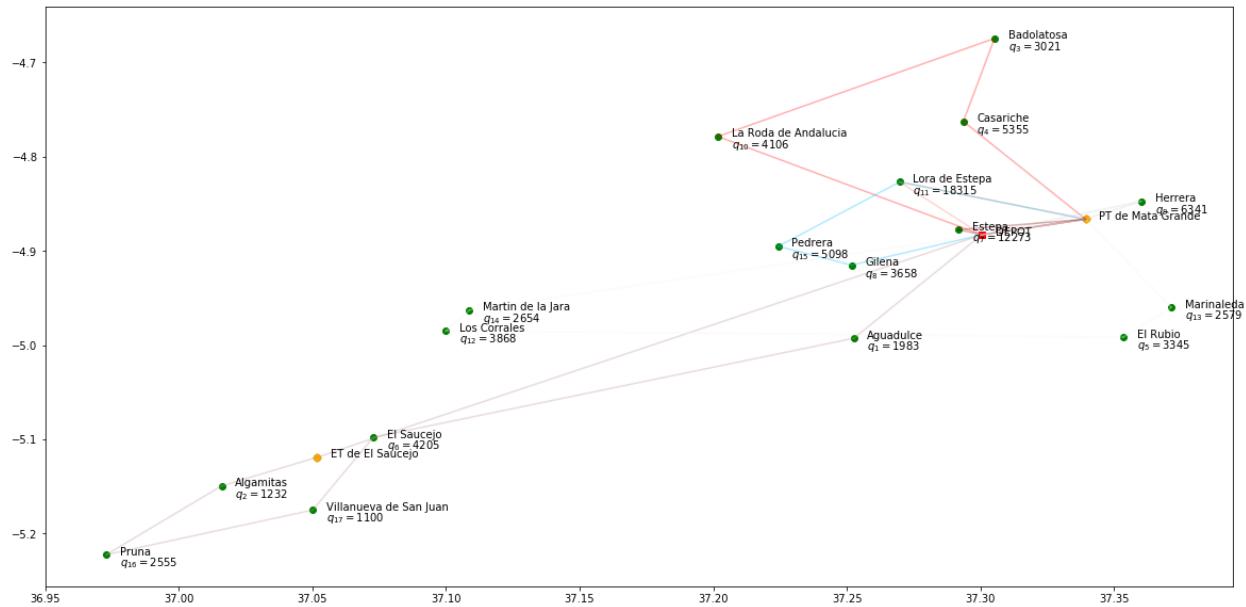


Figura 5.9. UGR nº5 – Gráfica con rutas

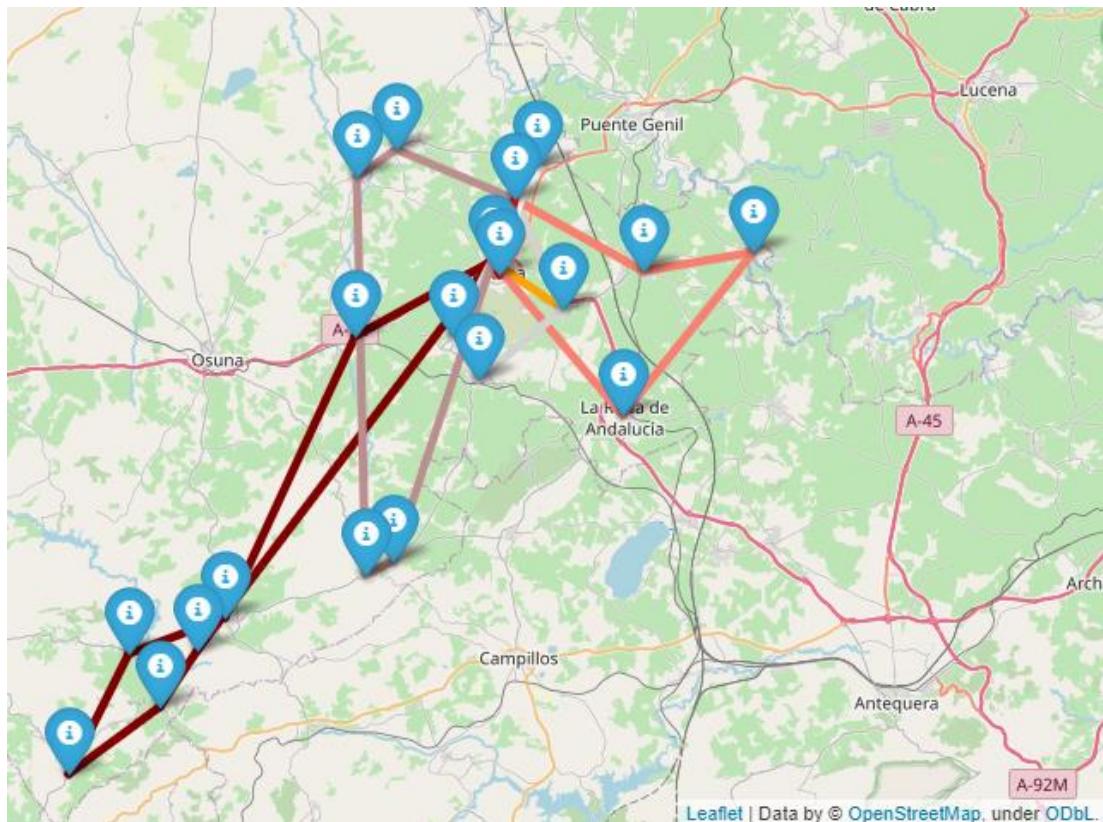


Figura 5.10. UGR nº5 – Mapa con rutas

### 5.3.6 UGR nº6 – Mancomunidad de Municipios Comarca de Écija

Esta entidad es la que cuenta con un menor número de municipios, cuatro en total. Se le ha asignado una flota de dos vehículos y tienen que recoger un total de 20283 kg de residuos. En la Tabla 5.15 se recogen los datos que se han obtenido del modelo, mientras que la Figura 5.11 y la Figura 5.12 muestran los recorridos generados. El valor que toma la función objetivo en el óptimo es **150,2 km**. En la siguiente lista se detallan las rutas seguidas por cada vehículo:

- **Ruta Vehículo 1:** DEPOT, La Luisiana, Fuentes de Andalucía, ET de Écija, DEPOT
- **Ruta Vehículo 2:** DEPOT, Cañada Rosal, La Campana, ET de Écija, DEPOT

Vehículo	c_veh (kg)	d_veh (km)	t_veh (h)	d_ext (km)	d_int (km)	t_ext (h)	t_int (h)	t_con (h)	t_vac (h)
1	13439	68,6	7,3	55,1	13,5	1,1	0,54	5,2	0,5
2	9885	81,6	6,2	69,6	12	1,4	0,48	3,8	0,5

Tabla 5.15. UGR nº6 – Resultados

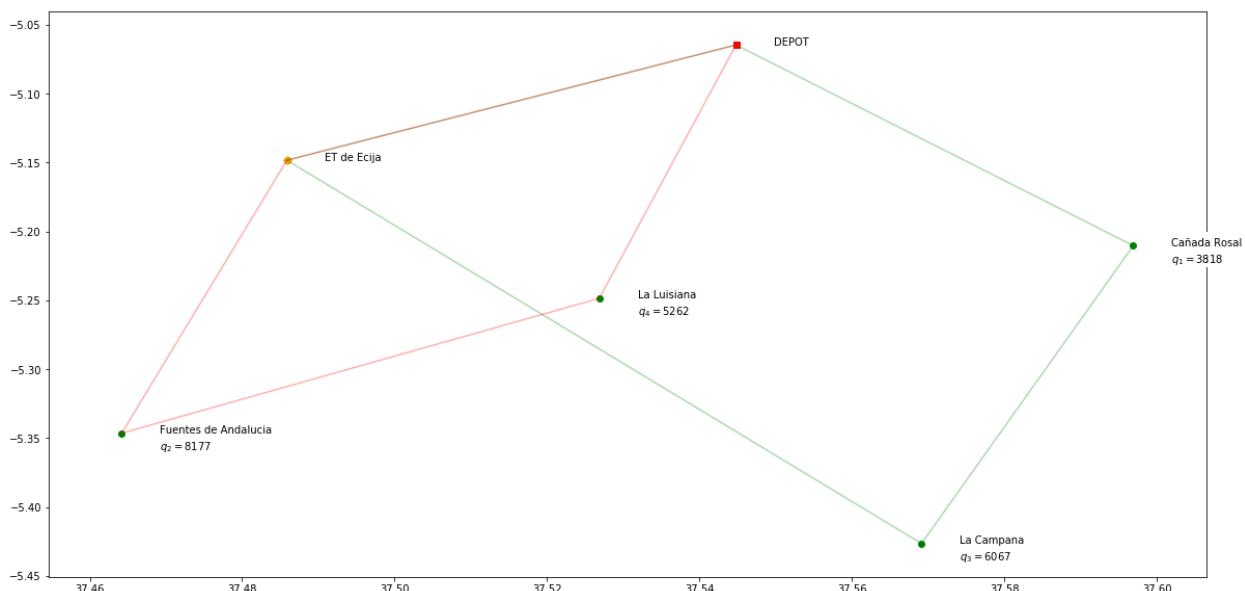


Figura 5.11. UGR nº6 – Gráfica con rutas

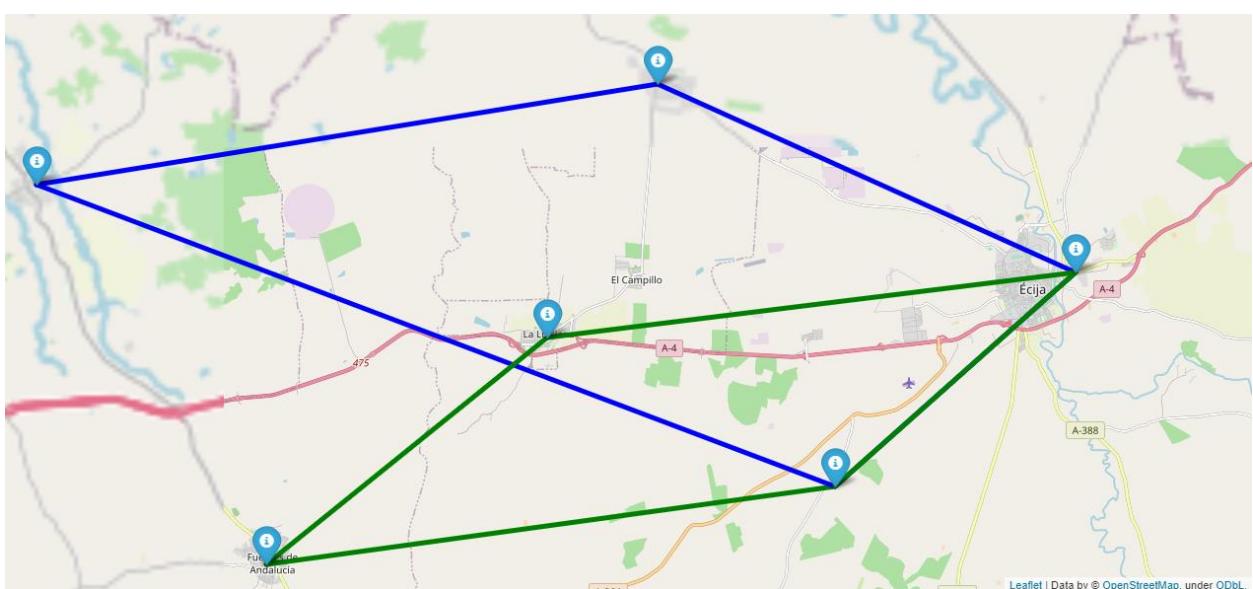


Figura 5.12. UGR nº6 – Mapa con rutas

### 5.3.7 UGR nº7 – Mancomunidad Intermunicipal Campiña 2000

Para este caso, el depósito coincide con la planta de destino, de manera que todos los vehículos comienzan y acaban su jornada en el mismo punto. Un total de nueve camiones de recogida dan servicio a los siete municipios que conforman esta mancomunidad, cuya producción de residuos estimada es de 122378 kg diarios. En la solución óptima se recorren un total de **445,6 km**. A continuación, se detallan los recorridos intermunicipales realizados por cada vehículo:

- **Ruta Vehículo 1:** DEPOT, Morón de la Frontera, PT Campiña 2000 (DEPOT)
- **Ruta Vehículo 2:** DEPOT, Morón de la Frontera, PT Campiña 2000 (DEPOT)
- **Ruta Vehículo 3:** DEPOT, Osuna, La Puebla de Cazalla, PT Campiña 2000 (DEPOT)
- **Ruta Vehículo 4:** DEPOT, Morón de la Frontera, La Puebla de Cazalla, PT Campiña 2000 (DEPOT)
- **Ruta Vehículo 5:** DEPOT, Arahal, PT Campiña 2000 (DEPOT)
- **Ruta Vehículo 6:** DEPOT, Paradas, Marchena, PT Campiña 2000 (DEPOT)
- **Ruta Vehículo 7:** DEPOT, Osuna, La Lanteljuela, PT Campiña 2000 (DEPOT)
- **Ruta Vehículo 8:** DEPOT, Arahal, Paradas, PT Campiña 2000 (DEPOT)
- **Ruta Vehículo 9:** DEPOT, Marchena, PT Campiña 2000 (DEPOT)

Del mismo modo que se ha venido haciendo con el resto de las mancomunidades, se presentan los datos obtenidos del modelo en la Tabla 5.16 y la representación gráfica del problema de enrutamiento en las Figura 5.13y la Figura 5.14.

Vehículo	c_veh (kg)	d_veh (km)	t_veh (h)	d_ext (km)	d_int (km)	t_ext (h)	t_int (h)	t_con (h)	t_vac (h)
1	14000	39,7	6,9	29,2	10,5	0,58	0,42	5,4	0,5
2	14000	39,7	6,9	29,2	10,5	0,58	0,42	5,4	0,5
3	14000	66,1	7,6	48,1	18	0,96	0,72	5,4	0,5
4	13332	53,3	7	36,8	16,5	0,74	0,66	5,1	0,5
5	14000	40	6,9	31	9	0,62	0,36	5,4	0,5
6	11046	48,3	6	31,8	16,5	0,64	0,66	4,3	0,5
7	14000	77,4	7,8	59,4	18	1,2	0,72	5,4	0,5
8	14000	49,9	7,2	33,4	16,5	0,67	0,66	5,4	0,5
9	14000	31,2	6,7	22,2	9	0,44	0,36	5,4	0,5

Tabla 5.16. UGR nº7 – Resultados

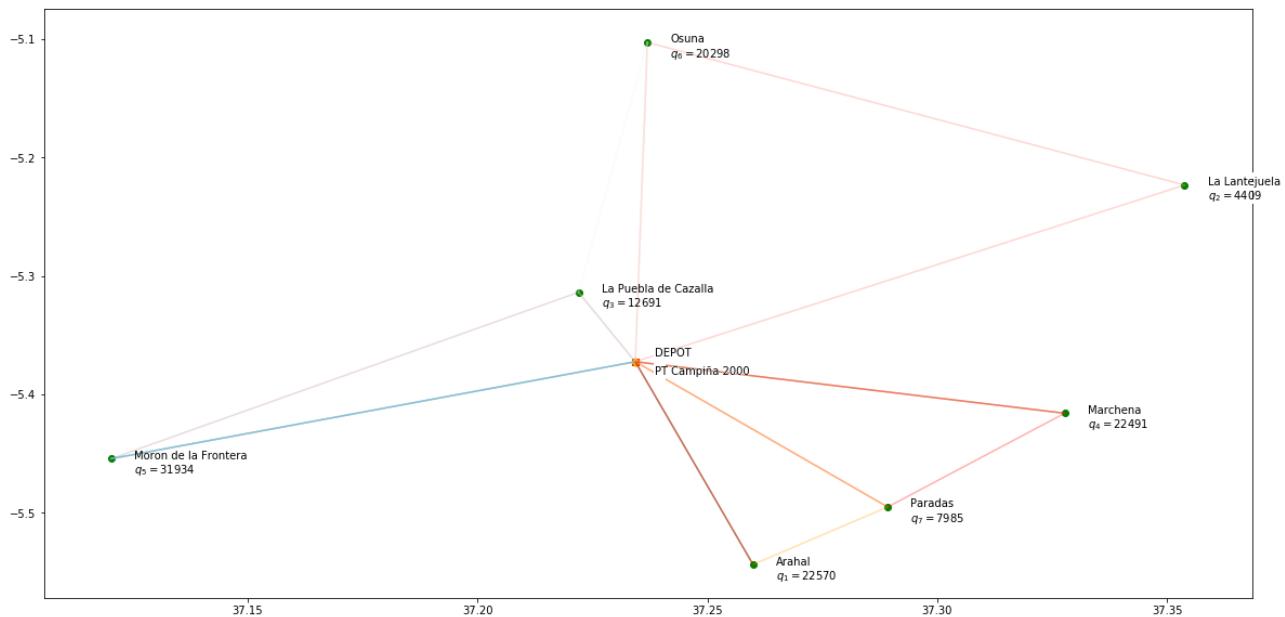


Figura 5.13. UGR n°7 – Gráfica con rutas

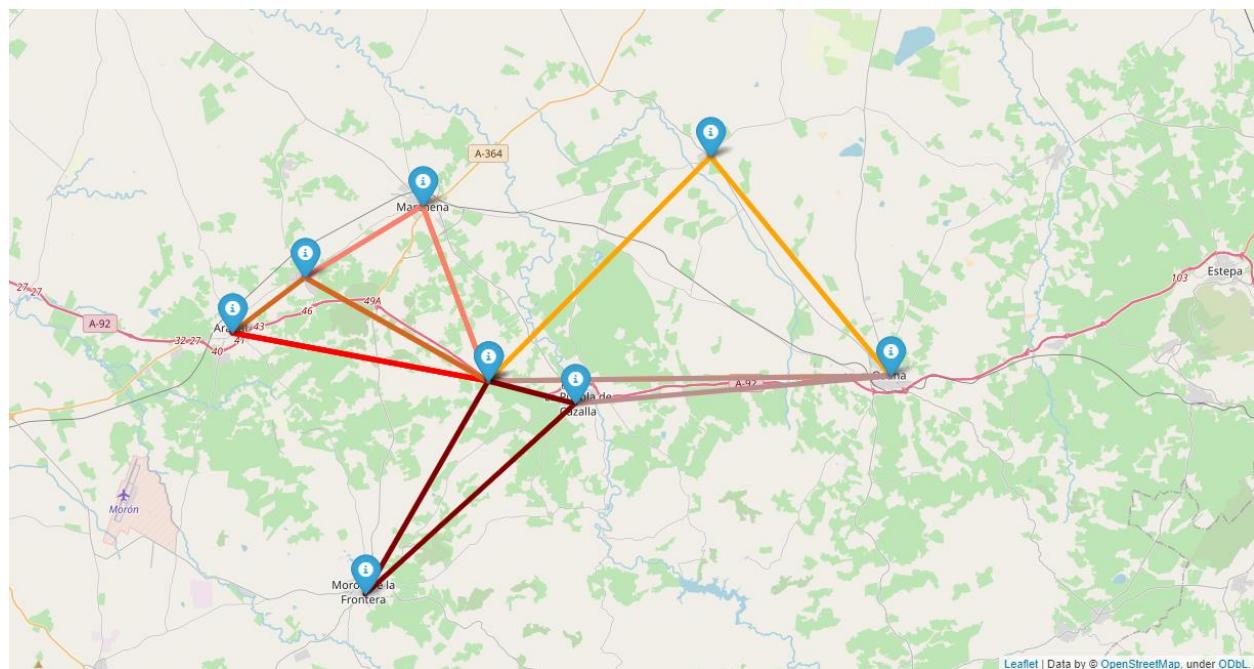


Figura 5.14. UGR n°7 – Mapa con rutas

# 6 CONCLUSIONES

---

Para finalizar la memoria de este trabajo, se recogen en este apartado las conclusiones tras analizar los resultados de la sección anterior. En primer lugar, se valora el cumplimiento de los objetivos marcados en el Capítulo 1. Posteriormente, se mencionan puntos de mejora del estudio, así como líneas de investigación futuras que pueden surgir en base al análisis y el estudio aquí realizado.

## 6.1 Valoración de resultados

El objetivo general planteado al comienzo de este proyecto consistía en elaborar una propuesta para la recogida conjunta de RSU en las mancomunidades que conformaban la Provincia de Sevilla. Esta propuesta se ha detallado en la Sección 5, donde para cada una de las UGRs se han presentado una serie de rutas que tendrían que recorrer las flotas de vehículos para cumplir con la demanda.

Como se ha venido comentando a lo largo de toda la memoria, la caracterización exhaustiva del problema ha permitido limitar el espacio de búsqueda de las soluciones. Sin embargo, esta misma caracterización junto a la organización del territorio por mancomunidades no ha permitido la interacción entre algunos municipios vecinos. Por este motivo, la creación de un órgano de gestión único para toda la provincia podría ser muy ventajoso ya que tendría la capacidad de reorganizar los municipios a favor de la eficiencia del servicio de recogida.

Atendiendo a los resultados del modelo aplicado, se observa que el aprovechamiento de la capacidad de los vehículos es en torno al 85%, lo que indica que la suposición del dimensionamiento de la flota basada en la generación de residuos ha sido acertada. Por otro lado, si se observa que el valor de la distancia en aquellas mancomunidades donde los municipios están muy dispersos, como ocurre por ejemplo con la Mancomunidad de los Alcores, es muy similar a la distancia recorrida por la flota de otras mancomunidades que atienden a bastantes más municipios, pero están más concentrados, como es el caso de la Mancomunidad del Guadalquivir.

Debido a la alta generación de residuos de algunos municipios es cierto que muchos vehículos visitan únicamente una localidad, puesto que alcanzan en una sola visita el total de su capacidad. Sin embargo, donde se aprecia una verdadera recogida conjunta es en núcleos de población relativamente cercanos entre sí, con una producción no muy alta de residuos. Llegando un solo vehículo a recoger basura hasta en cinco municipios distintos. Este aprovechamiento es el que se buscaba al comienzo de la investigación.

Por otro lado, la aproximación del recorrido interno de las poblaciones, realizada a través del factor de concentración y del perímetro no ha sido muy acertada, puesto que los valores obtenidos han sido demasiado bajos. Esto ha provocado que hubiera más tiempo disponible para la recogida de contenedores, que como se observa en todas las mancomunidades ha ocupado la mayoría del tiempo de la jornada establecida. De manera que la poca cantidad de kilómetros recorrido en el interior de las poblaciones se ha compensado con la contabilización del tiempo de recogida de contenedores.

## 6.2 Trabajos futuros

El emplazamiento óptimo de las bases de vehículos para cada mancomunidad podría considerarse como una posible mejora en la Provincia de Sevilla. Creando incluso más de una base en cada mancomunidad. Por otro lado, la distancia entre municipios es uno de los datos más relevantes e influyentes en el proceso de optimización. Por este motivo, el uso de una matriz de distancias reales, en vez de utilizar distancias geométricas entre coordenadas, ofrecería unos resultados muy valiosos, puesto que tendría en cuenta el recorrido realizado por la red de carreteras disponibles, no mediante las líneas rectas imaginarias que se han utilizado en este proyecto.

El desarrollo de técnicas como la recopilación y análisis de datos abre un gran abanico de posibilidades en problemas de este tipo, donde la información de la producción se utiliza como dato de entrada. Otras tecnologías relacionadas con la conectividad entre los elementos de una misma red organizativa están permitiendo tener un control detallado de las distancias recorridas por los vehículos, así como estimaciones muy precisas sobre cuánto tiempo se tarda en recorrer un determinado trayecto, gracias a los sistemas de geolocalización e información del tráfico en tiempo real.

En este trabajo, la modelización del problema se ha realizado como un problema estático tomando datos históricos de la producción de residuos y utilizándolos para resolver el modelo. Sin embargo, resulta muy interesante adaptar este planteamiento a un modelo dinámico, recibiendo información en tiempo real de la cantidad de basura depositada en cada contenedor de cada municipio. Estos datos se entregarían al modelo y se realizaría la simulación, permitiendo destinar los recursos disponibles a aquellos lugares donde realmente se necesitan. Ya existen evidencias de estos trabajos en algunos estudios como los realizados por (Grosso, 2017).

# REFERENCIAS

---

- Altinkemer, K., & Gavish, B. (1991). Parallel Savings Based Heuristics for the Delivery Problem. *Operations Research*, 39(3), 456–469. <https://doi.org/10.1287/opre.39.3.456>
- Anaconda Software Distribution. (2018). *Jupyter Notebook* (5.5.0). <https://anaconda.com>
- Anbuudayasanakar, S. P., Ganesh, K., & Mohapatra, S. (2014). Models for practical routing problems in logistics: Design and practices. In *Models for Practical Routing Problems in Logistics: Design and Practices* (Vol. 9783319050). <https://doi.org/10.1007/978-3-319-05035-5>
- Augerat, P., Belenguer, J. M., Benavent, E., Corberán, A., Naddef, D., & Rinaldi, G. (1995). *Computational results with a branch and cut code for the capacitated vehicle routing problem*.
- Baldacci, R., Christofides, N., & Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2), 351–385. <https://doi.org/10.1007/s10107-007-0178-5>
- Baldacci, R., Hadjiconstantinou, E., & Mingozzi, A. (2004). An Exact Algorithm for the Capacitated Vehicle Routing Problem Based on a Two-Commodity Network Flow Formulation. *Operations Research*, 52(5), 723–738. <https://search.proquest.com/docview/219151993?accountid=14744>
- Baldacci, R., Mingozzi, A., & Roberti, R. (2012). New State-Space Relaxations for Solving the Traveling Salesman Problem with Time Windows. *INFORMS Journal on Computing*, 24, 356–371. <https://doi.org/10.1287/ijoc.1110.0456>
- Baldacci, R., Toth, P., & Vigo, D. (2007). Recent advances in vehicle routing exact algorithms. *4or*, 5(4), 269–298. <https://doi.org/10.1007/s10288-007-0063-3>
- Baldacci, R., Toth, P., & Vigo, D. (2010). Exact algorithms for routing problems under vehicle capacity constraints. *Annals of Operations Research*, 175(1), 213–245. <https://doi.org/10.1007/s10479-009-0650-0>
- Beasley, J. (1983). Route First – Cluster Second Methods for Vehicle Routing. *Omega - International Journal of Management Science*, 11(4), 403–408.
- Benhida, S., & Mir, A. (2018). Generating subtour elimination constraints for the Traveling Salesman Problem. *IOSR Journal of Engineering*, 08(7), 17–21. [https://www.iosrjen.org/Papers/vol8\\_issue7/Version-1/D0807011721.pdf](https://www.iosrjen.org/Papers/vol8_issue7/Version-1/D0807011721.pdf)
- Booker, L., Forrest, S., Mitchell, M., & Riolo, R. (2005). *Perspectives on Adaptation in Natural and Artificial Systems : Essays in Honor of John Holland*. Oxford University Press, Incorporated. <http://ebookcentral.proquest.com/lib/uses/detail.action?docID=422378>
- Bramel, J., & Simchi - Levi, D. (1995). A Location Based Heuristic for General Routing Problems. *Operations Research*, 43(4), 649–660. <https://doi.org/10.1287/opre.43.4.649>
- Caballero, L. (2017). *Diseño y evaluación de algoritmos para VRP* [Universidad Politécnica de Cartagena]. <https://repositorio.upct.es/bitstream/handle/10317/6183/tfm-cab-dis.pdf?sequence=1&isAllowed=y>
- Christofides, N. (1976). The vehicle routing problem. *Revue Française d'automatique, Informatique, Recherche Opérationnelle. Recherche Opérationnelle*, 10(V1), 55–70.
- Christofides, N., Mingozzi, A., & Toth, P. (1981a). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20(1), 255–282. <https://doi.org/10.1007/BF01589353>
- Christofides, N., Mingozzi, A., & Toth, P. (1981b). State-space relaxation procedures for the computation of bounds to routing problems. *Networks*, 11(2), 145–164. <https://doi.org/10.1002/net.3230110207>

- Clarke, G., & Wright, J. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12(4), 568–581. <https://doi.org/10.1287/opre.12.4.568>
- Cordeau, J.-F., Laporte, G., & Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *The Journal of the Operational Research Society*, 52(8), 928–936. <https://search.proquest.com/docview/231348385?accountid=14744>
- Correa, S. (2019). Cplex - Python. 20 de Enero de 2019. [https://www.youtube.com/playlist?list=PL\\_wz\\_RHE6pEYEJO-vDNwHi2F7k-WNgOfQ](https://www.youtube.com/playlist?list=PL_wz_RHE6pEYEJO-vDNwHi2F7k-WNgOfQ)
- Dantzig, G. B., & Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, 6(1), 80–91. <https://doi.org/10.1287/mnsc.6.1.80>
- Davendra, D. (2010). Traveling Salesman Problem , Theory and Applications. In *An Efficient Solving the Travelling Salesman Problem : Global Optimization of Neural Networks by Using Hybrid Method*. <https://doi.org/10.5772/547>
- Desrochers, M., & Laporte, G. (1991). Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters*, 10(1), 27–36. [https://doi.org/10.1016/0167-6377\(91\)90083-2](https://doi.org/10.1016/0167-6377(91)90083-2)
- Díaz, J. (2017). *Curso Python desde 0*. Píldoras Informáticas, 24 de Enero de 2017. <https://www.youtube.com/playlist?list=PLU8oAlHdN5BlvPxziopYZRd55pdqFwkeS>
- Diputación de Sevilla. (2019). *Plan de Residuos no Peligrosos de la Provincia de Sevilla (2020-2035)*. IDOM Consulting, Engineering, Architecture, S.A.U. [https://www.dipusevilla.es/export/sites/diputacion-sevilla-corporativo/planderesiduos/\\_galleries/Plan-de-residuos/21064\\_13052019\\_Version-preliminar-del-Plan\\_v2\\_COMP.pdf](https://www.dipusevilla.es/export/sites/diputacion-sevilla-corporativo/planderesiduos/_galleries/Plan-de-residuos/21064_13052019_Version-preliminar-del-Plan_v2_COMP.pdf)
- Dorigo, M., & Stützle, T. (2004). *Ant Colony Optimization*. MIT Press. <http://ebookcentral.proquest.com/lib/uses/detail.action?docID=3338887>
- Doyuran, T., & Çatay, B. (2011). A robust enhancement to the Clarke–Wright savings algorithm. *Journal of the Operational Research Society*, 62(1), 223–231. <https://doi.org/10.1057/jors.2009.176>
- Dueck, G., & Scheuer, T. (1990). Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90(1), 161–175. [https://doi.org/https://doi.org/10.1016/0021-9991\(90\)90201-B](https://doi.org/https://doi.org/10.1016/0021-9991(90)90201-B)
- ECOEMBES. (2018). *ECOEMBES*. <https://www.ecoembes.com/es>
- Ergun, Ö. (2001). *New neighborhood search algorithms based on exponentially large neighborhoods*. Massachusetts Institute of Technology (MIT).
- España. (2010). *Ley 5/2010, de 11 de junio, de Autonomía Local de Andalucía*. Boletín Oficial del Estado, núm. 174 de 19 de julio de 2010, pp. 63395 a 63451.
- Fisher, M., & Ramchandran, J. (1981). A Generalized Assignment Heuristic for Vehicle Routing. *Networks*, 11(2), 109–124.
- Font, J., & Parrado Diez, S. (2000). *Eligiendo socios en la Administración municipal española: los consorcios y las mancomunidades*. Cuadernos de Gobierno y de Administración, núm. 3. [https://www.academia.edu/2467222/Eligiendo\\_socios\\_en\\_la\\_Administraci\\_n\\_municipal\\_espaola\\_los\\_consorcios\\_y\\_las\\_mancomunidades](https://www.academia.edu/2467222/Eligiendo_socios_en_la_Administraci_n_municipal_espaola_los_consorcios_y_las_mancomunidades)
- Fukasawa, R., Longo, H., Lysgaard, J., Aragão, M. P. de, Reis, M., Uchoa, E., & Werneck, R. F. (2006). Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem. *Mathematical Programming*, 106(3), 491–511. <https://doi.org/10.1007/s10107-005-0644-x>
- Gendreau, M., Hertz, A., & Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10), 1276. <https://search.proquest.com/docview/213236182?accountid=14744>
- Geoffrion, A. M. (1976). *A Priori Error Bounds for Procurement Commodity Aggregation in Logistics Planning Models* (U. S. D. of the N. C. U. L. O. S. A. W. M. S. INST (ed.)). Defense Technical Information Center. <https://doi.org/10.21236/ADA030007>
- Gillett, B. E., & Miller, L. R. (1974). Heuristic Algorithm for Vehicle Dispatch Problem. *Operations*

*Research*, 22(2), 340–349.

- Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8, 156–166. <https://doi.org/10.1111/j.1540-5915.1977.tb01074.x>
- Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5), 533–549. [https://doi.org/https://doi.org/10.1016/0305-0548\(86\)90048-1](https://doi.org/10.1016/0305-0548(86)90048-1)
- Gobierno de España. (2015). *Plan Estatal Marco de Gestión de Residuos PEMAR (2016-2022)* (pp. 1–182). Ministerio de Agricultura, Alimentación y Medio Ambiente. <https://www.miteco.gob.es/es/calidad-y-evaluacion-ambiental/planes-y-estrategias/Planes-y-Programas.aspx>
- Gómez, J. R. (2010). *Diseño de un Sistema de Recogida de Residuos Urbanos: Enfoque Multiojetivo y Uso de Metaheurísticos*. <https://riubu.ubu.es/handle/10259/105>
- Google. (2020). *Provincia de Sevilla*. <https://www.google.com/maps/place/Sevilla/@37.3753501,-6.0250983,12z/data=!3m1!4b1!4m5!3m4!1s0xd126c1114be6291:0x34f018621cf5648!8m2!3d37.3890924!4d-5.9844589>
- Gorostegui, L. (2011). *Modelado, Optimización Y Planificación de una Red de Distribución de Gas*. Universidad Complutense de Madrid.
- Grosso, R. (2017). *Optimización Sostenible y Gestión Eficiente de Flotas Urbanas Director : Jesús Muñozuri Sanz*. 172. [https://idus.us.es/bitstream/handle/11441/60383/Tesis RGV\\_V3.1\\_2017.pdf](https://idus.us.es/bitstream/handle/11441/60383/Tesis RGV_V3.1_2017.pdf)
- Hoffman, K., Padberg, M., & Rinaldi, G. (2016). *Traveling Salesman Problem (TSP)*. [https://doi.org/10.1007/978-3-319-23519-6\\_1406-2](https://doi.org/10.1007/978-3-319-23519-6_1406-2)
- IBM Corp. (2017). *IBM ILOG CPLEX Optimization Studio: CPLEX 12.7 User's Manual* (p. 562). [https://www.ibm.com/support/knowledgecenter/SSA5P\\_12.7.1/ilog.odms.studio.help/Optimization\\_Studio/topics/COS\\_home.html](https://www.ibm.com/support/knowledgecenter/SSA5P_12.7.1/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html)
- IBM ILOG. (2019). *CPLEX Studio IDE* (12.9.0).
- Instituto Nacional de Estadística. (2020). *Instituto Nacional de Estadística*. <https://www.ine.es/>
- Johnson, D. S., & McGeoch, L. A. (2003). 8. The traveling salesman problem: a case study. In *Local Search in Combinatorial Optimization* (pp. 215–310). Princeton University Press. <https://doi.org/https://doi.org/10.1515/9780691187563-011>
- Junta de Andalucía. (1999). *Plan Director Provincial de Residuos Sólidos Urbanos de Andalucía (1997-2002)* (Vol. 134). Consejería de Medio Ambiente, Boletín Oficial de la Junta de Andalucía, 18 de noviembre de 1999, núm. 134. <https://www.juntadeandalucia.es/boja/1999/134/boletin.134.pdf>
- Junta de Andalucía. (2010). *Plan Director de Residuos No Peligrosos de Andalucía (2010-2019)* (pp. 114–167). Consejería de Medio Ambiente, Boletín Oficial de la Junta de Andalucía, 25 de noviembre de 2010, núm. 231. [http://www.juntadeandalucia.es/medioambiente/portal\\_web/web/temas\\_ambientales/residuos/residuos\\_urbanos/2010\\_2019\\_plan\\_rnpa.pdf](http://www.juntadeandalucia.es/medioambiente/portal_web/web/temas_ambientales/residuos/residuos_urbanos/2010_2019_plan_rnpa.pdf)
- Kirkpatrick, S., Gelatt, C., & Vecchi, M. (1983). Optimization by Simulated Annealing. *Science (New York, N.Y.)*, 220, 671–680. <https://doi.org/10.1126/science.220.4598.671>
- Laporte, G. (1992). The Vehicle Routin Problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2), 345–358. [https://doi.org/10.1016/0377-2217\(92\)90138-Y](https://doi.org/10.1016/0377-2217(92)90138-Y)
- Laporte, G. (2006). What You Should Know about the Vehicle Routing Problem. *Naval Research Logistics*, 54(April 2007), 811–819. <https://doi.org/10.1002/nav>
- Laporte, G., & Nobert, Y. (1987). Exact Algorithms for the Vehicle Routing Problem. *North-Holland Mathematics Studies*, 132(C), 147–184. [https://doi.org/10.1016/S0304-0208\(08\)73235-3](https://doi.org/10.1016/S0304-0208(08)73235-3)
- Laporte, G., Nobert, Y., & Taillefer, S. (1987). A branch-and-bound algorithm for the asymmetrical distance-constrained vehicle routing problem. *Mathematical Modelling*, 9(12), 857–868. [https://doi.org/https://doi.org/10.1016/0270-0255\(87\)90004-2](https://doi.org/https://doi.org/10.1016/0270-0255(87)90004-2)

- Lin, S. (1965). Computer Solutions of the Traveling Salesman Problem. *Bell System Technical Journal*, 44(10), 2245–2269. <https://doi.org/10.1002/j.1538-7305.1965.tb04146.x>
- Little, J. D. C., Murty, K. G., Sweeney, D. W., & Karel, C. (1963). An Algorithm for the Traveling Salesman Problem. *Operations Research*, 11(6), 972–989.
- Lysgaard, J., Letchford, A. N., & Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2), 423–445. <https://doi.org/10.1007/s10107-003-0481-8>
- Machuca de Pina, J. M., Dorin, M., & García Yi, A. I. (2018). Evaluación experimental de un modelo de programación lineal para el problema de ruteo de vehículos (VRP). *Interfases*, 011, 103–117. <https://doi.org/10.26439/interfases2018.n011.2956>
- Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11), 1097–1100. [https://doi.org/10.1016/S0305-0548\(97\)00031-2](https://doi.org/10.1016/S0305-0548(97)00031-2)
- Moscato, P., & Cotta, C. (2003). Una Introducción a los Algoritmos Meméticos. *Inteligencia Artificial: Revista Iberoamericana de Inteligencia Artificial*, ISSN 1137-3601, Nº. 19, 2003, Pags. 131-148.
- Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8), 2403–2435. <https://doi.org/10.1016/j.cor.2005.09.012>
- Powell, W. B. (2011). *Approximate Dynamic Programming : Solving the Curses of Dimensionality*. John Wiley & Sons, Incorporated. <http://ebookcentral.proquest.com/lib/uses/detail.action?docID=697550>
- Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12), 1985–2002. [https://doi.org/10.1016/S0305-0548\(03\)00158-8](https://doi.org/10.1016/S0305-0548(03)00158-8)
- Python Software Foundation. (2018). *Python* (3.6.10). <https://www.python.org/>
- Radharamanan, R., & Choi, L. I. (1986). A branch and bound algorithm for the travelling salesman and the transportation routing problems. *Computers & Industrial Engineering*, 11(1), 236–240. [https://doi.org/10.1016/0360-8352\(86\)90085-9](https://doi.org/10.1016/0360-8352(86)90085-9)
- Ralphs, T. K. (2003). Parallel branch and cut for capacitated vehicle routing. *Parallel Computing*, 29(5), 607–629. [https://doi.org/10.1016/S0167-8191\(03\)00045-0](https://doi.org/10.1016/S0167-8191(03)00045-0)
- Real Academia Española. (2020). *Diccionario de la lengua española*, 23.<sup>a</sup> ed. [versión 23.3 en línea] (16 de mayo de 2020).
- Renaud, J., Boctor, F., & Laporte, G. (1996). A Fast Composite Heuristic for the Symmetric Traveling Salesman Problem. *INFORMS Journal on Computing*, 8, 134–143. <https://doi.org/10.1287/ijoc.8.2.134>
- Rochat, Y., & Taillard, É. (1995). Probabilistic Diversification and Intensification. *Journal of Heuristics*, 1, 147–167. <https://doi.org/10.1007/BF02430370>
- Rosenkrantz, D., Stearns, R., & II, P. (1977). An Analysis of Several Heuristics for the Traveling Salesman Problem. *SIAM J. Comput.*, 6, 563–581. <https://doi.org/10.1137/0206041>
- Salvador, C. (2012). *Planificación y Optimización de Flotas de Vehículos para la Recogida de Residuos Urbanos*. <http://www.fdi.ucm.es/profesor/jjruz/WebProyectos/CarlosVazquez/MemoriaCarlosSalvador.pdf>
- Serrano, A. (2015). *Análisis Metaheurístico en la Logística Inversa de Residuos* [Universidad de Málaga]. <https://riuma.uma.es/xmlui/handle/10630/12248>
- Silver, E. A., Victor, R., Vidal, V., & de Werra, D. (1980). A tutorial on heuristic methods. *European Journal of Operational Research*, 5(3), 153–162. [https://doi.org/10.1016/0377-2217\(80\)90084-3](https://doi.org/10.1016/0377-2217(80)90084-3)
- Sörensen, K., & Glover, F. W. (2013). Metaheuristics. In S. I. Gass & M. C. Fu (Eds.), *Encyclopedia of Operations Research and Management Science* (pp. 960–970). Springer US. [https://doi.org/10.1007/978-1-4419-1153-7\\_1167](https://doi.org/10.1007/978-1-4419-1153-7_1167)
- Tarantilis, C. D., & Kiranoudis, C. T. (2002). BoneRoute: An Adaptive Memory-Based Method for Effective Fleet Management. *Annals of Operations Research*, 115(1), 227–241.

<https://doi.org/10.1023/A:1021157406318>

Toth, P., & Vigo, D. (2000). An Overview of Vehicle Routing Problems. *Discrete Mathematics and Applications. The Vehicle Routing Problem*, SIAM. p. 1-26.

Toth, P., & Vigo, D. (2003). The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *INFORMS Journal on Computing*, 15(4), 333–346.  
<https://search.proquest.com/docview/200521672?accountid=14744>

Van Breedam, A. (1995). Improvement heuristics for the Vehicle Routing Problem based on simulated annealing. *European Journal of Operational Research*, 86(3), 480–490.  
[https://doi.org/https://doi.org/10.1016/0377-2217\(94\)00064-J](https://doi.org/https://doi.org/10.1016/0377-2217(94)00064-J)

White, D. J. (1969). Problems Involving Infrequent but Significant Contingencies. *Journal of the Operational Research Society: Special Issue*, 20(1), 45–57. <https://doi.org/10.1057/jors.1969.28>



# ANEXO I. CÓDIGO

---

## Modelo para UNA planta de destino

```
#Se carga la Estructura de Datos desde la hoja de Excel
df=pd.read_excel(open("Direccion de Memoria de la Estructura de Datos
correspondiente","rb"),sheet_name="Hoja2",converters={"n°":int},
index_col=[0])

#Se definen los vehiculos
m=29
Vehiculos=[x for x in range(1,m+1)]

#Se definen los contenedores
cont=10225

#Se definen los municipios
municipios=[]
#Lista con los nombres
for i in range(1,len(df)-1):
    municipios.append(df.iloc[i][0])
#Lista con la numeracion
n=[i for i in range(1,len(df)-1)]
n_veh=[(j,k) for j in n for k in range(1,m+1)]

#Se definen las Plantas de Tratamiento
estaciones=[]
#Lista con los nombres
for i in range(len(df)-1,len(df)):
    estaciones.append(df.iloc[i][0])
#Lista con la numeracion
index_estaciones=[i for i in range(len(df)-1,len(df))]
e1=index_estaciones[0]

#Se definen los nodos:
#Depot+municipios
depot=[df.iloc[0][0]]+municipios #Lista con los nombres
index_depot=[0]+n #Lista con la numeracion
#Municipios+estaciones
transf=municipios+estaciones #Lista con los nombres
index_transf=n+index_estaciones #Lista con la numeracion
#Depot+municipios+estaciones
nodos=[df.iloc[0][0]]+municipios+estaciones #Lista con los nombres
index_nodos=[0]+n+index_estaciones #Lista con la numeracion

#Se crean los arcos
#Diccionario con la numeracion
index_arcos={(i,j) for i in index_nodos for j in index_nodos if i!=j}
index_arcos_veh={(i,j,k) for k in range(1,m+1) for i in index_nodos
for j in index_nodos if i!=j}
index_arcos_z={(i,j,k) for k in range(1,m+1) for i in index_depot for
j in n if i!=j}
```

```

#PARAMETROS
Q=14000 #Capacidad de los camiones
U=100000 #Limite superior
Vc=50 #Velocidad en carretera
Vp=25 #Velocidad en el interior de las poblaciones
Tr=0.015 #Tiempo de recogida de un contenedor
Tv=0.5 #Tiempo de descarga

#Se define la produccion de residuos
q_total=0
for i in range(len(municipios)):
    q_total=q_total+df.iloc[i+1][2]
q={n[i]:df.iloc[i+1][2] for i in range(len(municipios))}

#Se definen factor de concentracion y perimetro
fcon=[]
for i in range(1,len(df)):
    fcon.append(df.iloc[i][5])
p=[]
for i in range(1,len(df)):
    p.append(df.iloc[i][6])

#Se definen las coordenadas
coord_x=[]
for i in range(len(df)):
    coord_x.append(df.iloc[i][3])
coord_y=[]
for i in range(len(df)):
    coord_y.append(df.iloc[i][4])

#Se calculan las distancias entre los municipios
c=[]
for n in range(len(df)):
    for i in range(len(df)):
        if n!=i:
            c.append(np.round(distance.distance((df.iloc[n][3],
df.iloc[n][4]),(df.iloc[i][3],df.iloc[i][4])).km, decimals=1))

#Se asocian las distancias a los arcos
c_arcos_veh={(i,j,k):np.round(distance.distance((df.iloc[i][3],df.iloc[i][4]),(df.iloc[j][3],df.iloc[j][4])).km,decimals=1) for k in
range(1,m+1) for i in index_nodos for j in index_nodos if i!=j}

#CREAMOS EL MODELO
mdl=Model("CVRP")

#Variables de decision
x=mdl.binary_var_dict(index_arcos_veh,name="x")
u=mdl.continuous_var_dict(index_nodos,name="u")
y=mdl.continuous_var_dict(n_veh,name="y")
z=mdl.continuous_var_dict(index_arcos_z,name="z")

#Variables auxiliares
c_veh=mdl.continuous_var_dict(Vehiculos, ub=Q, name="c_veh")
t_veh=mdl.continuous_var_dict(Vehiculos,ub=8, name="t_veh")
t_ext=mdl.continuous_var_dict(Vehiculos, name="t_ext")

```

```

t_con=mdl.continuous_var_dict(Vehiculos,name="t_con")
t_vac=mdl.continuous_var_dict(Vehiculos,name="t_vac")
t_int=mdl.continuous_var_dict(Vehiculos,name="t_int")
d_int=mdl.continuous_var_dict(Vehiculos,name="d_int")
d_ext=mdl.continuous_var_dict(Vehiculos,name="d_ext")

#Funcion Objetivo - Expresion 4.32
    mdl.minimize(mdl.sum(c_arcos_veh[i,j,k]*x[i,j,k]+x[i,j,k]*p[j-1]*
fcon[j-1] for i,j,k in index_arcos_veh))

#Restriccion 4.33
    mdl.add_constraints(mdl.sum(x[i,j,k] for j in index_nodos if i!=j)<=1
for i in index_nodos for k in range(1,m+1))

#Restriccion 4.34
    mdl.add_constraints(mdl.sum(x[i,j,k] for i in index_nodos if i!=j)<=1
for j in index_nodos for k in range(1,m+1))

#Restriccion 4.35
    mdl.add_constraints(u[i]-u[j]+len(n)*x[i,j,k]<=len(n)-1 for i in n
for j in n for k in range(1,m+1) if i!=j)

#Restriccion 4.36
    mdl.add_constraints(mdl.sum(x[i,s,k] for i in index_depot if i!=s) ==
mdl.sum(x[s,j,k] for j in index_transf if j!=s) for s in n for k in
range(1,m+1))

#Restriccion 4.37
    mdl.add_constraints(mdl.sum(z[i,j,k] for i in index_depot for k in
range(1,m+1) if i!=j)==q[j] for j in n)

#Restriccion 4.38
    mdl.add_constraints(z[i,j,k]<=U*x[i,j,k] for i in index_depot for j
in n for k in range(1,m+1) if i!=j)

#Restriccion 4.39
    mdl.add_constraints(z[i,j,k] for i in index_depot if
i!=j)<=y[j,k] for j in n for k in range(1,m+1))

#Restriccion 4.40
    mdl.add_constraints(z[i,j,k]-U*x[i,j,k] for i in index_depot
if i!=j)>=y[j,k]-U for j in n for k in range(1,m+1))

#Restriccion 4.41
    mdl.add_constraints(mdl.sum(y[j,k] for k in range(1,m+1))==q[j] for j
in n)

#Restriccion 4.42
    mdl.add_constraints(mdl.sum(x[i,j,k] for i in index_nodos if
i!=j)<=y[j,k] for j in n for k in range(1,m+1))

#Restriccion 4.43
    mdl.add_constraints(mdl.sum(y[j,k] for j in n)==c_veh[k] for k in
range(1,m+1))

```

```

#Restriccion 4.44
mdl.add_constraints(mdl.sum(c_arco_veh[i,j,k]*x[i,j,k] for i in
index_nodos for j in index_nodos if i!=j) == d_ext[k] for k in
range(1,m+1))

#Restriccion 4.45
mdl.add_constraints(mdl.sum(x[i,j,k]*p[j-1]*fcon[j-1] for i in
index_nodos for j in n if i!=j)==d_int[k] for k in range(1,m+1))

#Restriccion 4.46
mdl.add_constraints(mdl.sum(c_arco_veh[i,j,k]*x[i,j,k]/Vc for i,j in
index_arco)==t_ext[k] for k in range(1,m+1))

#Restriccion 4.47
mdl.add_constraints(d_int[k]/VP==t_int[k] for k in range(1,m+1))

#Restriccion 4.48
mdl.add_constraints(mdl.sum(y[j,k]*(cont/q_total)*Tr for j in n) ==
t_con[k] for k in range(1,m+1))

#Restriccion 4.49
mdl.add_constraints(mdl.sum(x[i,e1,k] for i in n)*Tv==t_vac[k] for k
in range(1,m+1))

#Restriccion 4.50
mdl.add_constraints(t_veh[k]==t_ext[k]+t_con[k]+t_vac[k]+t_int[k] for
k in range(1,m+1))

#Restriccion 4.51
mdl.add_constraints(mdl.sum(x[i,e1,k] for i in n)==mdl.sum(x[0,j,k]
for j in n) for k in range(1,m+1))

#Restriccion 4.52
mdl.add_constraints(mdl.sum(x[i,e1,k] for i in n)==x[e1,0,k] for k in
range(1,m+1))

#Restriccion 4.53
mdl.add_constraints(x[e1,0,k]==1 for k in range(1,m+1))

#Restriccion 4.54
mdl.add_constraints(mdl.sum(x[e1,j,k] for j in n)==0 for k in
range(1,m+1))

#Restriccion 4.55
mdl.add_constraints(x[0,e1,k]==0 for k in range(1,m+1))

#Restriccion 4.56
mdl.add_constraint(mdl.sum(x[i,e1,k] for i in n for k in
range(1,m+1))>=1)

#Imprime el modelo
print(mdl.export_to_string())

#Resolucion
solucion=mdl.solve(log_output=True)
mdl.get_solve_status()
solucion.display()

```

## Modelo para DOS plantas de destino

```
#Se carga la Estructura de Datos desde la hoja de Excel
df=pd.read_excel(open("Direccion de Memoria de la Estructura de Datos correspondiente","rb"),sheet_name="Hoja2",converters={"n°":int},
index_col=[0])

#Se definen los vehiculos
m=25
Vehiculos=[x for x in range(1,m+1)]

#Se definen los contenedores
cont=6000

#Se definen los municipios
municipios=[]
#Lista con los nombres
for i in range(1,len(df)-1):
    municipios.append(df.iloc[i][0])
#Lista con la numeracion
n=[i for i in range(1,len(df)-1)]
n_veh=[(j,k) for j in n for k in range(1,m+1)]

#Se definen las Plantas de Tratamiento
estaciones=[]
#Lista con los nombres
for i in range(len(df)-1,len(df)):
    estaciones.append(df.iloc[i][0])
#Lista con la numeracion
index_estaciones=[i for i in range(len(df)-1,len(df))]
e1=index_estaciones [0]
e2=index_estaciones [1]

#Se definen los nodos:
#Depot+municipios
depot=[df.iloc[0][0]]+municipios #Lista con los nombres
index_depot=[0]+n #Lista con la numeracion
#Municipios+estaciones
transf=municipios+estaciones #Lista con los nombres
index_transf=n+index_estaciones #Lista con la numeracion
#Depot+municipios+estaciones
nodos=[df.iloc[0][0]]+municipios+estaciones #Lista con los nombres
index_nodos=[0]+n+index_estaciones #Lista con la numeracion

#Se crean los arcos
#Diccionario con la numeracion
index_arcos={(i,j) for i in index_nodos for j in index_nodos if i!=j}
index_arcos_veh={(i,j,k) for k in range(1,m+1) for i in index_nodos
for j in index_nodos if i!=j}
index_arcos_z={(i,j,k) for k in range(1,m+1) for i in index_depot for
j in n if i!=j}

#PARAMETROS
Q=14000 #Capacidad de los camiones
U=100000 #Limite superior
Vc=50 #Velocidad en carretera
Vp=25 #Velocidad en el interior de las poblaciones
```

```

Tr=0.015 #Tiempo de recogida de un contenedor
Tv=0.5 #Tiempo de descarga

#Se define la produccion de residuos
q_total=0
for i in range(len(municipios)):
    q_total=q_total+df.iloc[i+1][2]
q={n[i]:df.iloc[i+1][2] for i in range(len(municipios))}

#Se definen factor de concentracion y perimetro
fcon=[]
for i in range(1,len(df)):
    fcon.append(df.iloc[i][5])
p=[]
for i in range(1,len(df)):
    p.append(df.iloc[i][6])

#Se definen las coordenadas
coord_x=[]
for i in range(len(df)):
    coord_x.append(df.iloc[i][3])
coord_y=[]
for i in range(len(df)):
    coord_y.append(df.iloc[i][4])

#Se calculan las distancias entre los municipios
c=[]
for n in range(len(df)):
    for i in range(len(df)):
        if n!=i:
            c.append(np.round(distance.distance((df.iloc[n][3],
df.iloc[n][4]),(df.iloc[i][3],df.iloc[i][4])).km, decimals=1))

#Se asocian las distancias a los arcos
c_arcos_veh={(i,j,k):np.round(distance.distance((df.iloc[i][3],df.iloc[i][4]),(df.iloc[j][3],df.iloc[j][4])).km,decimals=1) for k in
range(1,m+1) for i in index_nodos for j in index_nodos if i!=j}

#CREAMOS EL MODELO
mdl=Model("CVRP")

#Variables de decision
x=mdl.binary_var_dict(index_arcos_veh,name="x")
u=mdl.continuous_var_dict(index_nodos,name="u")
y=mdl.continuous_var_dict(n_veh,name="y")
z=mdl.continuous_var_dict(index_arcos_z,name="z")

#Variables auxiliares
c_veh=mdl.continuous_var_dict(Vehiculos, ub=Q, name="c_veh")
t_veh=mdl.continuous_var_dict(Vehiculos,ub=8, name="t_veh")
t_ext=mdl.continuous_var_dict(Vehiculos, name="t_ext")
t_con=mdl.continuous_var_dict(Vehiculos, name="t_con")
t_vac=mdl.continuous_var_dict(Vehiculos, name="t_vac")
t_int=mdl.continuous_var_dict(Vehiculos, name="t_int")
d_int=mdl.continuous_var_dict(Vehiculos, name="d_int")
d_ext=mdl.continuous_var_dict(Vehiculos, name="d_ext")

```

```

#Funcion Objetivo - Expresion 4.64
  mdl.minimize(mdl.sum(c_arcos_veh[i,j,k]*x[i,j,k]+x[i,j,k]*p[j-1]*
fcon[j-1] for i,j,k in index_arcos_veh))

#Restriccion 4.65
  mdl.add_constraints(mdl.sum(x[i,j,k] for j in index_nodos if i!=j)<=1
for i in index_nodos for k in range(1,m+1))

#Restriccion 4.66
  mdl.add_constraints(mdl.sum(x[i,j,k] for i in index_nodos if i!=j)<=1
for j in index_nodos for k in range(1,m+1))

#Restriccion 4.67
  mdl.add_constraints(u[i]-u[j]+len(n)*x[i,j,k]<=len(n)-1 for i in n
for j in n for k in range(1,m+1) if i!=j)

#Restriccion 4.68
  mdl.add_constraints(mdl.sum(x[i,s,k] for i in index_depot if i!=s) ==
mdl.sum(x[s,j,k] for j in index_transf if j!=s) for s in n for k in
range(1,m+1))

#Restriccion 4.69
  mdl.add_constraints(mdl.sum(z[i,j,k] for i in index_depot for k in
range(1,m+1) if i!=j)==q[j] for j in n)

#Restriccion 4.70
  mdl.add_constraints(z[i,j,k]<=U*x[i,j,k] for i in index_depot for j
in n for k in range(1,m+1) if i!=j)

#Restriccion 4.71
  mdl.add_constraints(mdl.sum(z[i,j,k] for i in index_depot if
i!=j)<=y[j,k] for j in n for k in range(1,m+1))

#Restriccion 4.72
  mdl.add_constraints(mdl.sum(z[i,j,k]-U*x[i,j,k] for i in index_depot
if i!=j)>=y[j,k]-U for j in n for k in range(1,m+1))

#Restriccion 4.73
  mdl.add_constraints(mdl.sum(y[j,k] for k in range(1,m+1))==q[j] for j
in n)

#Restriccion 4.74
  mdl.add_constraints(mdl.sum(x[i,j,k] for i in index_nodos if
i!=j)<=y[j,k] for j in n for k in range(1,m+1))

#Restriccion 4.75
  mdl.add_constraints(mdl.sum(y[j,k] for j in n)==c_veh[k] for k in
range(1,m+1))

#Restriccion 4.76
  mdl.add_constraints(mdl.sum(c_arcos_veh[i,j,k]*x[i,j,k] for i in
index_nodos for j in index_nodos if i!=j) == d_ext[k] for k in
range(1,m+1))

#Restriccion 4.77
  mdl.add_constraints(mdl.sum(x[i,j,k]*p[j-1]*fcon[j-1] for i in
index_nodos for j in n if i!=j)==d_int[k] for k in range(1,m+1))

```

```
#Restriccion 4.78
mdl.add_constraints(mdl.sum(c_arco_veh[i,j,k]*x[i,j,k]/Vc for i,j in
index_arco)==t_ext[k] for k in range(1,m+1))

#Restriccion 4.79
mdl.add_constraints(d_int[k]/VP==t_int[k] for k in range(1,m+1))

#Restriccion 4.80
mdl.add_constraints(mdl.sum(y[j,k]*(cont/q_total)*Tr for j in n) ==
t_con[k] for k in range(1,m+1))

#Restriccion 4.81
mdl.add_constraints(mdl.sum(x[i,e1,k]+x[i,e2,k] for i in n)*Tv ==
t_vac[k] for k in range(1,m+1))

#Restriccion 4.82
mdl.add_constraints(t_veh[k]==t_ext[k]+t_con[k]+t_vac[k]+t_int[k] for
k in range(1,m+1))

#Restriccion 4.83
mdl.add_constraints(mdl.sum(x[i,e1,k]+x[i,e2,k] for i in n) ==
mdl.sum(x[0,j,k] for j in n) for k in range(1,m+1))

#Restriccion 4.84
mdl.add_constraints(mdl.sum(x[i,e,k] for i in n)==x[e,0,k] for e in
index_estaciones for k in range(1,m+1))

#Restriccion 4.85
mdl.add_constraints(x[e,0,k]+x[i,e2,k]==1 for k in range(1,m+1))

#Restriccion 4.86
mdl.add_constraints(mdl.sum(x[e,j,k] for j in n)==0 for e in
index_estaciones for k in range(1,m+1))

#Restriccion 4.87
mdl.add_constraints(x[0,e,k]==0 for e in index_estaciones for k in
range(1,m+1))

#Restriccion 4.88
mdl.add_constraint(mdl.sum(x[i,e,k] for e in index_estaciones for i
in n for k in range(1,m+1))>=1)

#Imprime el modelo
print(mdl.export_to_string())

#Resolucion
solucion=mdl.solve(log_output=True)
mdl.get_solve_status()
solucion.display()
```

## ANEXO II. ESTRUCTURAS DE DATOS

---

### UGR nº1 – Mancomunidad de los Alcores

Nº	Municipio	Prod. Residuos (kg/día)	Latitud	Longitud	Factor de concentración	Perímetro (km)
0	DEPOT		37,338909	-5,825272		
1	Carmona	31627	37,471119	-5,649731	2	34
2	Coripe	1387	36,97215	-5,440838	2	26
3	El Coronil	5261	37,078122	-5,633626	1,5	34
4	El Cuervo de Sevilla	9545	36,853042	-6,03879	1,5	20
5	El Viso del Alcor	21357	37,3876	-5,72229	1	17
6	Las Cabezas de San Juan	18199	36,983763	-5,936506	1,5	54
7	Lebrija	30511	36,918706	-6,076706	1,5	69
8	Los Molares	3858	37,155859	-5,719333	1,5	24
9	Mairena del Alcor	26106	37,374979	-5,746844	1,5	30
10	Montellano	7822	36,993573	-5,571085	1,5	39
11	Planta de Tratamiento Montemarta - Cónica		37,220464	-5,886575		

Tabla 0.1. UGR nº1 – Estructura de Datos

Parámetro	Descripción	Valor asignado
m	Número de vehículos	13
cont	Número de contenedores	4000

Tabla 0.2. UGR nº1 – Parámetros particulares

**UGR nº2 – Mancomunidad del Guadalquivir**

Nº	Municipio	Prod. Residuos (kg/día)	Latitud	Longitud	Factor de concentración	Perímetro (km)
0	DEPOT		37,378648	-6,1921109		
1	Albaida del Aljarafe	3783	37,4255633	-6,1652929	1,5	12
2	Almensilla	7195	37,3098113	-6,1136061	1	14
3	Aznalcázar	5427	37,3047112	-6,2495425	2	76
4	Aznalcóllar	7208	37,5191373	-6,2681327	2	51
5	Benacazón	8569	37,3527667	-6,1964585	1,5	21
6	Bollullos de la Mitación	12765	37,339834	-6,1381188	1,5	28
7	Bormujos	26000	37,3722503	-6,0701344	1	13
8	Camas	32552	37,40007	-6,033412	1	13
9	Carrión de los Céspedes	3011	37,3696888	-6,3292537	1	9
10	Castilleja de Guzmán	3339	37,409549	-6,0569993	1	6
11	Castilleja de la Cuesta	20611	37,386032	-6,05524	1	6
12	Castilleja del Campo	745	37,3875697	-6,3368981	1,5	15
13	Coria del Rio	36419	37,289016	-6,053153	1	28
14	Espartinas	18686	37,3800349	-6,1328179	1	17
15	Gelves	12051	37,3409234	-6,0262352	1	11
16	Gines	15880	37,3873487	-6,0780749	1	7
17	Huévar del Aljarafe	3568	37,356338	-6,275504	1,5	27
18	La Puebla del Río	14044	37,2694065	-6,063133	2	69
19	Isla Mayor	6910	37,1331542	-6,16463	1,5	38
20	Olivares	11116	37,418035	-6,155966	1,5	24

Tabla 0.3. UGR nº2 – Estructura de Datos

Nº	Municipio	Prod. Residuos (kg/día)	Latitud	Longitud	Factor de concentración	Perímetro (km)
21	Palomares del Río	10374	37,3191342	-6,0684709	1	13
22	Pilas	16536	37,3034383	-6,2986857	1,5	25
23	Salteras	6544	37,4183764	-6,111619	1,5	27
24	San Juan de Aznalfarache	25342	37,358824	-6,03433	1	8
25	Sanlúcar la Mayor	16339	37,3870837	-6,2017649	1,5	42
26	Santiponce	10122	37,4352111	-6,0427732	1	11
27	Tomares	30007	37,375584	-6,044533	1	9
28	Umbrete	10525	37,3681512	-6,1566754	1	13
29	Valencina de la Concepción	9172	37,4148311	-6,0771864	1,5	18
30	Villamanrique de la Condesa	5277	37,2450283	-6,307323	1,5	27
31	Villanueva del Ariscal	7822	37,3952677	-6,1419402	1	8
32	Estación de transferencia Espartinas		37,39964	-6,106847		

Tabla 0.4. UGR nº2 – Estructura de Datos (Continuación)

Parámetro	Descripción	Valor asignado
m	Número de vehículos	32
cont	Número de contenedores	10225

Tabla 0.5. UGR nº2 – Parámetros particulares

**UGR nº3 – Mancomunidad de Servicios de la Vega**

Nº	Municipio	Prod. Residuos (kg/día)	Latitud	Longitud	Factor de concentración	Perímetro (km)
0	DEPOT		37,595629	-6,011890		
1	Alcalá del Río	17576	37,517980	-5,981255	1,5	33
2	Alcolea del Río	4929	37,614914	-5,666418	1,5	26
3	Brenes	18222	37,550077	-5,870369	1	17
4	Burguillos	9813	37,585713	-5,967938	1,5	23
5	Cantillana	15611	37,610688	-5,823293	1,5	37
6	Castilblanco de los Arroyos	7107	37,675176	-5,989428	2	64
7	El Castillo de las Guardas	2109	37,692959	-6,315104	2	58
8	El Garrobo	1155	37,625311	-6,171575	2	24
9	El Madroño	407	37,645394	-6,511296	2	36
10	El Ronquillo	1993	37,726299	-6,176526	2	32
11	Gerena	11083	37,528844	-6,153896	1,5	41
12	Guillena	18685	37,539807	-6,053855	1,5	54
13	La Algaba	23924	37,462107	-6,011724	1	15
14	Lora del Río	27267	37,657081	-5,527189	1,5	61
15	Peñaflor	5342	37,709488	-5,344757	1,5	33
16	Tocina	13882	37,609315	-5,734590	1	14
17	Villanueva del Río y Minas	7084	37,655528	-5,711018	1,5	44
18	Villaverde del Río	11423	37,588658	-5,874268	1,5	23
19	ET Lora del Río		37,647763	-5,555058		
20	PT Alcalá del Río (La Vega)		37,595629	-6,011890		

Tabla 0.6. UGR nº3 – Estructura de Datos

Parámetro	Descripción	Valor asignado
m	Número de vehículos	16
cont	Número de contenedores	5080

Tabla 0.7. UGR nº3 – Parámetros particulares

#### UGR nº4 – Mancomunidad de Municipios Sierra Norte

Nº	Municipio	Prod. Residuos (kg/día)	Latitud	Longitud	Factor de concentración	Perímetro (km)
0	DEPOT		37,946608	-5,762586		
1	Alanís	2063	38,038352	-5,713502	2	60
2	Almadén de la Plata	1623	37,873398	-6,081408	2	57
3	Cazalla de la Sierra	5649	37,932993	-5,759546	2	67
4	Constantina	7060	37,871078	-5,619927	2	78
5	El Pedroso	2417	37,841620	-5,765541	2	63
6	El Real de la Jara	1800	37,949522	-6,154701	2	45
7	Guadalcanal	3146	38,093355	-5,817786	2	59
8	La Puebla de los Infantes	3582	37,780672	-5,388433	2	45
9	Las Navas de la Concepción	1867	37,933312	-5,465335	2	29
10	San Nicolas del Puerto	714	37,993802	-5,653329	2	24
11	PT de la Vega		37,595629	-6,011890		
12	ET de Constantina		37,858210	-5,631136		

Tabla 0.8. UGR nº4 – Estructura de Datos

Parámetro	Descripción	Valor asignado
m	Número de vehículos	3
cont	Número de contenedores	780

Tabla 0.9. UGR nº4 – Parámetros particulares

**UGR nº5 – Consorcio Estepa – Sierra Sur**

Nº	Municipio	Prod. Residuos (kg/día)	Latitud	Longitud	Factor de concentración	Perímetro (km)
0	DEPOT		37,300506	-4,882831		
1	Aguadulce	1983	37,252649	-4,992688	1,50	14,00
2	Algamitas	1232	37,016021	-5,149505	1,50	17,00
3	Badolatosa	3021	37,305274	-4,674314	1,50	25,00
4	Casariche	5355	37,293594	-4,762451	1,50	26,00
5	El Rubio	3345	37,353464	-4,991520	1,50	17,00
6	El Saucejo	4205	37,072743	-5,098005	1,50	35,00
7	Estepa	12273	37,291927	-4,877523	1,50	49,00
8	Gilena	3658	37,251785	-4,915061	1,50	26,00
9	Herrera	6341	37,360235	-4,847224	1,50	26,00
10	La Roda de Andalucía	4106	37,201740	-4,778464	1,50	32,00
11	Lora de Estepa	18315	37,269795	-4,826703	1,00	16,00
12	Los Corrales	3868	37,099945	-4,984882	1,50	29,00
13	Marinaleda	2579	37,371432	-4,959865	1,50	18,00
14	Martin de la Jara	2654	37,108767	-4,962798	1,50	26,00
15	Pedrera	5098	37,224359	-4,895093	1,50	28,00
16	Pruna	2555	36,972809	-5,222531	2,00	36,00

Tabla 0.10. UGR nº5 – Estructura de Datos

Nº	Municipio	Prod. Residuos (kg/día)	Latitud	Longitud	Factor de concentración	Perímetro (km)
17	Villanueva de San Juan	1100	37,050118	-5,175106	2,00	21,00
18	PT de Mata Grande		37,339360	-4,866245		
19	ET de El Saucejo		37,051691	-5,119167		

Tabla 0.11. UGR nº5 – Estructura de Datos (Continuación)

Parámetro	Descripción	Valor asignado
m	Número de vehículos	7
cont	Número de contenedores	2100

Tabla 0.12. UGR nº5 – Parámetros particulares

#### UGR nº6 – Mancomunidad de Municipios Comarca de Écija

Nº	Municipio	Prod. Residuos (kg/día)	Latitud	Longitud	Factor de concentración	Perímetro (km)
0	DEPOT		37,544798	-5,064673		
1	Cañada Rosal	3818	37,596913	-5,210417	1,50	19,00
2	Fuentes de Andalucía	8177	37,464202	-5,346539	1,50	44,00
3	La Campana	6067	37,569126	-5,426627	1,50	40,00
4	La Luisiana	5262	37,526862	-5,248625	1,50	24,00
5	ET de Écija		37,485869	-5,148397		

Tabla 0.13. UGR nº6 – Estructura de Datos

Parámetro	Descripción	Valor asignado
m	Número de vehículos	2
cont	Número de contenedores	600

Tabla 0.14. UGR nº6 – Parámetros particulares

**UGR nº7 – Mancomunidad Intermunicipal Campiña 2000**

Nº	Municipio	Prod. Residuos (kg/día)	Latitud	Longitud	Factor de concentración	Perímetro (km)
0	DEPOT		37,234331	-5,372465		
1	Arahal	22570	37,259938	-5,543761	1,50	51,00
2	La Lantuejuela	4409	37,353854	-5,223267	1,50	15,00
3	La Puebla de Cazalla	12691	37,221959	-5,313825	1,50	49,00
4	Marchena	22491	37,327756	-5,416005	1,50	70,00
5	Moron de la Frontera	31934	37,120235	-5,454437	1,50	74,00
6	Osuna	20298	37,236945	-5,102718	2,00	87,00
7	Paradas	7985	37,289239	-5,495198	1,50	38,00
8	PT Campiña 2000		37,234294	-5,372465		

Tabla 0.15. UGR nº7 – Estructura de Datos

Parámetro	Descripción	Valor asignado
m	Número de vehículos	9
cont	Número de contenedores	3145

Tabla 0.16. UGR nº7 – Parámetros particulares

