



DOBLE GRADO EN  
MATEMÁTICAS Y ESTADÍSTICA

---

TRABAJO FIN DE GRADO

---

*El Problema Dinámico  
de Rutas de Vehículos*

---

Carlos Martín Carretié

Tutor: Antonio Beato Moreno

Sevilla, Junio de 2021



# Índice general

Resumen . . . . .	III
Abstract . . . . .	IV
Índice de Figuras . . . . .	V
Índice de Tablas . . . . .	VII
<b>1. Introducción</b>	<b>1</b>
1.1. El Problema de Rutas de Vehículos . . . . .	1
1.1.1. Travelling Salesman Problem . . . . .	1
1.1.2. Truck Dispatching Problem . . . . .	2
1.1.3. Algunas variantes . . . . .	3
<b>2. Problema Dinámico de Rutas de Vehículos</b>	<b>5</b>
2.1. Planteamiento del Problema Dinámico . . . . .	5
2.2. Medidas de Dinamismo . . . . .	6
2.2.1. Grado de Dinamismo . . . . .	7
2.2.2. Grado de Dinamismo Efectivo . . . . .	7
2.2.3. Grado de Dinamismo Efectivo con Ventanas de Tiempo . . . . .	7
2.3. Problemas estocásticos, clasificación . . . . .	8
2.4. Aplicaciones . . . . .	9
2.4.1. Transporte de Servicios . . . . .	9
2.4.2. Transporte de Bienes . . . . .	10
2.4.3. Transporte de Personas . . . . .	11
<b>3. Métodos de resolución</b>	<b>13</b>
3.1. Métodos exactos . . . . .	13
3.1.1. Programación Dinámica . . . . .	13
3.1.2. Programación Lineal . . . . .	14
3.1.3. Método de Ramificación y Poda . . . . .	15
3.2. Métodos Heurísticos y Metaheurísticos . . . . .	15
3.2.1. Método de los Ahorros de Clarke-Wright . . . . .	15
3.2.2. Algoritmos Genéticos . . . . .	16
3.2.3. Búsqueda Tabú . . . . .	17
<b>4. El Problema de Reequilibrio del Sistema de Bicicleta Pública: BiciMAD</b>	<b>19</b>
4.1. Presentación del problema . . . . .	19
4.2. Formulación lineal . . . . .	23
4.2.1. Definiciones previas . . . . .	24
4.2.2. Formulación 1 . . . . .	27
4.2.3. Formulación 2 . . . . .	29

4.2.4.	Variantes . . . . .	31
4.2.4.1.	Variante 1: Estaciones asistidas por más de un vehículo . . . . .	31
4.2.4.2.	Variante 2: Vehículos que repiten estaciones . . . . .	33
4.2.4.3.	Variante 3: Depósitos no iniciales en la ruta . . . . .	33
4.2.4.4.	Variante 4: Finalización en otro depósito . . . . .	34
4.2.4.5.	Variante 5: Restricción de Dantzig–Fulkerson–Johnson . . . . .	35
4.2.4.6.	Variante 6: Problema multiobjetivo . . . . .	35
4.3.	Resolución Lineal del BRP: BiciMAD . . . . .	36
4.3.1.	Descripción de la implementación . . . . .	36
4.3.2.	Descripción de la solución . . . . .	41
<b>A.</b>	<b>Apéndice: Soluciones</b>	<b>47</b>
A.1.	Resolución problema lineal . . . . .	47
A.1.1.	Cluster 1 . . . . .	47
A.1.2.	Cluster 2 . . . . .	51
A.1.3.	Cluster 3 . . . . .	54
A.1.4.	Cluster 4 . . . . .	57
A.1.5.	Cluster 5 . . . . .	60
<b>B.</b>	<b>Apéndice: Código</b>	<b>63</b>
B.1.	Código AMPL . . . . .	63
B.2.	Código R . . . . .	66
B.2.1.	Preparación de los datos, obtención de información del uso de BiciMAD . . . . .	66
B.2.2.	Clustering y resolución del problema lineal . . . . .	71
	<b>Bibliografía</b>	<b>79</b>

# Resumen

En este trabajo se estudiará el Problema de Rutas de Vehículos asociado a un problema de optimización. Comenzaremos dando una definición general y presentando los primeros y más sencillos problemas planteados, así como algunas variantes que se les pueden aplicar.

En el segundo capítulo se presenta el Problema Dinámico de Rutas de Vehículos, en el que algunos datos del problema sólo están disponibles una vez los vehículos han comenzado la ruta. Se definen parámetros, propuestos en la literatura, que miden el nivel de dinamismo de un problema y se hace una clasificación del Problema de Rutas de Vehículos en función de la evolución y calidad de la información disponible. Además, se exponen ejemplos de problemas dinámicos de rutas de vehículos presentes en la vida cotidiana.

En el tercer capítulo se plantean distintos métodos de resolución del Problema Dinámico de Rutas de Vehículos ofrecidos en la literatura, distinguiendo entre los métodos exactos, en los que se obtiene la solución óptima con mayor coste computacional, y los heurísticos y metaheurísticos, con menor tiempo computacional pero que no necesariamente alcanzan la mejor solución.

Finalmente, en el último capítulo se plantea el El Problema de Reequilibrio del Sistema de Bicicleta Pública, en el que se busca que las estaciones de bicicletas públicas de transporte urbano estén debidamente surtidas para ofrecer un buen servicio a los usuarios. Se desarrollará un algoritmo de programación lineal para resolver este problema. Nos centraremos en el Sistema de Bicicleta Pública de la ciudad de Madrid, BiciMAD, que analizaremos y al que aplicaremos el método propuesto.

# Abstract

This work will study the Vehicle Routing Problem associated to an optimization problem. First, we will provide a general definition and introduce the first and simplest problems that have been set out. Futhermore, some variations of these problems will be presented.

In the second chapter we introduce the Dynamic Vehicle Routing Problem, in which some data is revealed once the vehicle has already started the route. Parameters that measure the level of dynamism of a problem proposed in the literature will be shown, as well as a classification of the vehicle routing problems according to the evolution and reliability of the available information. In addition, we will give examples of dynamic vehicle routing problems which appear in our day to day life.

In the third chapter we will propose different resolution methods of the Dynamic Vehicle Routing Problem suggested in the literature. We will point out the difference between exact methods, that provide optimal solution with a higher computational cost, and heuristic and meta-heuristic problems, with a less computational time but also with a lack of guarantee of finding the best solution.

Finally, in the last chapter we will lay out the Bike Sharing Rebalancing Problem, whose objective is to provide the bike stations of an Urban Bike Sharing System with a correct amount of bicycles in order to provide a good service to the users of the system. We will develop a linear programming algorithm so as to solve this problem. We will focus our attention on the Bike Sharing System of the city of Madrid, BiciMAD, that we will analyse and apply the proposed method.

# Índice de figuras

1.1. Truck Dispatching Problem . . . . .	2
4.1. Demanda mensual por Distrito y Barrio . . . . .	21
4.2. Demanda mensual: Barrio Residencial . . . . .	21
4.3. Demanda mensual: Hospital . . . . .	22
4.4. Demanda mensual: Nodo de comunicación . . . . .	23
4.5. Anchura media según el número de clusters . . . . .	39
4.6. Estaciones BiciMAD . . . . .	40
4.7. Ruta Optima del Primer Cluster . . . . .	45
A.1. Ejecución Cluster 1 . . . . .	47
A.2. Ejecución Cluster 2 . . . . .	51
A.3. Ejecución Cluster 3 . . . . .	54
A.4. Ejecución Cluster 4 . . . . .	57
A.5. Ejecución Cluster 5 . . . . .	60
B.1. Código común a ambas formulaciones . . . . .	63
B.2. Código común a ambas formulaciones . . . . .	64
B.3. Código particular de la primera formulación . . . . .	64
B.4. Código particular de la segunda formulación . . . . .	65
B.5. Código particular de la variante 6 . . . . .	65





# Índice de tablas

2.1. Clasificación Problemas de Rutas de Vehículos . . . . .	8
3.1. Algoritmo de Clarke-Wright . . . . .	16
3.2. Algoritmos Genéticos . . . . .	16
4.1. Resumen de la notación . . . . .	26
4.2. Extracto Estaciones Bicimad . . . . .	36
4.3. Extracto del uso de estaciones por hora . . . . .	37
4.4. Solución variable x, Primer Cluster . . . . .	42
4.5. Solución variable f, Primer Cluster . . . . .	43
4.6. Identificación de las estaciones del primer cluster . . . . .	43
A.1. Solución Variable x Primer Cluster . . . . .	48
A.2. Solución Variable f Primer Cluster . . . . .	49
A.3. Identificación de las estaciones del primer cluster . . . . .	49
A.4. Solución Variable x Segundo Cluster . . . . .	51
A.5. Solución Variable f Segundo Cluster . . . . .	52
A.6. Identificación de las estaciones del segundo cluster . . . . .	52
A.7. Solución Variable x Tercer Cluster . . . . .	54
A.8. Solución Variable f Tercer Cluster . . . . .	55
A.9. Identificación de las estaciones del tercer cluster . . . . .	55
A.10.Solución Variable x Cuarto Cluster . . . . .	57
A.11.Solución Variable f Cuarto Cluster . . . . .	58
A.12.Identificación de las estaciones del cuarto cluster . . . . .	58
A.13.Solución Variable x Quinto Cluster . . . . .	60
A.14.Solución Variable f Quinto Cluster . . . . .	61
A.15.Identificación de las estaciones del quinto cluster . . . . .	61



# Capítulo 1

## Introducción

En las últimas décadas hemos experimentado un gran desarrollo en el ámbito del transporte y la logística a distintas escalas. El incremento del comercio internacional ha hecho posible el desarrollo de cadenas de transporte más complejas a nivel global. Por otro lado, el aumento de la población urbana ha propiciado una mayor demanda de productos y servicios cada vez más variados estableciéndose así grandes redes de transporte urbano que es preciso gestionar de forma adecuada. Además, el desarrollo tecnológico, especialmente en telecomunicaciones, no sólo ha abaratado los costes del transporte sino que hace posible la obtención y el procesamiento de un mayor volumen de datos que permite obtener un mayor conocimiento tanto a los proveedores sobre la demanda como a los clientes sobre el estado de sus pedidos. Debido a que los clientes esperan precios cada vez más bajos y a la vez satisfacer sus demandas de forma más rápida, el proceso de toma de decisiones por parte del proveedor es cada vez más complejo y relevante.

### 1.1. El Problema de Rutas de Vehículos

Un **Problema de Rutas de Vehículos** (VRP, del inglés *Vehicle Route Problem*) se define generalmente como un grafo  $G = (\mathcal{V}, \mathcal{E}, D)$ , siendo  $\mathcal{V} = \{v_0, \dots, v_n\}$  un conjunto de vértices,  $\mathcal{E} = \{(v_i, v_j) \mid (i, j) \in \mathcal{V}^2, i \neq j\}$  un conjunto de arcos que unen los vértices y  $D = (d_{ij})_{(i,j) \in \mathcal{E}}$  una matriz de costes definida sobre  $\mathcal{E}$ , que suele representar las distancias, los tiempos o los costes de viaje entre dos vértices adyacentes. Tradicionalmente, al vértice  $v_0$  se le llama *depósito*, mientras que el resto de los vértices suelen ser los clientes que demandan bienes o servicios. El objetivo de un problema de rutas de vehículos es hallar el valor óptimo de una función asociada satisfaciendo una serie de restricciones. Tradicionalmente la función objetivo es el coste empleado en la selección de un subconjunto de arcos de  $\mathcal{E}$ , es decir, en la construcción de la ruta.

#### 1.1.1. Travelling Salesman Problem

Los *Problemas de Rutas de Vehículos* son una generalización del **Problema del Vendedor Ambulante** (TSP, del inglés *Travelling Salesman Problem*), también conocido

como el *Problema del Viajante* o como el *Problema del Mensajero*, entre otros. Este problema trata de hallar la ruta más corta entre las ciudades donde un vendedor ambulante tiene sus clientes, visitando cada ciudad una sola vez y regresando finalmente a la ciudad de origen. El problema consiste realmente, tal y como lo formuló Merrill M. Flood [8] en 1955, en encontrar la permutación  $P = (1, i_1, i_2, \dots, i_n)$  que minimice la distancia total  $D_T$ .

$$D_T = d_{1,i_1} + d_{i_1,i_2} + \dots + d_{i_n,1}$$

Siendo  $d_{\alpha,\beta}$  la distancia entre las ciudades  $\alpha$  y  $\beta$ .

### 1.1.2. Truck Dispatching Problem

Dantzig y Ramser [4] generalizaron en 1959 el *Problema del Vendedor Ambulante* formulando el **Problema de Envío de Camiones** (*Truck Dispatching Problem*), llamado así porque modeliza el reparto de gasolina a distintas estaciones por camiones desde una central. Se impone la condición de que en cada estación de gasolina se demanda una cierta cantidad de gasolina  $q_i$ . En este caso se considera que la flota de  $m$  camiones es homogénea, es decir, que todos tienen una misma capacidad  $C$  que cumple que:

$$C < \sum_{i=1}^n q_i$$

Siendo  $n = |\mathcal{V}|$ . Es decir, que la capacidad de cada camión no puede satisfacer toda la demanda y por tanto debe llevarse a cabo más de una ruta realizando así una partición del conjunto de vértices  $\mathcal{V} \setminus \{v_0\}$ . En este caso,  $D$  representa los costes de viaje en lugar de las distancias. La **finalidad** de los Problemas de Rutas de Vehículos es encontrar rutas para los  $m$  vehículos que satisfagan la demanda de los clientes y minimicen el coste total de la ruta.

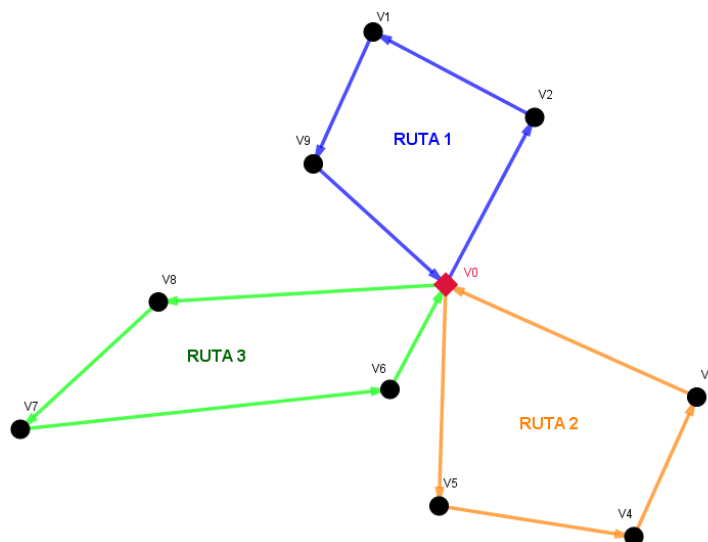


Figura 1.1: Truck Dispatching Problem

Figura 1: TDP

### 1.1.3. Algunas variantes

Los Problemas de Rutas de Vehículos que encontramos en la vida real suelen ser variantes de los anteriores. A continuación mostramos algunos ejemplos:

- **Problema de Rutas de Vehículos con Ventanas de Tiempo** (VRPTW, del inglés *Vehicle Route Problem with Times Windows*), que consiste en asociar a cada cliente un intervalo de tiempo en el que su demanda debe ser satisfecha.
- **Problema de Rutas de Vehículos con Recogida y Entrega** (PDP o VRPPD del inglés *Pick and Delivery Problem*), en el que diferentes bienes o personas deben de ser transportadas de un vértice a otro.
- A veces las rutas no tienen por qué acabar en el depósito y estamos ante un **Problema de Rutas de Vehículos Abierto** (OVRP, por sus siglas en inglés) o hay más de un depósito y se trata de un **Problema de Rutas de Vehículos con Múltiples Depósitos** (MDVRP, por sus siglas en inglés).
- **Problema de Rutas de Vehículos de Flota mixta** (MFVRP). En estos problemas la flota de vehículos no es homogénea, es decir, las capacidades de cada vehículo son diferentes.

En definitiva, muchas son las variaciones que pueden hacerse y combinarse en un Problema de Rutas de Vehículos.



# Capítulo 2

## Problema Dinámico de Rutas de Vehículos

### 2.1. Planteamiento del Problema Dinámico

Ninguna de las variantes que hemos visto tiene en cuenta la *evolución* de la información. Hasta ahora, hemos considerados *Problemas Estáticos de Rutas de Vehículos*. En ellos, toda la información relevante para el diseño de las rutas es asumida como conocida por el decisor antes de que la ruta comience y no cambia una vez que la ruta es establecida.

Con el tiempo, los problemas en los que no toda la información es sabida de antemano se van haciendo cada vez más frecuentes. Los avances tecnológicos en telecomunicaciones han conllevado un uso generalizado de teléfonos móviles inteligentes y precisos Sistemas de Información Geográfica. En muchos casos esto permite a los clientes comunicarse con el proveedor telemáticamente y transmitir de forma casi inmediata cambios en las características de su demanda antes de que esta haya sido satisfecha. También supone que los proveedores y los clientes pueden monitorear los envíos en tiempo real, lo que puede alterar las decisiones de los propios clientes pero también permite a los proveedores adaptarse y administrar los envíos de forma más eficiente. En estos casos estamos ante un *Problema Dinámico de Rutas de Vehículos* (DVRP, del inglés *Dynamic Vehicle Route Problem*).

Podemos **definir un Problema Dinámico de Rutas de Vehículos** como aquel en el que no toda la información relevante es sabida al inicio del establecimiento de las rutas y puede añadirse y modificarse después de que la ruta haya sido establecida.

Cualquier aspecto del problema puede ser dinámico. Sin duda la fuente de dinamismo más habitual es la aparición de nuevos clientes una vez la ruta ha comenzado y los cambios en la demanda de clientes ya conocidos. También pueden ser dinámicos, por ejemplo, los costes o tiempos de transporte o el número de vehículos disponibles en la flota.

Si el problema original tiene muchas restricciones, como por ejemplo pueden ser las anteriormente citadas Ventanas de Tiempo, la adición de un nuevo cliente puede elevar significativamente los costes o incluso provocar que sea imposible satisfacer la demanda de todos los clientes a tiempo. Por este motivo es más común en los problemas dinámicos que en los estáticos que el proveedor pueda rechazar a un cliente. En estos casos el objetivo no es encontrar las rutas que atiendan a todos los clientes con mínimo coste sino que suele ser maximizar los beneficios resultantes de restar los costes a los ingresos obtenidos satisfaciendo las demandas de los clientes. También puede ser que el objetivo sea satisfacer la máxima cantidad de demanda o de clientes para un coste máximo fijado.

Powell et al. [16] distinguieron entre el dinamismo de un problema, de su modelo y de su aplicación. Un problema es dinámico si alguno de sus parámetros varían con el tiempo, mientras que un modelo de ese problema es dinámico si incorpora explícitamente ese dinamismo del problema. Muchos modelos estáticos realmente explican problemas dinámicos en los que la fuente de dinamismo ha sido desdeñada. Por ejemplo, muchas veces los tiempos de viaje entre un cliente y otro no son fijos, pueden depender del tráfico o de situaciones variadas que alarguen o acorten los tiempos considerados a priori en el problema, de tal forma que puede pasar que la solución hallada de forma estática no sea una buena solución. Por último, una aplicación es dinámica si el modelo subyacente es resuelto repetidamente conforme se recibe la nueva información.

Se puede entender la salida de la resolución de un modelo dinámico no como un conjunto de rutas sino como un conjunto de instrucciones de cómo las rutas deben cambiar en función de cómo va entrando la nueva información. La aplicación dinámica del problema es la ejecución de esas instrucciones. Para ello se necesita un equipamiento tecnológico que permita al proveedor controlar y comunicarse con la flota y con los clientes. Además, resolver modelos con aplicaciones dinámicas precisa de grandes esfuerzos computacionales. En general, a mayor dinamismo es más costoso generar nuevas soluciones de forma suficientemente rápida.

Consecuentemente, la resolución de un problema dinámico se verá afectada no sólo por los aspectos comunes con los problemas estáticos sino con el nivel de dinamismo. Este puede variar de forma muy significativa entre distintos problemas y entre aplicaciones de un mismo modelo de problema. Medir el dinamismo nos puede ser de utilidad para evaluar el desempeño de los algoritmos bajo distintas condiciones y ayudarnos a decidir el mejor para cada caso.

## 2.2. Medidas de Dinamismo

A continuación vamos a ver parámetros que sirven para medir el nivel de dinamismo. Vamos a considerar que la fuente de dinamismo es únicamente la aparición de nuevos clientes. Para medir el nivel de dinamismo deben tenerse en cuenta dos aspectos: la frecuencia de la entrada de nueva información y la urgencia de los pedidos, es decir, el tiempo que se dispone para satisfacer la demanda desde que es solicitada.



### 2.2.1. Grado de Dinamismo

Lund et al [13] propone la primera medida de dinamismo de un problema, el **grado de dinamismo**  $\delta$ , definido por el número de pedidos dinámicos  $n_d$ , es decir, recibidos en el transcurso de la ruta, y el número de pedidos totales  $n_{tot}$ .

$$\delta = \frac{n_d}{n_{tot}} \quad (2.1)$$

Esta medida no es muy buena puesto que no tiene en cuenta muchos aspectos que afectan al dinamismo de un problema. En particular no tiene en cuenta cuándo llegan los pedidos dinámicos. Esto afecta considerablemente al dinamismo puesto que en un problema donde los nuevos pedidos lleguen en cortos intervalos de tiempo se debe procesar la nueva información y recalcular las rutas de forma más rápida que en otro problema donde los nuevos pedidos se vayan realizando de forma distanciada en el tiempo. Si el objetivo es atender a los clientes cuanto antes conviene que los nuevos pedidos no aparezcan a la vez sino de forma más distanciada. Por el contrario, si lo que se desea es minimizar la distancia recorrida o el coste de las rutas es preferible que la nueva información llegue cuanto antes para evitar realizar recorridos que habría costado menos realizar si el pedido hubiera sido conocido cuando se estaba más cerca de él.

### 2.2.2. Grado de Dinamismo Efectivo

Larsen [10] propone el *grado de dinamismo efectivo*  $\delta^e$ , que puede entenderse como una media normalizada de los tiempos de revelación de los pedidos, es decir, una medida de cuánto de tarde se reciben los nuevos pedidos respecto de lo tarde que podrían haberlo sido. Denotamos por  $T$  la longitud del horizonte de sucesos y por  $t_i$  al tiempo en el que el pedido  $i$  es realizado. Nótese que  $0 < t_i < T \forall i \in \{1..n\}$  y que si  $i$  corresponde a un pedido realizado antes del comienzo de la ruta  $t_i = 0$ . La expresión es la siguiente:

$$\delta^e = \frac{1}{n_{tot}} \sum_{i \in \mathcal{R}} \frac{t_i}{T} \quad (2.2)$$

### 2.2.3. Grado de Dinamismo Efectivo con Ventanas de Tiempo

Larsen [10] también creó una medida similar aplicable a los problemas con Ventanas de Tiempo que tiene en cuenta la urgencia de los pedidos. Denotamos como  $l_i$  el tiempo en el que se cierra la ventana de tiempo del pedido  $i$ . La expresión es la siguiente:

$$\delta_{TW}^e = \frac{1}{n_{tot}} \sum_{i \in \mathcal{R}} \left( 1 - \frac{l_i - t_i}{T} \right) \quad (2.3)$$

Es fácil ver que las tres medidas están comprendidas en el intervalo  $[0, 1]$  y que si aumenta el nivel de dinamismo del problema las tres medidas aumentan también. Larsen et al. (2002, 2007) [12] [11] utilizaron estas medidas para clasificar los problemas según el grado de dinamismo: Débilmente dinámicos si  $\delta^e < 0.3$ , moderadamente dinámico si  $0.3 < \delta^e < 0.6$  y fuertemente dinámico si  $\delta^e > 0.6$

Estas medidas, como se ha mencionado anteriormente, miden el dinamismo únicamente cuando la fuente de éste es la aparición de nuevos pedidos, y tampoco tienen en cuenta factores como la dispersión geográfica de los clientes.

## 2.3. Problemas estocásticos, clasificación

Además de la *evolución* del problema hay otra dimensión a tener en cuenta: La *calidad de la información* del problema [17], es decir, la posible incertidumbre de los datos disponibles. Así, podemos clasificar los problemas como determinísticos o estocásticos, donde lo que se puede obtener de antemano son estimaciones de los parámetros reales del modelo, que son aleatorios. El problema estocástico más habitual es el *Problema de Rutas de Vehículos con Demanda Estocástica* (VRPSD, del inglés *Stochastic Demands*) donde es conocido un rango estimado de la demanda real, que no es revelada hasta que el cliente es atendido. Esto no significa que la demanda haya sufrido cambios, como en un problema dinámico, sino que la información que posee el distribuidor es inexacta. Fijándonos en la evolución y la calidad de la información, podemos clasificar los problemas de rutas de vehículos en cuatro categorías.

Tabla 2.1: Clasificación Problemas de Rutas de Vehículos

	Calidad de la información	
	<i>Input determinista</i>	<i>Input estocástico</i>
<i>Toda la información está disponible de antemano</i>	Problema estático y determinista	Problema estático y estocástico
<b>Evolución</b>		
<i>Información relevante es modificada en el transcurso del problema</i>	Problema dinámico y determinista	Problema dinámico y estocástico

- Entre los **problemas estáticos y deterministas** nos encontramos los clásicos problemas de rutas de vehículos ya mencionados. En ellos toda la información es conocida de antemano y las rutas no se modifican una vez comenzadas.
- En los **problemas estáticos y estocásticos** parte de los parámetros de entrada son variables aleatorias cuyo valor se obtiene durante la ejecución de las rutas. Estas son diseñadas a priori y los cambios que se pueden realizar por la aleatoriedad de las variables deben ser cambios menores que no afecten significativamente el resultado. Además, estos posibles cambios en la ruta deben ser contemplados en el diseño de la solución del problema, de forma que no es necesario resolver un nuevo problema una vez la ruta ha comenzado. Por este motivo, no es necesario el uso de la tecnología para mantener en contacto permanente entre el diseñador de la ruta y los conductores de los vehículos. Cualquier parámetro puede ser estocástico, pero

lo más habitual es que sean variables aleatorias los tiempos de viaje, si un cliente necesita ser atendido o no y el valor de la demanda de un cliente.

- En los **problemas dinámicos y deterministas** parte de los parámetros de entrada son desconocidos a priori y revelados en el transcurso de la ruta. En estos problemas sí es preciso el uso de tecnologías que permitan el contacto entre los vehículos y el diseñador de las rutas, pues es preciso rediseñarlas o introducir modificaciones oportunas conforme se añaden los nuevos datos.
- En los **problemas dinámicos y estocásticos** también sucede que parte de los parámetros de entrada son desconocidos a priori y revelados en el transcurso de la ruta. También se necesita que el diseñador de la ruta esté en contacto con la flota de vehículos. Estos problemas se diferencian de los anteriores en que se posee información estocástica sobre los datos que se revelarán dinámicamente.

## 2.4. Aplicaciones

Las aplicaciones de Problemas Dinámicos de Rutas de Vehículos en la vida real son muchas y muy diversas. Podemos encontrar multitud de Problemas Dinámicos de Rutas de Vehículos con objetivos, restricciones y particularidades propias. Veamos algunos ejemplos.

### 2.4.1. Transporte de Servicios

El Problema de Rutas de Vehículos más sencillo es el ya mencionado **Problema del Vendedor Ambulante**. Una variante dinámica muy usual es la aparición de clientes nuevos. Son aplicaciones habituales de este tipo de problemas los servicios de reparaciones y mantenimiento, con mayor grado de dinamismo que los problemas de ventas. En estas situaciones cada empleado tiene asociada una serie de clientes prefijados en el día o turno anterior y clientes que aparecen de forma dinámica conforme a lo largo del tiempo se suceden distintas averías. En estos problemas el dinamismo puede estar presente también en los tiempos o costes de viaje y en el tiempo en el que los empleados reparan la avería. De este último parámetro a priori desconocido es frecuente poseer información estocástica.

Es conveniente en muchas de estas aplicaciones una **evaluación** de la emergencia de las averías para atender primero las que son más urgentes y rechazar o posponer las que lo son menos. En estos problemas entra en conflicto la rápida reparación de las averías conocidas y la disponibilidad de los empleados para atender demandas más importante que puedan aparecer. Tras la evaluación del problema se le asocia a cada cliente una ventana de tiempo en la que el empleado debe atenderlo.

En los problemas de transporte de servicios encontramos usualmente restricciones en las **horas laborales de los empleados**. Esta restricción se define por un intervalo de horas de operabilidad de los empleados. Fuera de ese intervalo de tiempo, los vehículos deben

estar bien en un nodo base o bien fuera de la circulación. Esta restricción puede afectar a los pedidos de los clientes no sólo en las horas excluidas del intervalo de operabilidad, también en horas próximas a los extremos al ser rechazados o postpuestos pedidos para que el empleado tenga tiempo de llevar el vehículo a la base. Un problema análogo al de reparación de averías es el de la asistencia médica a domicilio.

Un tipo de Problema de Rutas de Vehículos es el **Problema del establecimiento de Rutas sobre Arcos** (ARP, del inglés *Arc Routing Problem*), en el que los vehículos deben prestar un servicio en los *arcos* del problema, en lugar de en los nodos, minimizando el coste total. Tagmouti et al. [21] aplicaron este problema a las máquinas quitanieves que deben despejar las calles de una población adaptándose a los eventos que vayan sucediendo.

En el Transporte de Servicios suele haber dos **objetivos** principales a considerar: El coste total de las rutas y el grado de satisfacción de los clientes. Se obtiene un alto grado de satisfacción de los clientes si se satisfacen sus demandas rápidamente. Ambos objetivos pueden entrar en conflicto y el peso de uno u otro dependerá de la naturaleza del problema.

### 2.4.2. Transporte de Bienes

Las aplicaciones de transporte urbano de bienes poseen unas características comunes que han permitido la definición de una categoría de VRP denominada **Logística Urbana** (*City Logistics*) que, según Taniguchi et. al. [22] es el “*proceso de total optimización de las actividades de transporte y logística de compañías privadas en áreas urbanas teniendo en consideración el tráfico y el consumo de energía en el marco de una economía de mercado*”. En la Logística Urbana encontramos, además de los repartidores, los clientes y el diseñador de la ruta, a los proveedores, situados en ubicaciones fijas donde dos repartidores recogen los bienes que entregan a los cliente. Suelen ser problemas de **Recogida y Entrega** (VRPPD) donde la recogida puede efectuarse en un único depósito central o en distintos puntos. En estas aplicaciones, a diferencia de los Problemas de Transporte de servicios, la **capacidad** de los vehículos juega un rol importante y debe tenerse en cuenta. Al tener en cuenta el estado del tráfico, el dinamismo en la Logística Urbana suele estar presente en los tiempos de transporte.

Un problema de transporte de bienes ya mencionado es el Problema de **Envío de Camiones** de Dantzig y Ramser [4]. En problemas parecidos donde el bien transportado es homogéneo (en este caso gasolina) y los pedidos de los clientes suelen hacerse con relativa antelación el dinamismo es débil y la solución dinámica o en línea no dista mucho de la ruta establecida en un modelo estático. Una aplicación parecida es la **recogida de basura** en las ciudades. En este problema los camiones deben recoger basura en los nodos y descargarla en un depósito. Se trata de un Problema de Recogida y Entrega donde la demanda, esto es, la cantidad de basura presente en los contenedores, suele ser un parámetro **estocástico**. El objetivo de estas aplicaciones es minimizar los costes derivados del transporte, la satisfacción de los clientes no tiene tanto peso como en el transporte de servicios.

El caso más habitual de Logística Urbana es el servicio de **correos** o mensajería, que puede consistir en entregar paquetes a clientes desde un depósito central, llevar paquetes desde determinados puntos a un depósito central o en la recogida y entrega de paquetes a distintas localizaciones. En estos casos la cantidad de demanda no suele ser estocástica pero la aparición de nuevos clientes es dinámica y puede poseerse información estocástica al respecto. El grado de **dinamismo** es mayor en problemas de recogida y entrega de paquetes. La restricción de la capacidad de los vehículos debe ser tenida en cuenta. Además, las flotas de las empresas de mensajería no tienen por qué ser homogéneas y los vehículos suelen tener **distintas capacidades**. En los problemas de Logística Urbana solemos encontrar también restricciones en las horas laborales de los repartidores

La satisfacción del cliente y la fiabilidad del servicio cobran en el envío de paquetes un protagonismo mayor que en la recogida de basuras o en el Problema de Envío de Camiones. No obstante, la reducción de costes de transporte sigue manteniendo un rol esencial en la búsqueda de la ruta óptima. En los servicios de **compra a domicilio**, sin embargo, la satisfacción del cliente y la fiabilidad de la empresa cobra una relevancia mayor que los costes de transporte. Estos servicios, cada vez más solicitados, son **fuertemente dinámicos**, especialmente los de comida a domicilio, ya que el tiempo transcurrido desde que se solicitan los pedidos hasta la entrega suele ser corto. En los problemas de comida a domicilio la demanda es determinista y conocida, pero la llegada de nuevos clientes tiene un alto grado de dinamismo. Los tiempos de transporte también son dinámicos y pueden variar no sólo en función del tráfico sino también del vehículo de la flota escogido.

Además del ámbito urbano, los problemas de transporte de bienes se aplican en puertos, grandes fábricas o almacenes para transportar de manera eficiente diversos materiales o contenedores, y en hospitales, para transportar documentos o instrumentos médicos rápidamente.

### 2.4.3. Transporte de Personas

El transporte de personas en una red urbana se realiza mediante distintos medios. En el transporte de personas mediante **autobuses** la demanda de pasajeros en cada parada es desconocida, aunque se puede tener información estocástica. Los tiempos de transporte entre una parada y otra son parámetros dinámicos que dependen del tráfico, al igual que el tiempo que un autobús permanece en una parada para cargar o descargar pasajeros. No obstante, los autobuses por lo general no se desvían de su ruta. Las decisiones que toma el administrador de la flota pueden ser reforzar una línea con más demanda en detrimento de otra o indicar al conductor de un autobús que lleva un gran número de personas y va con retraso respecto al diseño inicial que no acepte nuevos pasajeros para aligerar el recorrido y vaciar el autobús si otro vehículo de la misma línea se encuentra cerca de la parada donde los pasajeros han sido rechazados. Si bien se trata de una aplicación dinámica al existir la necesidad de una comunicación constante entre el administrador y los vehículos para tomar decisiones en tiempo real, el **nivel de dinamismo es débil** puesto que las rutas finales no difieren mucho de las planeadas antes de comenzar la ruta.

En cambio, el caso del transporte en **taxi** o mediante plataformas que realizan un servicio similar sí es claramente dinámico, de hecho, no es viable diseñar las rutas del vehículo antes de empezar el recorrido. Si la política a seguir es que el vehículo más cercano a un nuevo cliente es el que debe transportarlo, el margen de toma de decisiones del administrador de la flota se reduce una vez el cliente ha hecho un pedido. En estos casos se deben diseñar rutas para los vehículos disponibles con el objetivo de minimizar el tiempo total de circulación de vehículos disponibles, por lo que **información estocástica** que sirva para predecir la llegada de clientes nuevos es muy útil.

Las aplicaciones en las que los vehículos pueden transportar varias personas y recoger nuevos clientes cuando aún están transportando a otros son más complejas. Estos problemas reciben el nombre de **dial-a-ride** (DARP). Se trata de un problema equivalente al VRPPD pero aplicado al transporte de personas en vez de al de bienes. Son ejemplos de estos casos el **transporte escolar**, el **transporte especial adaptado a personas con discapacidad** y el **transporte de pacientes a centros de asistencia sanitaria**. Cordeau et. al.[3] estudiaron esta clase de problemas. Zang et. al. [24] estudiaron en particular el caso de pacientes que deben ser trasladados a un centro médico. En estos problemas los transportistas deben transportar a un número de clientes a centros médicos y devolverlos a sus hogares posteriormente, respetando el horario laboral de los trabajadores. La capacidad de los vehículos es limitada y el servicio debe realizarse en unas determinadas ventanas de tiempo. El dinamismo de estas aplicaciones no sólo está presente en los tiempos de transporte, también en el tiempo de subida y bajada de los clientes al vehículo, puesto que pueden necesitar ayuda, y en el tiempo que tardan en atenderles en el centro médico. Además, pese a que en este tipo de aplicaciones los clientes suelen solicitar el servicio por adelantado pueden surgir clientes imprevistos, luego se trata de un problema **fuertemente dinámico** en el que el servicio debe adaptarse rápidamente a cualquier cambio respecto a la planificación inicial.

Los **servicios de emergencias** también son casos de transporte de personas **fuertemente dinámicos**. Al igual que en el transporte de personas a un centro sanitario, el objetivo es maximizar la **fiabilidad del servicio**, es decir, minimizar los tiempos de respuesta. En estos casos el uso de tecnología para comunicar los vehículos con una centralita y la capacidad de esta para generar nuevas rutas ante cualquier cambio de información es esencial.

# Capítulo 3

## Métodos de resolución

El Problema de Rutas de Vehículos más sencillo es el Problema del Vendedor Ambulante, este problema es equivalente a encontrar el menor ciclo hamiltoniano en un grafo empezando por un vértice dado. El problema del ciclo hamiltoniano es un problema NP-completo. Los Problemas de Rutas de Vehículos son generalmente más complejos que el TSP, son problemas **NP-hard** por lo que no es siempre posible encontrar soluciones óptimas a problemas de cierto tamaño en tiempos razonables. Por este motivo es común emplear métodos heurísticos, es decir, procedimientos más sencillos que permiten encontrar soluciones satisfactorias en un menor tiempo computacional del que se tendría empleando técnicas exactas. En este capítulo veremos algunos métodos de resolución.

### 3.1. Métodos exactos

#### 3.1.1. Programación Dinámica

La Programación Dinámica es un conjunto de técnicas de resolución de problemas que se dividen de forma secuencial. Se pretende hallar el conjunto de decisiones que optimicen el valor de una variable (el menor coste, la mayor satisfacción...) Los problemas se descomponen en varios subproblemas denominados etapas. En cada etapa hay un conjunto finito de estados posibles. Debemos tomar una decisión en cada etapa, de forma que la solución óptima en una etapa no depende de las decisiones tomadas en etapas anteriores, sino únicamente del estado de la etapa en cuestión. La decisión que se toma en una etapa determina una transición a un estado distinto en la siguiente etapa.

En un problema de Programación Dinámica se tiene un funcional  $f$ . El valor del funcional en cada etapa es valor óptimo de la variable objetivo del problema a partir de esa etapa. El funcional depende del estado que encontremos en la etapa y está asociado a una decisión o decisiones concretas: **las decisiones óptimas**.

El funcional  $f$  de una etapa debe relacionar el valor de la variable objetivo en la etapa en cuestión con el funcional de la siguiente etapa.

El empleo de la Programación Dinámica en Problemas de Rutas de Vehículos fue Propuesto por Eilon, Watson-Gandy y Christofides en 1971 [7]. Psaraftis en 1980 [17] fue el primero en aplicar la Programación Dinámica a un **Problema Dinámico** de Rutas de Vehículos. En particular, Psaraftis ofreció una solución a un DARP con un único vehículo con restricciones de carga en el que la fuente de dinamismo es la aparición de nuevos clientes. Su solución consiste en resolver inicialmente **problema estático** y determinista con los clientes disponibles en un estado inicial y en **reevaluar** el problema con la aparición de cada nuevo cliente. La solución final no es la óptima si se conocieran todas las solicitudes de los clientes de antemano, pero sí se toma la **decisión óptima en cada momento** si se consideran únicamente los clientes conocidos. Se trata de un problema dinámico determinista.

### 3.1.2. Programación Lineal

La Programación lineal tiene como función optimizar una función lineal denominada función objetivo de forma que las variables de dicha función satisfagan un conjunto de restricciones lineales que pueden ser ecuaciones o inecuaciones.

Yang et. al. aplicaron en 2004 [23] la programación lineal a la resolución de un Problema de Múltiples Vehículos de Recogida y Entrega donde cada vehículo solo puede atender un pedido a la vez. En este problema el administrador de la flota puede permitirse rechazar pedidos. Para resolver este problema desarrollaron dos técnicas:

El método **MYOPT** comienza resolviendo mediante Programación Lineal un problema estático considerando únicamente los pedidos conocidos, obteniendo una **solución óptima**. Con cada aparición de un nuevo pedido el problema se **reevalúa** considerando la situación actual de vehículos y pedidos. Si el valor objetivo resultante de recalcular la ruta excede de cierta cantidad el administrador de la flota rechaza el pedido. La función objetivo tiene en cuenta la distancia total recorrida y el retraso en los pedidos, con pesos asociados a cada sumando según sea apropiado dar mas importancia a un parámetro u otro.

El método **OPTUN** es muy similar al MYOPT, la diferencia es que en el OPTUN se le aplica una transformación a la función objetivo. En lugar de la distancia nos encontramos con una variable que no sólo se tiene en cuenta la distancia entre dos puntos, también el **coste de oportunidad** provocado al desplazarse. Es decir, se tiene en cuenta **información estocástica** de dónde es más probable que surjan nuevos pedidos. En el OPTUN se castiga a los desplazamientos a nodos situados a grandes distancias de donde surja el resto, tendiendo a rechazar los pedidos en áreas más aisladas. Además, a la hora de asociar un vehículo a un pedido no sólo se tiene en cuenta la distancia de éste al pedido sino también el hecho de que al desplazar el vehículo hacia el pedido se esté acercando a la zona donde más pedidos nuevos pueden surgir.

Por tanto, el MYOPT ofrece un enfoque dinámico y determinista mientras que el OPTUN es un modelo dinámico y estocástico.



### 3.1.3. Método de Ramificación y Poda

Al igual que con la programación dinámica, el método de Ramificación y Poda (*Branch and Bound*) divide el problema en varios niveles en los que en cada uno se puede tomar un conjunto de decisiones que ramifican el estado del problema. Las hojas del árbol de decisión se corresponden con las soluciones finales del problema. En el Problema del Vendedor Ambulante, por ejemplo, cada nivel se corresponde con una ciudad visitada por la ruta, y la decisión que se plantea en ese nivel es qué ciudad de las no visitadas es la siguiente. El algoritmo va recorriendo el árbol descartando las ramas no factibles hasta dar con la hoja asociada a la solución óptima. La característica principal de este algoritmo es que para cada rama del árbol en el que nos encontremos el algoritmo es capaz de calcular una cota inferior o superior del valor de la función objetivo. De esta manera, si una solución a la que se ha llegado en otra rama es menor (en caso de que la función objetivo se esté minimizando) que la cota inferior de esa rama la descartamos, reduciendo el espacio de búsqueda.

Para aplicar este método a un problema dinámico se puede, al igual que con los métodos anteriores, reevaluar el problema de forma periódica o conforme se añada nueva información. De este método se pueden encontrar muchas variantes, algunas de ellas no son métodos exactos pues se combina con métodos heurísticos para reducir el tiempo computacional. Un ejemplo de ello es la combinación con algoritmos de búsqueda voraces (Amico et. al 2014 [6]) que sirven para construir soluciones iniciales suficientemente buenas que ahorran el tiempo de ejecución total al permitir así la “poda” de más ramas.

## 3.2. Métodos Heurísticos y Metaheurísticos

Las heurísticas son algoritmos que obtienen soluciones de buena calidad con un tiempo de ejecución considerablemente menor que los algoritmos exactos. Las técnicas metaheurísticas realizan una exploración más profunda que los métodos heurísticos del espacio de soluciones. Por otro lado, tienen un mayor tiempo de ejecución, aunque menor que el de los métodos exactos.

### 3.2.1. Método de los Ahorros de Clarke-Wright

Este método heurístico fue expuesto en 1964 [2] y busca encontrar una solución aceptable (en la búsqueda del mínimo coste) al Problema de Envío de Camiones [4].

Para describirlo usaremos la notación vista anteriormente en la sección 1.1, donde, recordemos, cada vértice está asociado a un número  $i \in \{0, 1, \dots, n\}$  siendo  $v_0$  el depósito.

Consideraremos que  $d_{ij} = d_{ji} \forall i, j \in \{0, 1, \dots, n\}$ , es decir, que los costes de desplazamiento entre los nodos son simétricos.

Los pasos a seguir son los siguientes:

Tabla 3.1: Algoritmo de Clarke-Wright

<b>Algoritmo de Clarke-Wright</b>	
1.	Se crean $n$ rutas de ida y vuelta al depósito, de la forma $(0, i, 0)$
2.	Se calcula el ahorro de costes que supone unir cada par de vértices:
$s_{ij} = 2(d_{0i} + d_{0j}) - (d_{0i} + d_{ij} + d_{0j}) = d_{0i} + d_{0j} - d_{ij}$	
3.	Se ordenan de forma decreciente los resultados obtenidos
4.	Se escogen los vértices $v_i, v_j$ asociados al primer (mayor) valor de los $s_{ij}$
5.	Si $v_i$ y $v_j$ son adyacentes al depósito en la ruta: Se unen las rutas que contienen ambos vértices, introduciendo el arco $(i, j)$ y eliminando una vez los arcos $(0, i)$ y $(0, j)$ , siempre que la ruta resultante sea factible
6.	Se escoge el siguiente par de vértices de la lista ordenada de ahorros. Si la lista ya se ha recorrido, hemos terminado

Las causas por las que una ruta puede no ser factible pueden ser, por ejemplo, a existencia de restricciones sobre la longitud de una ruta. También se pueden rechazar rutas por restricciones de carga. Si la capacidad máxima de un vehículo es  $C$  y la demanda de un vértice  $v_i$  es  $q_i$ , para toda ruta  $r$  debe tenerse que  $C < \sum_{i \in r} q_i$ .

Este algoritmo obtiene resultados buenos mediante una aplicación simple y con un tiempo de resolución corto.

### 3.2.2. Algoritmos Genéticos

Estas técnicas metaheurísticas están inspiradas en la Teoría de la Evolución de Darwin. El algoritmo se le aplica a un conjunto de soluciones factibles pero subóptimas que reciben el nombre de cromosomas. Estas son combinadas entre sí y modificadas creándose así mejores soluciones que mantienen algunas de las características de las funciones de las que provienen. Se define una función objetivo  $f$  que evalúa lo idónea de cada solución. Los pasos a seguir son los siguientes:

Tabla 3.2: Algoritmos Genéticos

<b>Algoritmos Genéticos</b>	
1.	<b>Inicialización:</b> Se genera un conjunto de soluciones factibles (o cromosomas) iniciales. Estas soluciones serán listas de secuencias (rutas) de los nodos del VRP. Las soluciones pueden ser generadas aleatoriamente o no, pero deben ser diversas pues de lo contrario el algoritmo convergería rápidamente pero obviando buena parte del espacio factible.
2.	<b>Evaluación:</b> Se evalúa la función objetivo $f$ en las soluciones obtenidas.

---

### Algoritmos Genéticos

---

3. **Selección:** Se escogen las mejores soluciones para realizarles los siguientes pasos.
  4. **Cruzamiento:** Se generan dos soluciones “hijas” a partir de cada par de las soluciones “padres” seleccionadas en el apartado anterior.
  5. **Mutación:** Se alteran de forma aleatoria algunas características de las soluciones “hijas” con el fin de abarcar otras zonas del espacio de soluciones factible. Ejemplos de mutaciones pueden ser permutaciones de dos o más nodos en una ruta, la inversión del orden de una ruta o subruta o la inserción de un nodo en una ruta distinta a la anterior
  6. **Reemplazo:** Se vuelven a aplicar los pasos 2,3,4 y 5 a las soluciones disponibles. El algoritmo se detiene cuando ya no se produzcan mejoras significativas en la población de soluciones o cuando se alcance una cantidad prefijada de iteraciones.
- 

Cuando los Algoritmos Genéticos se aplican a Problemas Dinámicos de Rutas de Vehículos [9] se pueden aplicar políticas similares a las de anteriores métodos: reevaluar el algoritmo conforme aparece la nueva información. No obstante, generar soluciones nuevas cada vez que se incluya una nueva observación puede ser algo lento y costoso. Si el grado de dinamismo es alto, no es conveniente empezar de nuevo cada vez que se actualice la información, el algoritmo podría no haber terminado de ejecutarse antes de la nueva inserción de datos. En estos casos las mejores soluciones de la última “generación” previa a la interrupción por la actualización pueden modificarse para tener en cuenta la nueva información y continuar con la optimización mientras los vehículos reciben actualizaciones provisionales de sus rutas. El dinamismo puede ser tenido en cuenta también en la función objetivo, que puede modificarse para premiar aquellas soluciones que son más adaptables a la aparición de nuevas soluciones. Para esto es conveniente poseer información estocástica sobre la aparición de nuevos pedidos.

### 3.2.3. Búsqueda Tabú

- Se trata de una técnica metaheurística que parte de múltiples **soluciones factibles iniciales** que se van mejorando de forma iterativa.
- Para evitar estancarse en **óptimos locales** la solución de una etapa no tiene que tener necesariamente menor costo que la de la anterior. Se definen umbrales para que la nueva solución no esté por debajo del umbral definido en la etapa anterior.
- La principal característica de esta técnica es que **almacena información a corto plazo** para mejorar la búsqueda de la solución. En la memoria, llamada **lista tabú** se almacenan tanto soluciones (**soluciones tabú**) como modificaciones a realizar a las mismas (**movimientos tabú**). La búsqueda de una nueva solución se realizará entre aquellas cercanas a la solución actual que no sean soluciones tabú, se les denomina **soluciones admisibles**.

- Se enumeran los arcos, en cada paso se realiza un movimiento, consistente en eliminar dos arcos y añadir otros dos. En cada iteración se añaden a la lista tabú los dos arcos que se han creado.
- La lista tabú tiene un **espacio limitado**. Se establece un criterio que elimine de la lista los pares de arcos de más antigüedad conforme se van añadiendo los nuevos.
- Cuando se alcanza cierto número de etapas, o cuando el **proceso no produce una mejora** tras cierta cantidad de iteraciones, el proceso se detiene.

# Capítulo 4

## El Problema de Reequilibrio del Sistema de Bicicleta Pública: BiciMAD

### 4.1. Presentación del problema

Con el fin de ofrecer una alternativa de movilidad urbana sostenible muchas ciudades ofrecen un **Sistema de Bicicleta Pública**, también llamado **Sistema de Bicicletas Compartidas** (**BSS**, del inglés *Bike Sharing System*). En la mayor parte de estas ciudades cuenta con un número de estaciones repartidas por la ciudad. Cada estación dispone de una cantidad de enganches que pueden tener o no acopladas bicicletas. Cada usuario del sistema puede desenganchar una bicicleta de una estación con bicicletas disponibles y desplazarse hasta otra estación con plazas libres donde depositarla.

El primer Sistema de Bicicleta Pública tuvo lugar en Ámsterdam en 1965, consistió en una flota de bicicletas blancas a disposición de los ciudadanos que podían recoger y depositar donde desearan, sin ningún medio para inmovilizarlas.

Con el objetivo de reducir los robos de bicicletas se implantó en Copenhague un **sistema de bloqueo** de las bicicletas mediante reembolso de dinero. Este BSS introdujo una cuota anual de usuarios y se convirtió en el primer BSS usado de forma masiva por los ciudadanos.

En 1998 en Rennes nació la tercera generación de Sistemas de Bicicleta Pública, muy similar a los modelos existentes hoy en día. Este sistema incluía estaciones con **enganches automáticos** e **información en tiempo real** sobre la disponibilidad de bicicletas y plazas libres.

Desde entonces el número de Sistemas de Bicicleta Pública ha aumentado en todo el mundo. Según la Federación Europea de Ciclistas [26] en 2013 había más de 535 BSS repartidas por el mundo, con aproximadamente 410 de ellas en Europa.

En España, según el Observatorio de la Bicicleta Pública en España [27] en el año 2010 se alcanzó del máximo de 100 Sistemas de Bicicletas Compartidas. Desde entonces, el número se ha reducido a 52 BSS en 2018. No obstante, el número total de estaciones y bicicletas ha crecido de forma continuada. Esto es debido a que en los Sistemas de Bicicletas Compartidas han fracasado en localidades pequeñas, mientras que en las grandes urbes su implementación ha tenido éxito y las redes de estaciones han aumentado su tamaño con los años. En 2018 el Sistema de Bicicleta Pública de Madrid, **BiciMAD**, contaba con 264 estaciones, 6.315 anclajes y 2.964 bicicletas.

Tal y como explican Amico et. al. [5] la dinámica de recogida y entrega de bicicletas varía en función de la estación y la hora del día:

- Los usuarios tienden a extraer bicicletas de zonas elevadas y entregarlas en zonas llanas, mientras que para trayectos inversos se prefieren otros medios de transporte urbano.
- En zonas residenciales los usuarios demandan bicicletas al inicio de la jornada y espacios libres cuando vuelven de trabajar.
- De forma complementaria, en áreas que albergan hospitales, oficinas, universidades o zonas comerciales se demandan plazas libres al inicio de la jornada laboral y bicicletas al final.
- En estaciones céntricas o cercanas a grandes nodos de comunicación la demanda de bicicletas y plazas libres es alta a lo largo de todo el día.

Esta diferencia en la dinámica de uso del Sistema Público de Bicicletas es observable en la ciudad de Madrid. A continuación se muestra un gráfico en el que se visualizan las **extracciones** de bicicletas de estaciones de BiciMAD en ocho distritos de Madrid en un mismo mes. Cada línea de puntos localizada dentro de un distrito representa las extracciones de bicicletas en uno de los barrios de ese distrito.

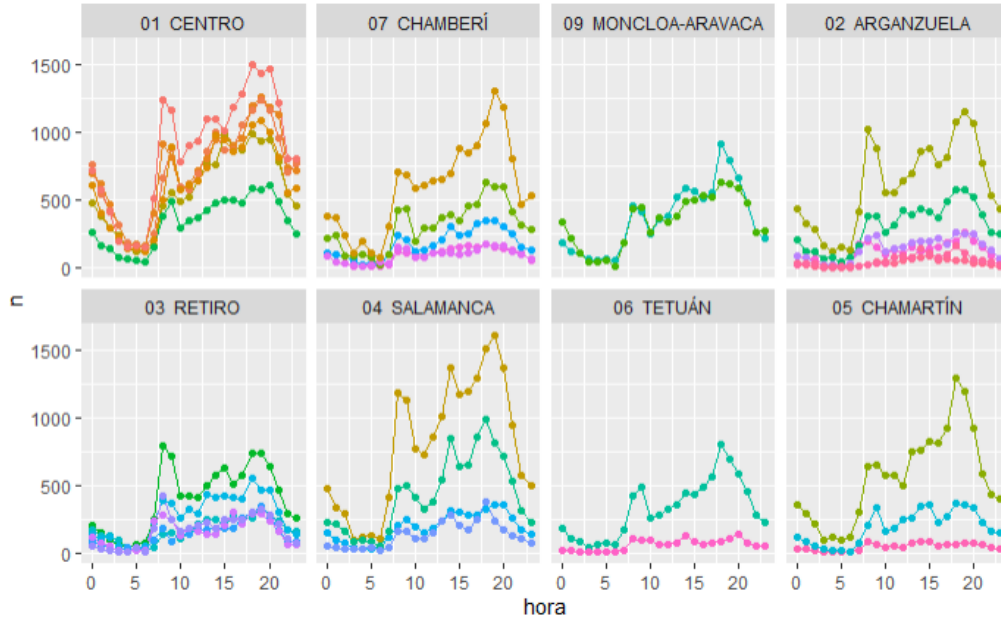


Figura 4.1: Demanda mensual por Distrito y Barrio

Observamos que cuando se extrae mayor número de bicicletas es en torno a las siete de la tarde en casi todos los barrios de Madrid. Al comienzo de la jornada, en torno a las ocho de la mañana también observamos repuntes en la demanda de bicicletas, si bien su magnitud relativa respecto al pico de la tarde varía en función del barrio en el que nos encontremos.

Fijémonos en algunas estaciones concretas. Por ejemplo, seleccionemos una estación de BiciMAD del barrio residencial Adelfas, en el distrito de El Retiro:

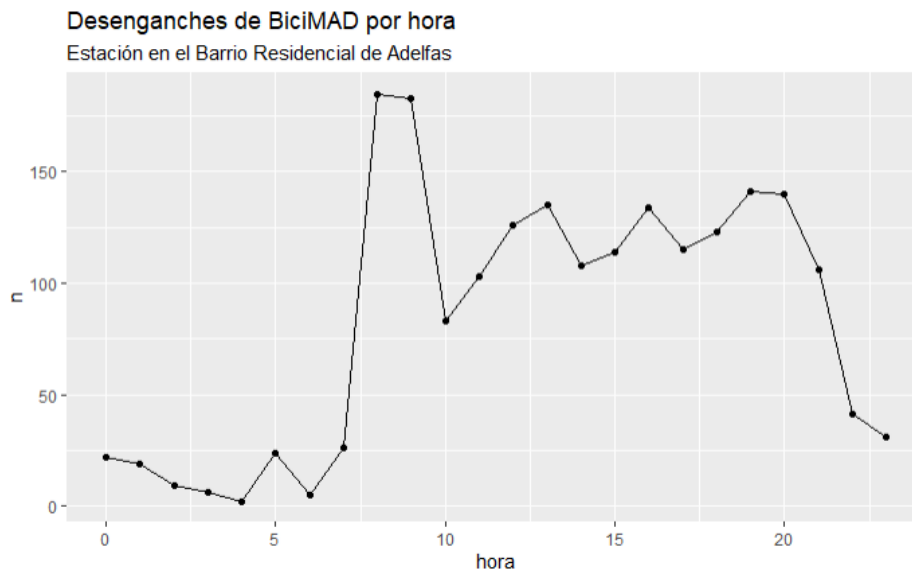


Figura 4.2: Demanda mensual: Barrio Residencial

Observamos que, efectivamente, la estación sigue un patrón de zona residencial. El máximo de demanda se alcanza en torno a las ocho o nueve de la mañana, cuando los

vecinos extraen bicicletas de la estación para desplazarse a sus lugares de trabajo. Un patrón muy distinto sigue la siguiente estación, situada muy cerca del Hospital Universitario Gregorio Marañón, en el barrio de Ibiza, también en el distrito de El Retiro. Fijémonos que a primera hora de la jornada son muy pocas (en comparación) las personas que demandan bicicletas. De hecho se extraen incluso más bicicletas a medianoche que en dicha hora punta:

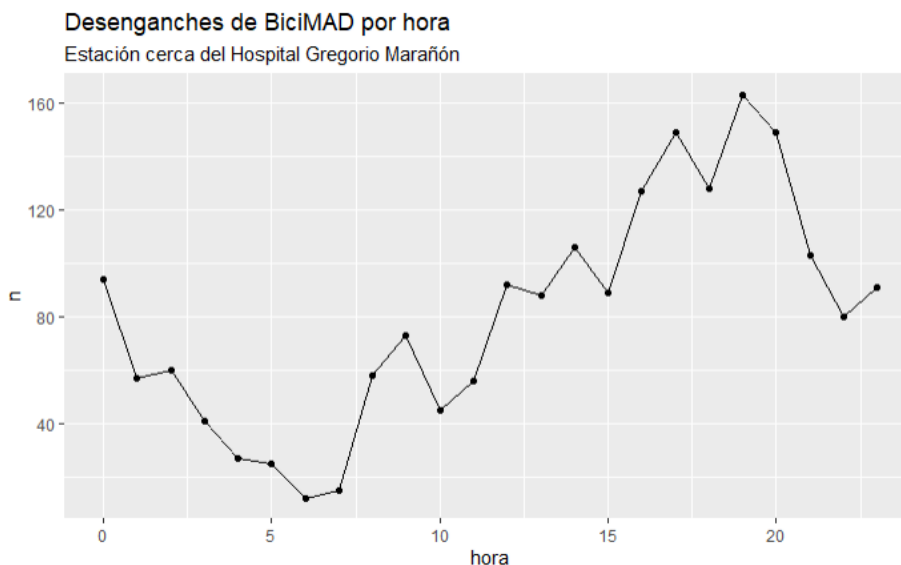


Figura 4.3: Demanda mensual: Hospital

Por último, fijémonos en el patrón de extracción de bicicletas de una estación de BiciMAD cercana a la Estación de Trenes de Atocha, en el barrio del mismo nombre, en el distrito de Arganzuela. Vemos que se extraen más bicicletas que en las anteriores estaciones. Esto no sólo es debido a la alta demanda de bicicletas en esa zona sino también a la alta demanda de enganches libres, que posibilita una mayor capacidad de extracción de bicicletas. Vemos también que la demanda se mantiene alta a lo largo del día, con los principales picos situados en las horas puntas.



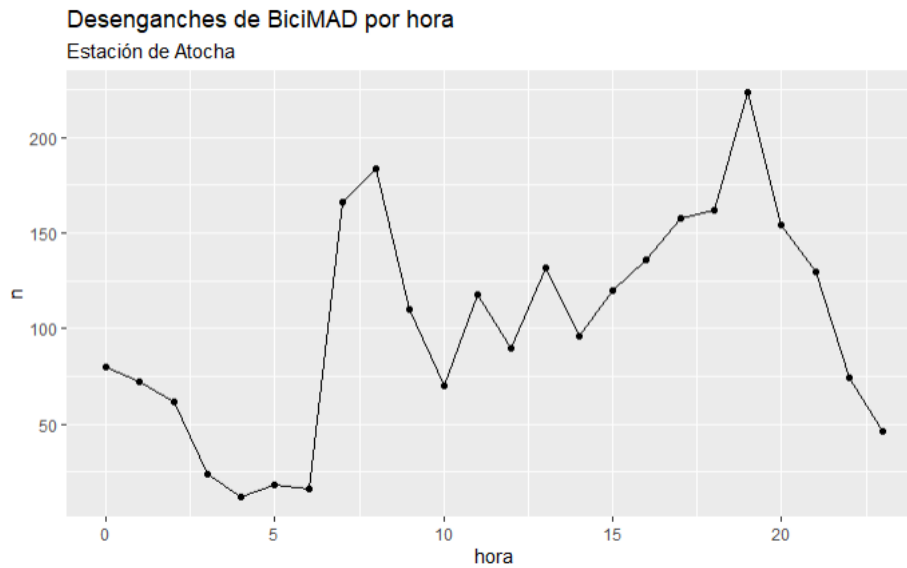


Figura 4.4: Demanda mensual: Nodo de comunicación

Así, se tiene que si no se interviene en el sistema, las estaciones sufrirán escasez en horas de mayor demanda de bicicletas y estarán llenas cuando la mayor parte de los usuarios deseen depositar sus bicicletas. Con el fin de mitigar este problema las empresas u organismos públicos que ofrecen el servicio cuentan con empleados que trasladan bicicletas de una estación a otra, así como del depósito a las estaciones y viceversa. Surge así el **Problema de Reequilibrio del Sistema de Bicicletas Compartidas (BRP**, del inglés *Bike Sharing Rebalancing Problem*), que pretende diseñar la mejor ruta posible para los empleados que trasladan las bicicletas. El objetivo puede ser minimizar los tiempos o costes de transporte [5] o minimizar el número de veces que un usuario trata de coger una bicicleta y la estación está vacía o trata de dejar una bicicleta y la estación está llena [1]. El **BRP** es un **Problema de Rutas de Vehículos con Recogida y Entrega** (*Pick and Delivery Problem*)

En este trabajo se analiza un Problema de Reequilibrio del Sistema de Bicicleta Pública de la ciudad de Madrid: BiciMAD

## 4.2. Formulación lineal

A continuación formularemos el Problema de Reequilibrio del Sistema de Bicicleta Pública desde el punto de vista de la programación lineal. Teniendo en consideración el dinamismo el problema puede ser evaluado de forma periódica de forma que se obtengan rutas que se adapten a la información que se obtiene en el transcurso de las mismas, imponiendo que las nuevas rutas que se calculan sigan el trayecto que los vehículos ya han recorrido.

### 4.2.1. Definiciones previas

Los nodos  $v_i$  se corresponden con las estaciones de BiciMAD y el coste  $c_{ij}$  se corresponderá con los tiempos de desplazamiento de los vehículos transportadores de bicicletas entre una estación y otra. Los nodos de la forma  $v_{0_i}$  se corresponden con cada uno de los  $K$  depósitos de biciletas que posee la Empresa Municipal de Transportes de Madrid.

Para la demanda de bicicletas de cada estación podemos tomar dos definiciones distintas:

- *Primera definición*

Podemos considerar que nuestro objetivo es evitar que las estaciones se vacíen o se llenen demasiado, es decir, que la proporción de plazas ocupadas de cada estación esté dentro de unos umbrales. Si llamamos  $p_i$  al número de plazas de  $v_i$ , nos interesa que el número de bicicletas de esa estación no sea inferior a  $\lceil \alpha \cdot p_i \rceil$  ni superior a  $\lfloor (1 - \alpha) \cdot p_i \rfloor$  con  $0 < \alpha < 0,5$ .

En este caso la demanda de cada estación se correspondería con las bicicletas que se deben enganchar o extraer para que la proporción de plazas libres esté contenida en  $(\alpha, 1 - \alpha)$ .

Cabe destacar que la demanda puede ser negativa, esto significa que los trabajadores deben retirar bicicletas de la estación porque está demasiado llena. Denotando a  $b_i$  como el número de bicicletas en la estación  $v_i$  y a  $d_i$  como la demanda de la estación tenemos la siguiente expresión.

$$d_i = \begin{cases} \lceil 0,5 \cdot p_i \rceil - b_i & \text{si } \lceil \alpha \cdot p_i \rceil > b_i \quad \text{ó} \quad \lfloor (1 - \alpha) \cdot p_i \rfloor < b_i \\ 0 & \text{cc} \end{cases} \quad (4.1)$$

Vemos que algunos nodos pueden tener demanda nula. Definimos el conjunto  $\mathcal{V}^*$  como el subconjunto  $\mathcal{V}^* \subset \mathcal{V}$  tal que si  $v_i \in \mathcal{V}^*$  entonces  $d_i \neq 0$ . Se entiende que los depósitos están incluidos de este conjunto

Definimos también los conjuntos  $\mathcal{V}_0$  como el conjunto de los depósitos y a  $\mathcal{V}'$  como el conjunto de las estaciones (sin contar con los depósitos).

Esta definición podría alterarse sustituyendo el valor  $0,5$  por  $1 - \alpha$  si la estación presenta un exceso de bicicletas y  $\alpha$  si presenta escasez. En ese caso los vehículos proveerían o extraerían la mínima cantidad de bicicletas necesarias para considerar que la capacidad de una estación está en el intervalo  $((1 - \alpha) \cdot p_i, \alpha \cdot p_i)$

- *Segunda definición*

Haciendo uso de los datos proporcionados por el Ayuntamiento de Madrid [25] podemos obtener información estocástica sobre la frecuencia con la que los usuarios demandan bicicletas o plazas libres en cada estación y hora del día. Denotemos  $\omega_i(t)$  a la demanda estimada de bicicletas de los usuarios para la estación  $v_i$  e instante de tiempo  $t$  concretos. Podemos establecer entonces que hay demanda de bicicletas o huecos libres si la diferencia entre la demanda estimada de los usuarios en un cierto intervalo temporal y las bicicletas presentes en ese intervalo son mayores que una cierta proporción de las plazas disponibles. Al igual que en el caso anterior, la demanda es positiva si se demandan bicicletas o negativa si se demandan huecos libres. Puede ser que los usuarios demanden más bicicletas que plazas posea la estación pero la demanda de cada estación no puede ser superior en valor absoluto al número de plazas.

$$d_i = \begin{cases} \omega_i - b_i & \text{si } \lfloor \beta \cdot p_i \rfloor < |\omega_i - b_i| \text{ y } |\omega_i| \leq p_i \\ p_i - b_i & \text{si } \omega_i > p_i \\ b_i - p_i & \text{si } \omega_i < -p_i \\ 0 & \text{cc} \end{cases} \quad (4.2)$$

Tanto  $\omega_i$  como  $d_i$  pueden estar asociadas a intervalos de tiempo en vez de a momentos puntuales.

El inconveniente de esta definición de demanda es que es difícil estimar  $\omega_i(t)$ . Una opción es hallar el flujo medio de bicicletas en cada estación e intervalo de tiempo, pero no es una buena solución del todo, puesto que hay una diferencia entre las inserciones y extracciones de bicicletas que se realizan cada día y la demanda real. El motivo es que los usuarios del sistema pueden ser disuadidos de efectuar su ruta inicial si la estación de la que quieren tomar bicicletas está vacía o la estación a la que pretenden llegar está demasiado llena. Los usuarios disponen de información en tiempo real de la situación de cada estación pero no se dispone información sobre la intención inicial de los usuarios.

Ambas definiciones de la demanda podrían combinarse, de forma que se evite tanto que una estación se llene o se vacíe como que el número de bicicletas disponibles se aleje demasiado de la demanda de los usuarios.

Además de los datos ya presentados, contamos con el parámetro  $h_{ik} \in \{0, 1\}$  con  $i \in \mathcal{V}_0$  y  $k \in \{1, 2, \dots, m\}$ . Se tiene que  $h_{ik} = 1$  si el vehículo  $k$  empieza el recorrido en la estación  $v_i$  y 0 en caso contrario. Este parámetro permite definir el conjunto  $H_k$ ,  $k \in K$ , que contiene únicamente al depósito en el que empieza el vehículo  $k$ .

Definimos una variable binaria  $x_{ijk}$  con  $i, j \in \mathcal{V}$  y  $k \in \{1, 2, \dots, m\}$  que toma el valor 1 si el vehículo  $k$  se desplaza de la estación  $v_i$  a la  $v_j$  sin pasar por estaciones intermedias y 0 en caso contrario.

Definimos también la variable entera positiva flujo  $f_{ijk}$  de bicicletas del vehículo  $k$  en el arco  $(i, j)$ , es decir, la cantidad de bicicletas que el vehículo  $k$  carga de una estación a otra.

Por último, definimos la variable auxiliar entera no negativa  $u_{ik}$  con  $i \in \mathcal{V}$  y  $k \in K$  que guarda relación con el orden en el que el vehículo  $k$  recorre la estación  $i$ . Si la variable toma el valor 0 es que no recorre esa estación

A continuación presentamos una tabla resumiendo los parámetros y variables del problema:

Tabla 4.1: Resumen de la notación

<b>Notación BRP</b>	
$\mathcal{V}$	Conjunto de estaciones y depósitos de BiciMAD.
$\mathcal{V}'$	Conjunto de estaciones de BiciMAD.
$\mathcal{V}_0$	Conjunto de depósitos de BiciMAD.
$\mathcal{E}$	Conjunto de recorridos entre estaciones y depósitos.
$\mathcal{C}$	Matriz de costes de desplazamiento entre estaciones y depósitos.
$n$	Número de estaciones ( $n \approx 264$ ).
$n_0$	Número de depósitos.
$K$	Conjunto de vehículos transportadores de bicicletas de la flota.
$m$	Número de vehículos transportadores de bicicletas de la flota.
$Q_k$	Capacidad del vehículo $k$ .
$h_{ik}$	Parámetro de los datos que toma el valor 1 si el vehículo $k$ comienza en la estación $i$ y 0 en caso contrario.
$H_k$	Conjunto que contiene al depósito en el que comienza el vehículo $k$
$p_i$	Enganches de cada estación.
$b_i$	Bicicletas en cada estación.
$d_i$	Demanda de bicicletas de cada estación.
$\mathcal{V}^*$	Conjunto de estaciones de BiciMAD con demanda no nula.
$\omega_i(t)$	Demanda estimada de bicicletas de los usuarios para cada estación.
$x_{ijk}$	Variable que toma el valor 1 si el vehículo $k$ atiende a $v_i$ y seguidamente a $v_j$ y cero en caso contrario.
$f_{ijk}$	Flujo de bicicletas entre las estaciones $i$ y $j$ por el vehículo $k$ .
$u_{ik}$	Variable auxiliar que expresa el orden en el que el vehículo $k$ recorre dos estaciones consecutivas.

*Nota:* En lo sucesivo se comete un abuso de notación identificando  $i$  con  $v_i$ . La forma correcta de expresar  $i \in \mathcal{V}^*$  sería  $v_j \in \mathcal{V}^*$ .

### 4.2.2. Formulación 1

Podemos considerar que la función objetivo debe buscar reducir los tiempos de desplazamiento de los operarios. En ese caso la función objetivo es la siguiente.

$$\min \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} \sum_{k \in K} c_{ij} \cdot x_{ijk} \quad (4.3)$$

#### Restricciones

$$\sum_{i \in \mathcal{V}} \sum_{k \in K} x_{ijk} \leq 1, \quad \forall j \in \mathcal{V}' \quad (4.4)$$

$$\sum_{i \in \mathcal{V}} \sum_{k \in K} x_{jik} \leq 1, \quad \forall j \in \mathcal{V}' \quad (4.5)$$

$$\sum_{j \in \mathcal{V}} x_{ijk} = \sum_{j \in \mathcal{V}} x_{jik} \quad \forall k \in K, \quad \forall i \in \mathcal{V} \quad (4.6)$$

$$1 \leq \sum_{j \in \mathcal{V}} x_{ijk} \quad i \in H_k, \quad \forall k \in K \quad (4.7)$$

$$\sum_{j \in \mathcal{V}} x_{ijk} = 0 \quad \forall i \in \mathcal{V}_0 \setminus H_k, \quad \forall k \in K \quad (4.8)$$

$$\sum_{i \in \mathcal{V}} x_{jik} = 0 \quad \forall i \in \mathcal{V}_0 \setminus H_k, \quad \forall k \in K \quad (4.9)$$

$$u_{ik} - u_{jk} + n \cdot x_{ijk} \leq n - 1 \quad \forall i, j \in \mathcal{V}', \quad i \neq j, \quad \forall k \in K \quad (4.10)$$

$$\sum_{i \in \mathcal{V}} \sum_{k \in K} f_{ijk} - \sum_{i \in \mathcal{V}} \sum_{k \in K} f_{jik} = d_j \quad \forall j \in \mathcal{V}' \quad (4.11)$$

$$x_{ijk} \in \{0, 1\}, \quad \forall i, j \in \mathcal{V}, \quad k \in K \quad (4.12)$$

$$\max \{0, -d_i, d_j\} x_{ijk} \leq f_{ijk} \leq \min \{Q, Q - d_i, Q + d_j\} x_{ijk} \quad \forall i, j \in \mathcal{V}, \quad k \in K \quad (4.13)$$

$$-b_j \leq \sum_{i \in \mathcal{V}} \sum_{k \in K} f_{ijk} - \sum_{i \in \mathcal{V}} \sum_{k \in K} f_{jik} \leq p_j - b_j \quad \forall j \in \mathcal{V}_0 \quad (4.14)$$

$$1 \leq u_{ik} \leq n, \quad \forall i \in \mathcal{V}' \quad \forall k \in K \quad (4.15)$$

- Las restricciones 4.4 y 4.5 garantizan que cada estación sea atendida por un vehículo a lo sumo una vez.
- La restricción 4.6 impone continuidad en las rutas que pasan por los nodos, es decir, que cada vehículo que llegue a una estación o vehículo debe partir a otro nodo y viceversa.
- La restricción 4.7 se asegura de que cada vehículo comience su ruta en su depósito correspondiente, la restricción 4.6 se asegurará de que el vehículo termine su ruta en su depósito. Esto no impide que un vehículo reposte en su depósito y vuelva a salir. Tal y como está formulado se impone que todos los vehículos deben usarse. Esto puede parecer lo más natural pero la solución óptima no precisa necesariamente de todos los vehículos. Para solucionar esto se podrían crear  $n_0$  depósitos auxiliares cada uno asociado a un depósito ya existente de forma que el coste de desplazamiento entre un depósito y su auxiliar sean nulos y los costes de desplazamiento desde los depósitos auxiliares a nodos distintos de su depósito asociado sean muy altos. Si en la solución del problema un vehículo se desplaza al depósito auxiliar y vuelve y no hace nada más se debe interpretar que no se ha movido.
- Con 4.8 y 4.9 se impone que un vehículo no puede ir a depósitos distintos del que tiene asociado. Realmente, con 4.6 basta con cualquiera de las dos para imponer esta condición.
- La expresión 4.10 es una modificación de la de **Miller-Tucker-Zemlin** [19] adaptada a las características de este problema. Esta restricción impone que en dos estaciones consecutivas en la ruta de un vehículo la variable  $u_{ik}$  debe tomar un valor menor en la estación saliente que en la estación entrante. De esta manera se impide que existan rutas cíclicas que no incluyan al depósito.
- En la expresión 4.11 se está imponiendo que se satisfaga la demanda de los nodos que se visitan.
- La expresión 4.12 define la variable  $x_{ijk}$ .
- La restricción 4.13 define y acota a  $f_{ij}$ : Si no hay conexión entre dos estaciones el flujo de bicicletas es nulo. La cantidad de bicicletas que un vehículo transporta no puede exceder de su capacidad o ser negativa. Además, el vehículo debe haber satisfecho la demanda de la estación  $i$ , por lo que si  $i$  demandaba bicicletas no podrá tener más que la capacidad total menos las bicicletas que ha repuesto en  $i$ , y si demandaba que le liberaran enganches, el vehículo debe contar, por lo menos, con esas bicicletas que ha desenganchado. De forma equivalente, el vehículo debe estar preparado para asumir las bicicletas que a  $j$  le sobren o a proporcionar las bicicletas que le falten.
- La restricción 4.14 tiene utilidad únicamente si algún depósito no tiene mucha capacidad o tiene demasiado exceso o defecto de bicicletas. Así aseguramos que el depósito no reciba bicicletas de forma que se exceda su capacidad, ni se le sustraigan bicicletas de forma que se obtenga una cantidad negativa. Esta restricción impide estos escenarios al final de la ejecución del problema, no en el transcurso. No obstante ese problema puede solventarse alterando el orden de salida del depósito de los vehículos. No es necesario aplicar esta restricción a las estaciones pues las anteriores ya aseguran que no sobrepasarán su capacidad ni quedarán con “bicicletas negativas”

- La expresión 4.15 define y acota la variable  $u_{ik}$ . Si cada vehículo tuviera que recorrer todos los nodos entonces  $u_{ik}$  sería una variable que indicaría el orden en el que se recorren las estaciones. Sin embargo, como no tenemos esta imposición la variable nos puede servir para saber si una estación se visita antes que otra adyacente, pero la diferencia de los valores de  $u_{ik}$  entre dos nodos consecutivos no necesariamente es 1 y los valores de  $u_{ik}$  pueden repetirse.

### 4.2.3. Formulación 2

En esta segunda formulación se busca maximizar la satisfacción global de la demanda y se incluirán restricciones sobre los costes. Se trata de un planteamiento más realista y adaptado a la duración de los turnos de los trabajadores. Se define un nuevo parámetro  $T$  que representa el costo máximo asociado a cada vehículo. La función objetivo sería la siguiente

$$\min \sum_{j \in \mathcal{V}'} \left| \sum_{i \in \mathcal{V}} \sum_{k \in K} f_{ijk} - \sum_{i \in \mathcal{V}} \sum_{k \in K} f_{jik} - d_j \right|$$

No obstante, incluir valores absolutos en la formulación del problema rompe la linealidad. Debemos modificar la formulación del problema. Sustituiremos  $|\sum_{i \in \mathcal{V}} \sum_{k \in K} f_{ijk} - \sum_{i \in \mathcal{V}} \sum_{k \in K} f_{jik} - d_j|$  por  $y_j$

Función objetivo:

$$\min \sum_{j \in \mathcal{V}'} y_j \quad (4.16)$$

Restricciones

$$\sum_{i \in \mathcal{V}} \sum_{k \in K} x_{ijk} \leq 1, \quad \forall j \in \mathcal{V}' \quad (4.17)$$

$$\sum_{i \in \mathcal{V}} \sum_{k \in K} x_{jik} \leq 1, \quad \forall j \in \mathcal{V}' \quad (4.18)$$

$$\sum_{j \in \mathcal{V}} x_{ijk} = \sum_{j \in \mathcal{V}} x_{jik} \quad \forall k \in K, \quad \forall i \in \mathcal{V} \quad (4.19)$$

$$1 \leq \sum_{j \in \mathcal{V}} x_{ijk} \quad i \in H_k, \quad \forall k \in K \quad (4.20)$$

$$\sum_{j \in \mathcal{V}} x_{ijk} = 0 \quad \forall i \in \mathcal{V}_0 \setminus H_k, \quad \forall k \in K \quad (4.21)$$

$$\sum_{i \in \mathcal{V}} x_{jik} = 0 \quad \forall i \in \mathcal{V}_0 \setminus H_k, \quad \forall k \in K \quad (4.22)$$

$$u_{ik} - u_{jk} + n \cdot x_{ijk} \leq n - 1 \quad \forall i, j \in \mathcal{V}', \quad i \neq j, \quad \forall k \in K \quad (4.23)$$

$$\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} c_{ij} \cdot x_{ijk} \leq T \quad \forall k \in K \quad (4.24)$$

$$x_{ijk} \in \{0, 1\}, \quad \forall i, j \in \mathcal{V}, \quad k \in K \quad (4.25)$$

$$0 \leq f_{ijk} \leq Q \cdot x_{ijk} \quad \forall i, j \in \mathcal{V}, \quad k \in K \quad (4.26)$$

$$-b_j \leq \sum_{i \in \mathcal{V}} \sum_{k \in K} f_{ijk} - \sum_{i \in \mathcal{V}} \sum_{k \in K} f_{jik} \leq p_j - b_j \quad \forall j \in \mathcal{V} \quad (4.27)$$

$$1 \leq u_{ik} \leq n, \quad \forall i \in \mathcal{V}' \quad \forall k \in K \quad (4.28)$$

$$\sum_{i \in \mathcal{V}} \sum_{k \in K} f_{ijk} - \sum_{i \in \mathcal{V}} \sum_{k \in K} f_{jik} - d_j \leq y_j \quad \forall j \in \mathcal{V}' \quad (4.29)$$

$$-\sum_{i \in \mathcal{V}} \sum_{k \in K} f_{ijk} + \sum_{i \in \mathcal{V}} \sum_{k \in K} f_{jik} + d_j \leq y_j \quad \forall j \in \mathcal{V}' \quad (4.30)$$

- En esta formulación la restricción 4.24 limita el coste total de desplazamiento de cada vehículo.
- La restricción 4.26 acota a  $f_{ijk}$ . Impone que la carga no sea negativa ni sobrepase la capacidad del vehículo. Esta restricción no es igual a 4.13 porque en esta formulación del problema el vehículo no debe satisfacer necesariamente la demanda de los vehículos por los que pasa.
- La restricción 4.27 realiza una función similar a la de la restricción 4.14 en el *apartado anterior* pero cumpliéndose para todas las estaciones en lugar de únicamente para los vértices. Impide que a cada estación se le suministren más bicicletas de las que puede albergar o que se le extraigan más bicicletas de las que tiene.
- Las restricciones 4.29 y 4.30 definen  $y_j$  como el valor absoluto de la satisfacción de la demanda de la estación  $j$ . Realmente acotan inferiormente la variable  $y_j$  pero como en la función objetivo 4.16 los coeficientes que acompañan a las  $y_j$  son todos positivos y es un problema de minimización las  $y_j$  tomarán el valor de la cota inferior, esto es, del valor absoluto.
- El resto de restricciones son las mismas que en el *apartado anterior*.



#### 4.2.4. Variantes

A continuación se plantearán distintas variaciones en el planteamiento del problema que afectan a las restricciones que tienen en común

##### 4.2.4.1. Variante 1: Estaciones asistidas por más de un vehículo

Si la capacidad de los vehículos es comparable al número de plazas de las estaciones es razonable pensar que un sólo vehículo no puede o es costoso para el problema satisfacer la demanda de una estación. Una variante a los problemas anteriormente formulados sería considerar que una estación puede ser asistida por varios vehículos, de forma que sea visitada a lo sumo una vez por cada vehículo. Las restricciones 4.4 y 4.5 deben sustituirse en ese caso por las siguientes:

$$\sum_{i \in \mathcal{V}} x_{ijk} \leq 1, \quad \forall j \in \mathcal{V}', \quad \forall k \in K \quad (4.31)$$

$$\sum_{i \in \mathcal{V}} x_{jik} \leq 1, \quad \forall j \in \mathcal{V}', \quad \forall k \in K \quad (4.32)$$

Este cambio en el planteamiento del problema obliga a hacer ciertos cambios. Para empezar, la restricción 4.13 en la primera formulación del problema habría de sustituirse por la restricción 4.26, puesto que como a una estación la abastecen más de un vehículo, cada uno de estos no está obligado a cumplir por sí solo las condiciones asociadas al abastecimiento de la demanda de los nodos  $i$  y  $j$ .

Además, tanto en la primera como en la segunda formulación del problema la restricción 4.14 o la 4.27 no serían suficientes: La siguiente restricción

$$-b_i \leq \sum_{i \in \mathcal{V}} \sum_{k \in K} f_{ijk} - \sum_{i \in \mathcal{V}} \sum_{k \in K} f_{jik} \leq p_j - b_j \quad \forall j \in \mathcal{V}$$

Impide que en el flujo entrante-saliente de bicicletas en una estación sea superior a los enganches libres que le quedaban o inferior a las bicicletas que tenía inicialmente, pero no impide, por ejemplo, un escenario en el que una estación sobrepase su capacidad a mitad de la ejecución del problema y posteriormente esto se corrija de forma que al finalizar la ejecución de las rutas la estación tenga un número coherente de bicicletas. Cuando por una estación sólo pasaba un vehículo sólo se alteraba su estado una vez y no se tenía este problema. Sí era posible que una situación así ocurriera para un depósito; no obstante, tal y como dijimos, es una situación poco probable debido a que a un depósito se le supone una capacidad y cantidad de bicicletas suficientemente altas y, si se diera, se puede solucionar fácilmente alterando el orden de salida de los vehículos del depósito.

En esta variante, alterar el orden de salida de los vehículos no necesariamente soluciona esta casuística, puesto que las rutas de dos o más vehículos podrían coincidir en dos o más puntos con problemas de exceso o defecto puntual de bicicletas. Siempre se podría

detener algún vehículo en mitad plena ruta a la espera de que otro vehículo llegue antes a la estación conflictiva y haga factible la recogida o entrega de bicicletas del vehículo que se detiene. Sin embargo, esta no es una solución aceptable ya que se estarían desperdiciando los minutos en los que el vehículo se detiene y en una casuística más compleja ni aun así la situación sería factible.

Para resolver este problema se sugiere incluir en ambas formulaciones la siguiente restricción:

$$-b_j \leq \sum_{k \in S} \left( \sum_{i \in \mathcal{V}} f_{ijk} - \sum_{i \in \mathcal{V}} f_{jik} \right) \leq p_j - b_j \quad \forall j \in \mathcal{V}, S \subseteq K \quad (4.33)$$

De esta manera se impide que en algún momento de la ejecución de la ruta una estación tenga una cantidad negativa de bicicletas o que exceda su capacidad. Cabe mencionar que esta expresión es algo más restrictiva de lo deseado, veámoslo con ejemplos:

- Al ejecutar un problema sin 4.33, a una estación con 3 bicicletas y 16 enganches un primer vehículo le sustrae 4 bicicletas y un segundo le añade 9, de forma que cuando las rutas finalizan la estación tiene 8 bicicletas, satisfaciendo su demanda. No obstante, si el primer vehículo llega antes que el segundo, la estación se quedaría puntualmente con -1 bicicletas y la situación carecería de sentido. Si se ejecuta el problema con 4.33 esta solución que realmente no es factible no se tendría puesto que, para  $S = \{1\}$  y  $j$  la estación en cuestión se tendría que:

$$\sum_{k \in S} \left( \sum_{i \in \mathcal{V}} f_{ijk} - \sum_{i \in \mathcal{V}} f_{jik} \right) = -4$$

y que  $-b_j = -3$  por lo que 4.33 no se cumple y la situación no podría darse.

- Al ejecutar otro problema distinto sin la restricción 4.33 a una estación con 12 bicicletas y 16 enganches un primer vehículo le añade 3 bicicletas, un segundo le sustrae 9 y un tercero le añade 2. Si los vehículos llegan en este orden la estación pasa de tener 12 a tener 15, 6 y 8 bicicletas en distintos momentos de la ejecución de las rutas, por lo que la solución tiene sentido. Si se incluye la restricción 4.33, para  $S = \{1, 3\}$  y  $j$  la estación en cuestión se tiene que:

$$\sum_{k \in S} \left( \sum_{i \in \mathcal{V}} f_{ijk} - \sum_{i \in \mathcal{V}} f_{jik} \right) = 5$$

y que  $p_j - b_j = 4$  por lo que 4.33 no se cumple y la situación, que es válida, no podría darse.

En resumen, vemos que la restricción puede eliminar soluciones válidas en casos en los que a una estación acudan más de dos vehículos pero garantiza que una estación no se sobrecargue o quede en números negativos, por lo que se recomienda usarla si en el problema se plantea que vehículos distintos accedan a una misma estación.

#### 4.2.4.2. Variante 2: Vehículos que repiten estaciones

Esta variante es una ampliación de la anterior. En esta ocasión una estación no sólo puede ser visitada por varios vehículos sino también varias veces por un mismo vehículo. La formulación del problema se haría de forma idéntica al apartado anterior con la excepción de la eliminación de 4.31 y 4.32 (entendiendo que tampoco estarían ni 4.4 ni 4.5).

Otra forma de plantearlo es considerar cada ciclo que comienza y termina en el depósito como la ruta de un vehículo. Se crearían así rutas de vehículos en el problema que corresponderían con subrutas cíclicas de un mismo vehículo en la realidad que pasan por el depósito. Se aplicarían las restricciones 4.31 y 4.32 tanto a las estaciones como a los vehículos, de forma que aseguremos que cada ruta es un ciclo. Quedarían de la siguiente manera:

$$\sum_{i \in \mathcal{V}} x_{ijk} \leq 1, \quad \forall j \in \mathcal{V}, \quad \forall k \in K \quad (4.34)$$

$$\sum_{i \in \mathcal{V}} x_{jik} \leq 1, \quad \forall j \in \mathcal{V}, \quad \forall k \in K \quad (4.35)$$

En la segunda formulación, además, debemos alterar las restricciones relacionadas con el límite de costes asociados a un mismo vehículo. La restricción 4.24 limita el coste total empleado por cada vehículo. Ahora nos interesa limitar el coste empleado por cada grupo de vehículos. Podemos denotar a  $G$  como el grupo de vehículos reales y como  $K_g$  al grupo de rutas cíclicas en torno al depósito asociadas al vehículo  $g \in G$ . Sustituiríamos entonces 4.24 por:

$$\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} \sum_{k \in K_g} c_{ij} \cdot x_{ijg} \leq T \quad \forall g \in G \quad (4.36)$$

El resto de restricciones serían las mismas que en la variante anterior

#### 4.2.4.3. Variante 3: Depósitos no iniciales en la ruta

En ambas formulaciones del problema hemos considerado que un vehículo no puede acudir a un depósito distinto del que tiene asociado. En un planteamiento alternativo podría interesar que los vehículos cargen o suelten bicicletas en depósitos distintos al inicial.

En ese caso debemos eliminar las restricciones 4.8 y 4.9. También debemos modificar las restricciones 4.10 y 4.15 de tal forma que se incluyan todos los depósitos excepto el de partida. Quedarían de la siguiente forma:

$$u_{ik} - u_{jk} + (n + n_0 - 1) \cdot x_{ijk} \leq n + n_0 - 2 \quad \forall i, j \in \mathcal{V} \setminus H_k, \quad i \neq j, \quad \forall k \quad (4.37)$$

$$1 \leq u_{ik} \leq n + n_0 - 1, \quad \forall i \in \mathcal{V} \setminus H_k \quad \forall k \in K \quad (4.38)$$

De esta manera la variable  $u_{ik}$  también toma valores asociados a los depósitos para evitar que se creen rutas cíclicas disconexas que no pasen por el depósito inicial. Debe entonces ampliar su rango posible de valores añadiendo el número de depósitos menos el inicial.

#### 4.2.4.4. Variante 4: Finalización en otro depósito

Otro aspecto a considerar es que los vehículos deban regresar a un depósito al terminar la ruta pero no necesariamente a su depósito inicial. Es algo que tiene sentido especialmente si se trata de una flota homogénea. En este planteamiento del problema la restricción 4.6 ya no puede aplicarse a los depósitos, aunque sí a las estaciones. se busca que pueda existir una discontinuidad en la ruta pero que sea a lo sumo una y que la ruta empiece en el nodo inicial asociado al vehículo, de forma que el vehículo sale de ese depósito una vez más de las veces que entra y finalice en un nodo distinto donde entra una vez más de las que sale. Esto implicaría, además de realizar los cambios indicados en la variante anterior, cambiar 4.6 por las siguientes restricciones:

$$\sum_{i \in \mathcal{V}_0} \sum_{j \in \mathcal{V}} x_{ijk} = \sum_{i \in \mathcal{V}_0} \sum_{j \in \mathcal{V}} x_{jik} \quad \forall k \in K \quad (4.39)$$

$$\sum_{j \in \mathcal{V}} x_{ijk} = \sum_{j \in \mathcal{V}} x_{jik} \quad \forall k \in K, \quad \forall i \in \mathcal{V}' \quad (4.40)$$

$$\sum_{j \in \mathcal{V}} x_{jik} \leq \sum_{j \in \mathcal{V}} x_{ijk} \quad \forall k \in K, \quad \forall i \in H_k \quad (4.41)$$

$$\sum_{j \in \mathcal{V}} x_{ijk} \leq \sum_{j \in \mathcal{V}} x_{jik} \quad \forall k \in K, \quad \forall i \in \mathcal{V}_0 \setminus H_k \quad (4.42)$$

$$\sum_{j \in \mathcal{V}} x_{ijk} \leq 1 + \sum_{j \in \mathcal{V}} x_{jik} \quad \forall i \in H_k, \quad \forall k \in K \quad (4.43)$$

- La restricción 4.39 asegura que los vehículos que salgan de los depósitos regresen finalmente a los mismos. Nótese que no se impone que un vehículo deba regresar al depósito donde comenzó, pero sí que regrese a alguno.
- La restricción 4.40 impone continuidad en las rutas que pasan por las estaciones, es decir, que cada vehículo que llegue a una estación debe salir de ella.
- Las restricciones 4.41 y 4.42 realizan para los depósitos una función parecida a la que realiza la restricción 4.40 con las estaciones. La diferencia estriba en que una ruta debe finalizar en un depósito pero no necesariamente en el que comenzó. Con 4.41 expresamos que si el depósito  $i$  es el inicial del vehículo  $k$ , entonces el vehículo saldrá del depósito una cantidad igual o mayor de veces de las que entrará. Por otro lado 4.42 expresa que si el depósito no es el inicial, entonces el vehículo entrará un número igual o superior de veces de las que saldrá.

- Falta imponer de alguna manera que sólo puede romperse la continuidad de la ruta a lo sumo una vez. Esto es, no puede tenerse que un vehículo llegue a un depósito distinto del inicial y a continuación salga del inicial. Con 4.43 limitamos que el número de veces que un vehículo sale de su nodo inicial no puede ser mayor en más de una unidad que el número de veces que el vehículo entra. Combinada con las restricciones anteriores se impone que las veces que un vehículo entra pero no sale de un depósito que no es su lugar de comienzo no puede ser mayor que la unidad.

#### 4.2.4.5. Variante 5: Restricción de Dantzig–Fulkerson–Johnson

En este problema se ha impuesto que no se creen rutas cíclicas que no pasen por depósitos. Hay otras maneras de imponer esa misma condición [19]. Una de las más conocidas es la restricción de **Dantzig–Fulkerson–Johnson**, que expresamos a continuación

$$\sum_{i \in S} \sum_{j \in S} x_{ijk} \leq |S| - 1, \quad S \subseteq \mathcal{V}', S \neq \emptyset \quad \forall k \in K \quad (4.44)$$

Se impone que para cada subconjunto de las estaciones el número de aristas existentes debe ser menor que el cardinal del subconjunto menos la unidad. Si esta restricción no se cumpliera, se tendría que al menos en un subconjunto el número de aristas es igual al número de estaciones. Como cada estación sólo puede tener dos aristas, una de salida y otra de entrada, se tendría que las estaciones de ese subconjunto son un ciclo. Si en el subconjunto escogido el número de aristas es superior a  $|S|$ , entonces se tendría más de un ciclo y se puede buscar otro subconjunto cíclico.

La desventaja de usar esta restricción es la elevada cantidad de subconjuntos que se deben generar, habiendo de evaluar en cada uno la expresión de la restricción. Para nuestro problema el número de subconjuntos a calcular, habiendo unas 264 estaciones, podría ascender a  $|\mathcal{P}(\mathcal{V}')| = 2^{|\mathcal{V}'|} = 2^{264} \approx 2.96428 \cdot 10^{79}$ .

Con la restricción de Miller-Tucker-Zemlin, en cambio, el número de variables nuevas se reduce a  $n$  y el de restricciones a  $\frac{n^2}{2}$ . En definitiva, es mejor mantener 4.10.

#### 4.2.4.6. Variante 6: Problema multiobjetivo

Las funciones objetivo se pueden combinar de forma que se busque una solución que equilibre la satisfacción de la demanda con el tiempo empleado por los trabajadores. Una opción posible sería, con las restricciones de *la segunda formulación*, plantear la siguiente función objetivo

$$\min \quad \alpha \cdot \left( \sum_{j \in \mathcal{V}'} y_j \right) + \beta \cdot \left( \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} \sum_{k \in K} c_{ij} \cdot x_{ijk} \right) \quad (4.45)$$

Siendo  $\alpha$  y  $\beta$  números reales que se fijan según la importancia de un aspecto u otro del problema.

## 4.3. Resolución Lineal del BRP: BiciMAD

### 4.3.1. Descripción de la implementación

Se ha escogido para la implementación de este problema el Sistema de Bicicleta Pública de la ciudad de Madrid debido a la buena disponibilidad de sus datos en el *Portal de Datos Abiertos del Ayuntamiento de Madrid*. En dicha página se encuentran disponibles los movimientos de bicicletas efectuados de una parada a otra recogidos de mes en mes, así como información sobre las plazas disponibles y la dirección y situación geográfica de cada estación.

A continuación presentaremos tablas con la información que se puede extraer del Portal de Datos Abiertos. En primer lugar mostraremos la selección de algunas estaciones con la información que podemos obtener de ellas: nombre, plazas totales de bicicletas, códigos que la identifican, su latitud y longitud y su dirección. Esta información data del año 2018.

Tabla 4.2: Extracto Estaciones Bicimad

name	total_bases	number	id	longitude	latitude	address
Agustín de Betancourt	24	134	153	-3.695605	40.44403	Calle María de Guzmán nº 58
Alberto Alcocer	24	158	138	-3.684715	40.45853	Avenida de Alberto Alcocer nº 22
Alcalá	27	97	102	-3.680131	40.42269	Avenida de Menéndez Pelayo nº 3
Alcántara	24	99	104	-3.673871	40.42619	Calle Alcántara nº 2
Almadén	24	67	71	-3.693225	40.41085	Calle Almadén nº 28
Alonso Martínez	24	8	9	-3.695440	40.42787	Plaza de Alonso Martínez nº 5

En segundo lugar, en caso de que se desee emplear la segunda definición de  $d_i$  (4.2) se presenta el número total de salidas y de entradas de bicicletas en cada estación y hora en el mes de abril del año 2018. En la columna derecha se muestra  $\omega_i(t)$ , la demanda estimada de bicicletas por parte de los usuarios en cada hora y estación, que se ha calculado restando las entradas a las salidas y dividiendo la cantidad resultante entre el número de días del mes.

Tabla 4.3: Extracto del uso de estaciones por hora

hora	estacion	entradas	salidas	omega
7	57	17	124	4
8	41	43	149	4
8	74	71	180	4
8	95	221	28	-6
8	108	260	58	-7
8	129	59	178	4
9	64	210	76	-4
9	100	135	22	-4
18	64	258	363	4
20	64	176	300	4

Pese a que hemos calculado el parámetro  $\omega_i(t)$  asociado a la segunda definición de la demanda de bicicletas de las estaciones (4.2), vamos a decantarnos por la primera definición (4.1) por los motivos expuestos *anteriormente*.

Si evaluara el problema la Empresa Municipal de Transporte de Madrid podría contar con los datos de  $b_i$  en tiempo real. De hecho el número de bicicletas presente en cada estación es el dato que aporta dinamismo al problema. Si por algún motivo no conviene resolver el problema de forma dinámica contando con las demandas estimadas de bicicletas por hora y estación por parte de los usuarios del sistema,  $\omega_i(t)$  y unos datos iniciales del número de bicicletas en cada estación se podría estimar el número de bicicletas presentes en cada estación cada hora. Sería un planteamiento estático del problema.

En esta implementación los datos del número de bicicletas en cada estación serán generados aleatoriamente al no estar publicados en el Portal de Datos Abiertos del Ayuntamiento de Madrid. Estos datos, no obstante, sí están disponibles en la *web en tiempo real*, pero no están publicados en un formato estático puesto que carece de sentido al ser datos que cambian continuamente.

Conociendo la situación y características de las estaciones, las bicicletas disponibles en cada estación y la demanda de bicicletas de cada estación, queda conocer las distancias entre las estaciones. Como poseemos las coordenadas geométricas de cada estación, se podría calcular la distancia euclídea o geodésica entre cada par de estaciones, pero eso no se correspondería del todo con la realidad. Como los vehículos que reponen bicicletas se desplazan por las calles de una gran ciudad como Madrid, la distancia entre dos estaciones, la distancia recorrida para llegar de una a otra, y el tiempo transcurrido no se pueden relacionar de forma trivial. Consideramos que la magnitud de interés para asociarla a nuestra matriz de costes  $\mathcal{C}$  es el tiempo transcurrido conduciendo entre una estación y otra.

Para obtener la matriz  $\mathcal{C}$  de tiempos de desplazamiento entre una estación y otra hemos recurrido a la **API de BING MAPS**. Siguiendo las indicaciones de *Microsoft Bing* y de *este Blog* se introdujeron en la API BING MAPS las coordenadas de las estaciones y se obtuvo la matriz  $\mathcal{C}$  de tiempos de desplazamiento en coche (o camión) entre pares de estaciones de BiciMAD.

Un dato que no ha sido posible obtener es la localización de los depósitos de bicicletas. Se ha optado por incluir un depósito en la localización de la estación cuya distancia promedio del resto sea mínima.

Ambas formulaciones se han definido en ficheros `.mod` del lenguaje **AMPL**. Una vez comprobado que conjuntos de datos de prueba en ficheros `.dat` funcionan al ejecutar los respectivos ficheros `.run`, se procedió a incluir los datos del problema mediante la *API de R de AMPL*. Los conjuntos de datos se leyeron, modificaron y añadieron al problema empleando **código R en RMarkdown**. El código empleado puede consultarse en el anexo.

Debido al elevado tamaño del problema no se ha conseguido ejecutar el problema con las 170 estaciones de los datos. Para solucionar este problema, se han agrupado las estaciones siguiendo el **método de los k-medioides**, en concreto siguiendo el algoritmo **PAM** (de Particionamiento Alrededor de Medioides). De esta forma subdividimos el problema en k-problemas de menor tamaño en los que incluimos un depósito en la estación más representativa de cada uno. El algoritmo PAM consiste en elegir esos k puntos, los medioides. Se buscan las k estaciones para las que se alcance el valor objetivo de la siguiente función:

$$\sum_{i=1}^n \left\{ \min_{t=1\dots k} d(i, m_t) \right\} \quad (4.46)$$

Siendo  $i$  las estaciones del problema,  $m_t$  el medioide al que está asociada y  $d$  la distancia o disimilaridad de una estación al medioide, en este caso el tiempo de viaje entre ellos.

En primer lugar se hallan cada uno de los k medioides, denotados  $m_k$ :

$$\begin{aligned} m_1 &= \arg \min_{j=1\dots n} \sum_{i=1}^n d(i, j) \\ m_2 &= \arg \min_{j=1\dots n, j \neq m_1} \sum_{i=1}^n d(i, j) \\ &\dots \\ m_k &= \arg \min_{j=1\dots n, j \neq m_1, \dots, m_{k-1}} \sum_{i=1}^n d(i, j) \end{aligned} \quad (4.47)$$

Posteriormente, se le asocia a cada estación el medioide más cercano, obteniéndose una primera agrupación con su correspondiente valor de la función objetivo. Por último, mientras el coste de la configuración disminuya, se realiza la siguiente iteración:



- Para cada medioide  $m_t$  y cada estación  $i$  se intercambia el par de forma que  $i$  pasaría a ser medioide y  $m_t$  a no serlo.
- Se reagrupan las estaciones y se recalcula la función objetivo. Si disminuye, mantener el intercambio, si no, se deshace.

El particionamiento alrededor de mediodes se realizará mediante la función *pam* de la librería de R *cluster*. Los parámetros de entrada que debemos introducirle a la función son:

- *x*: La matriz de las posiciones de los datos o la matriz de distancias, en nuestro caso, la matriz de distancias.
- *diss*: Parámetro lógico que toma el valor **TRUE** si *x* es una matriz de distancia (de disimilaridad) y **FALSE** en caso contrario.
- *k*: El número de mediodes que queremos seleccionar y por ende el número de clusters.

La función *pam* devuelve varios parámetros, uno de ellos, la anchura media de la silueta, es una medida usada para validar el agrupamiento realizado. Este parámetro toma un valor entre 1 y -1. Cuanto más cercano se esté a 1, mejor es el agrupamiento. Podemos ver cómo varía este parámetro en función de los clusters que realicemos:

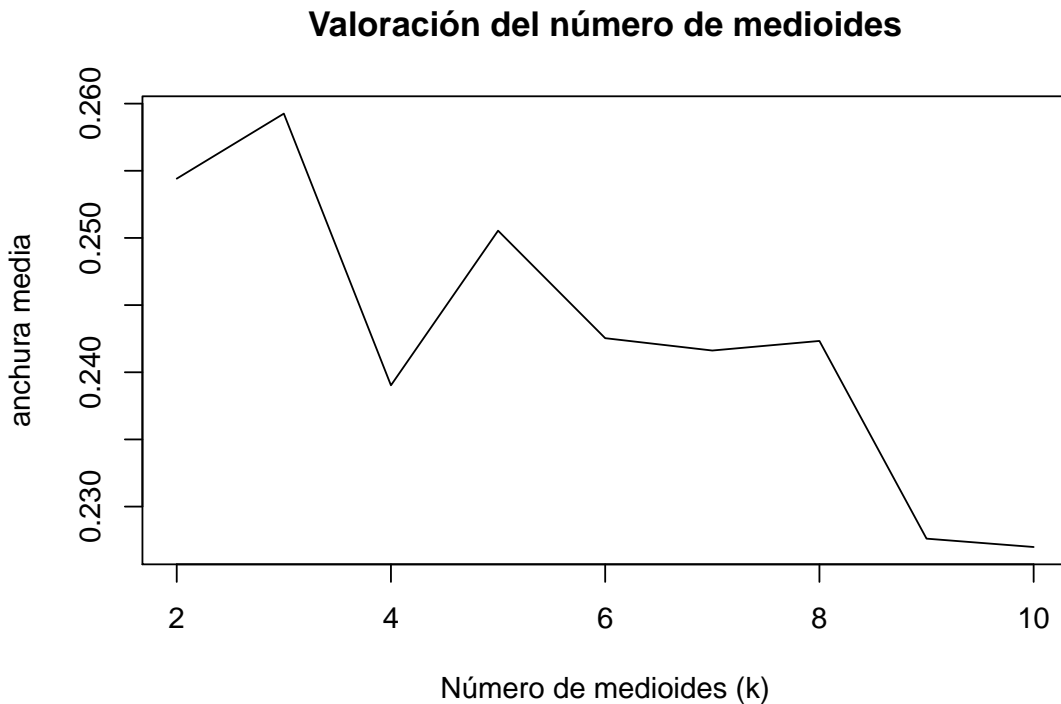


Figura 4.5: Anchura media según el número de clusters

Viendo la anterior gráfica concluiríamos que lo idóneo sería dividir el problema en 3 clusters. No obstante de esta manera los clusters siguen poseyendo un tamaño demasiado elevado para poder ejecutar el problema. Dividiremos entonces el problema en 5 clusters.

Aplicando el algoritmo de Partición Alrededor de Mediodes, obtenemos la siguiente división de las estaciones en clusters, con sus correspondientes medioides:

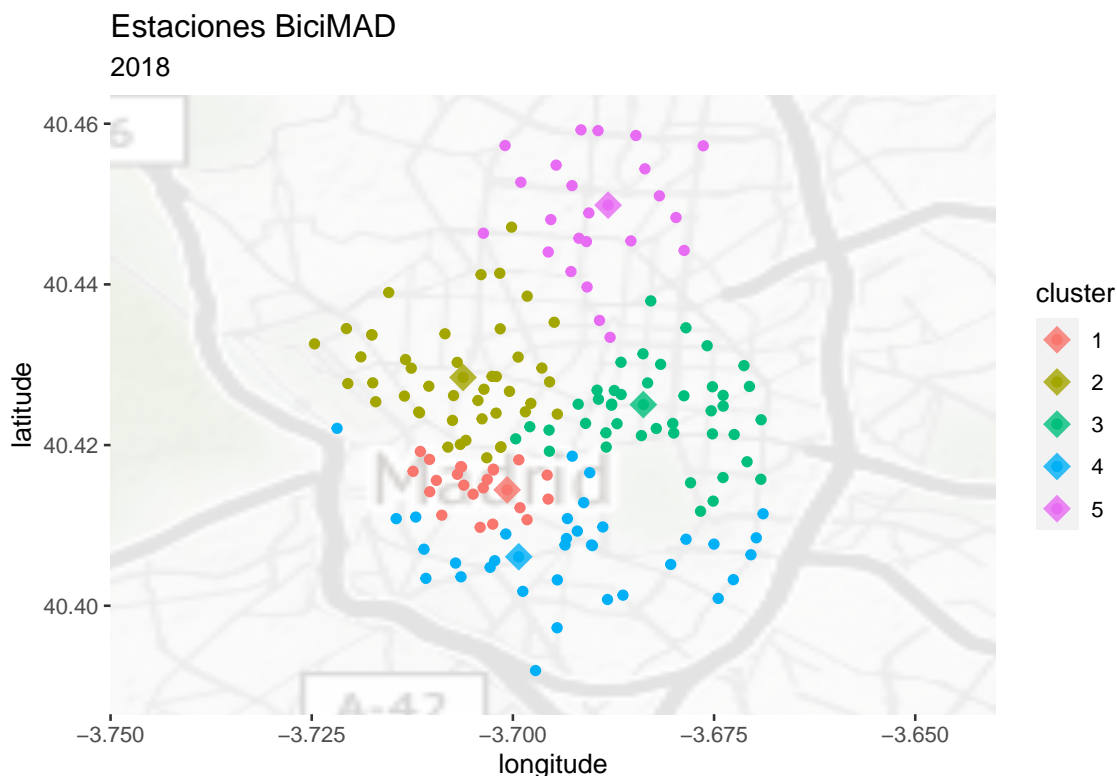


Figura 4.6: Estaciones BiciMAD

Vemos que la división del problema efectuada asocia algunas estaciones a medioides que no son el más cercano. Esto es debido a que el PAM es un método heurístico y de ahí que nuestra clasificación sea aceptable aunque no óptima.

Finalmente, ejecutamos el problema en cada uno de los clusters. Como se ha visto, son muchas las variantes del problema que pueden realizarse dependiendo de la capacidad del ordenador y de los criterios que se prioricen. Las condiciones de los problemas que se han implementado son las siguientes:

- Se efectúa la *Variante 6*, en la que se combinaban las dos formulaciones dadas. Las restricciones son las mismas que en la *Formulación 2* del problema. La función objetivo es la siguiente:

$$\text{mín} \quad \beta \cdot \left( \sum_{j \in \mathcal{V}'} y_j \right) + (1 - \beta) \cdot \alpha \cdot \left( \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} \sum_{k \in K} c_{ijk} \cdot x_{ijk} \right)$$

Donde  $\alpha$  es un parámetro que pone en una misma escala los valores de ambas funciones objetivo. Se ha calculado de la siguiente forma:

1. Calculamos el coste en el supuesto que desde el depósito se efectúe un recorrido de ida y vuelta para equilibrar las estaciones una a una (peor escenario para la formulación 1).
2. Calculamos la suma de los valores absolutos de las demandas insatisfechas (peor escenario para la formulación 2).
3. Dividimos la segunda cantidad entre la primera. Ese es nuestro valor  $\alpha$ . Finalmente, asociamos a  $\beta$  un valor entre 0 y 1 según la importancia de un criterio y otro. En nuestro caso, puesto que en las restricciones se limita el exceso de tiempo en la ruta de los vehículos,  $\beta = 0,9$ . De este modo el problema priorizará la satisfacción de la demanda y si esta se ve satisfecha buscará la ruta más corta.
  - Se considera un único vehículo por cada cluster que no debe superar una duración de ruta de 240 minutos. Se interpreta que los turnos de los trabajadores de BiciMAD son de 4 horas.
  - Se sitúa un depósito de 40 plazas y 20 bicicletas en la posición del medioide del cluster.

### 4.3.2. Descripción de la solución

Vamos a analizar la solución propuesta para el primer cluster, las demás se pueden consultar en el *apéndice*:

Tabla 4.4: Solución variable x, Primer Cluster

	-2	-1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
-2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
22	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla 4.5: Solución variable f, Primer Cluster

	-2	-1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
-2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0
8	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

El vector demanda de las estaciones del cluster 1 es el siguiente:

$$d_{C_1} = (0, -7, 0, 0, 0, 0, 9, -12, -10, 0, -8, 9, 0, 0, 7, 0, 0, 0, 0, 9, 9, 5, 0)$$

Se han asignado valores naturales ordenados a las estaciones del primer cluster, en la siguiente tabla presentamos en la columna de la derecha el identificador de cada estación del primer cluster y a la izquierda el número que le corresponde en las tablas anteriores:

Tabla 4.6: Identificación de las estaciones del primer cluster

numero	idEstacion
1	1
2	2
3	10
4	27
5	28
6	29
7	31
8	32
9	33

Tabla 4.6: Identificación de las estaciones del primer cluster (*continued*)

numero	idEstacion
10	34
11	35
12	36
13	37
14	38
15	39
16	40
17	43
18	44
19	45
20	46
21	47
22	48
23	56

Conociendo los trayectos que se realizan en la ruta y los tiempos de desplazamiento de una estación a otra podemos calcular el **tiempo total de la ruta: 89 minutos y 24 segundos**.

Mirando las tablas anteriores concluimos lo siguiente:

- El vehículo se desplaza al depósito auxiliar y vuelve. No se extrae información útil de este dato
- El vehículo transporta 13 bicicletas a la estación 12, que corresponde con la estación de id 36.
- El vehículo transporta 4 bicicletas de la estación 12 a la estación 11 (la estación de id 35)
- La estación 12 obtiene un balance de +9 bicicletas.
- El vehículo transporta 12 bicicletas de la estación 11 a la estación 6 (id 29).
- La estación 11 obtiene un balance de -8 bicicletas.
- El vehículo transporta 12 bicicletas de la estación 6 a la estación 15 (id 39).
- La estación 6 obtiene un balance de 0 bicicletas.
- El vehículo transporta 5 bicicletas de la estación 15 a la estación 22 (id 48).
- La estación 15 obtiene un balance de +7 bicicletas.
- El vehículo transporta 0 bicicletas de la estación 22 a la estación 2 (id 2).
- La estación 22 obtiene un balance de +5 bicicletas.

- El vehículo transporta 7 bicicletas de la estación 2 a la estación 1 (id 1).
- La estación 2 obtiene un balance de -7 bicicletas.
- El vehículo transporta 7 bicicletas de la estación 1 a la estación 9 (id 33).
- La estación 1 obtiene un balance de 0 bicicletas.
- El vehículo transporta 17 bicicletas de la estación 9 a la estación 8 (id 32).
- La estación 9 obtiene un balance de -10 bicicletas.
- El vehículo transporta 29 bicicletas de la estación 8 a la estación 7 (id 31).
- La estación 8 obtiene un balance de -12 bicicletas.
- El vehículo transporta 20 bicicletas de la estación 7 a la estación 21 (id 47).
- La estación 7 obtiene un balance de +9 bicicletas.
- El vehículo transporta 11 bicicletas de la estación 21 a la estación 20 (id 46).
- La estación 21 obtiene un balance de +9 bicicletas.
- El vehículo transporta 2 bicicletas de la estación 20 al depósito.
- El balance de la estación 20 es de +9 bicicletas.

Visualicemos la ruta descrita en la gráfica que se muestra a continuación, en la cual se representan las estaciones por su número de identificación, cada paso de la ruta con una flecha y el depósito con un rombo de color rojo situado donde la estación medioide (la 56).

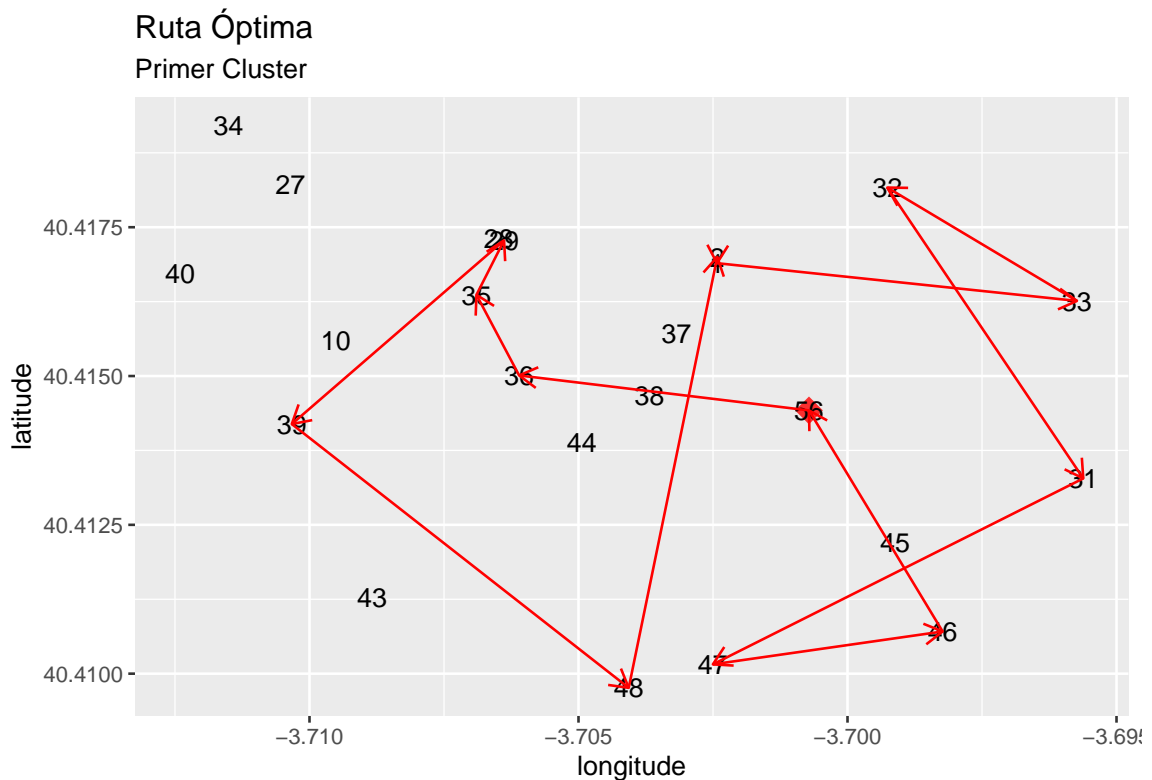


Figura 4.7: Ruta Optima del Primer Cluster

Percatémonos de que se ha satisfecho la demanda de todas las estaciones del primer cluster en el turno de 4 horas del empleado que conduce el vehículo.

Todas estaciones que no son atendidas por el vehículo poseían demanda nula, pero no todas las atendidas tienen demanda no nula. Analicemos esos casos:

La ruta pasa por la estación 1, que tiene demanda nula. La explicación es que, como puede verse, las estaciones 1 y 2 están situadas a una distancia tan pequeña que la API de Bing Maps no distingue entre una localización y otra. De hecho, ambas estaciones poseen la misma dirección: Puerta del Sol nº1. Se podría decir que ambas constituyen una misma estación que tiene dos servidores de acceso al servicio. Por tanto no debe preocuparnos el hecho de que el vehículo realice un viaje en balde de la estación 2 a la 1, realmente no se estaría desplazando.

La ruta también contiene a la estación 29, que también posee demanda nula. De acuerdo a nuestra solución, el vehículo se desplazaría de la estación 35 a la 29 empleando 0,3333 minutos y de la 29 a la 39 empleando 5,3833 minutos, resultando en 5,7166 minutos empleados en acudir de la estación 35 a la 39. El tiempo de viaje de la estación 35 a la 39 en nuestra matriz de costes es de 5,7167. Lo que sucede realmente es que la estación 29 está en la ruta entre la 35 y la 39 y pasar por la estación 29 no supone pérdida de tiempo alguna.

Observamos también en la gráfica que las rutas parecen dar rodeos. Teniendo en cuenta la localización de las estaciones, a simple vista la opción más corta es acudir de la estación 48 a la 47, en vez de ir a la estación dos. En el tradicional Problema del Vendedor Ambulante quizás esa sería la ruta óptima, pero no en nuestro problema, en el que entran en juego demandas negativas. Cuando el vehículo sale en nuestra solución de la estación 48 no carga con ninguna bicicleta, por lo que es razonable acudir a la estación 2, con exceso de bicicletas, que a la estación 47, con escasez. Si el vehículo se desplazara siempre a las estaciones más cercanas tendría que acudir a reponer bicicletas al depósito, algo que no ha sido necesario en la solución presentada.

Otros tramos de la ruta extraños en la visualización son las subrutas 1-33-32-31 y 31-47-46-56. Parece que el vehículo recorre más distancia que la necesaria. Debemos tener en cuenta que la matriz de costes no mide las distancias euclídeas sino el tiempo en el desplazamiento en coche por las calles de Madrid. Realizar el recorrido 1-32-33-31 (en vez de 1-33-32-31) supondría un aumento en el tiempo del recorrido de aproximadamente 6 minutos, y realizar el recorrido 31-46-47-56 (en vez del 31-47-46-56) supondría un aumento de algo menos de 2 minutos.

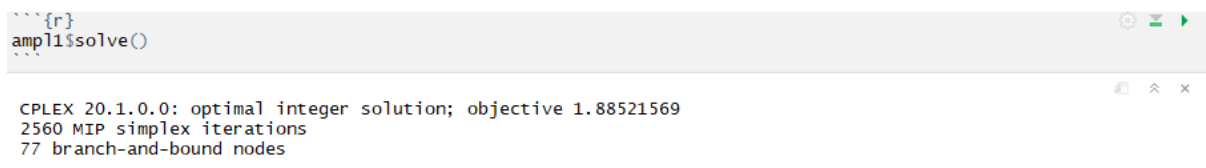


# Apéndice A

## Apéndice: Soluciones

### A.1. Resolución problema lineal

#### A.1.1. Cluster 1



```
{r}
ampl solve()

CPLEX 20.1.0.0: optimal integer solution; objective 1.88521569
2560 MIP simplex iterations
77 branch-and-bound nodes
```

Figura A.1: Ejecución Cluster 1

El tiempo total de duración de la ruta es de 89.3999 minutos: **1 hora 29 minutos y 24 segundos.**

Tabla A.1: Solución Variable x Primer Cluster

	-2	-1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
-2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
22	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla A.2: Solución Variable f Primer Cluster

	-2	-1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
-2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0
8	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla A.3: Identificación de las estaciones del primer cluster

numero	idEstacion
1	1
2	2
3	10
4	27
5	28
6	29
7	31
8	32
9	33
10	34
11	35
12	36
13	37
14	38
15	39
16	40

Tabla A.3: Identificación de las estaciones del primer cluster (*continued*)

numero	idEstacion
17	43
18	44
19	45
20	46
21	47
22	48
23	56





Tabla A.6: Identificación de las estaciones del segundo cluster (*continued*)

numero	idEstacion
18	26
19	30
20	58
21	59
22	60
23	61
24	63
25	116
26	117
27	118
28	119
29	120
30	121
31	122
32	123
33	124
34	125
35	130
36	131
37	150
38	156
39	157
40	160
41	161
42	163
43	164
44	168
45	169







Tabla A.9: Identificación de las estaciones del tercer cluster (*continued*)

numero	idEstacion
18	94
19	95
20	96
21	97
22	98
23	99
24	100
25	101
26	102
27	103
28	104
29	105
30	106
31	107
32	108
33	109
34	110
35	111
36	112
37	113
38	114
39	115
40	162
41	166
42	170
43	171





Tabla A.12: Identificación de las estaciones del cuarto cluster (*continued*)

numero	idEstacion
18	81
19	82
20	83
21	84
22	85
23	86
24	87
25	89
26	91
27	126
28	127
29	128
30	129
31	132
32	133
33	134
34	135
35	136

### A.1.5. Cluster 5

```

{r}
amp15$solve()

CPLEX 20.1.0.0: optimal integer solution; objective 1.64036754
1061 MIP simplex iterations
0 branch-and-bound nodes
    
```

Figura A.5: Ejecución Cluster 5

El tiempo total de duración de la ruta es de 57.7333 minutos: **57 minutos y 44 segundos**.

Tabla A.13: Solución Variable x Quinto Cluster

	-2	-1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
-2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla A.14: Solución Variable f Quinto Cluster

	-2	-1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
-2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla A.15: Identificación de las estaciones del quinto cluster

numero	idEstacion
1	137
2	138
3	139
4	140
5	141
6	142
7	143
8	144
9	145
10	146
11	147
12	148
13	149
14	151
15	152
16	153

Tabla A.15: Identificación de las estaciones del quinto cluster (*continued*)

numero	idEstacion
17	154
18	155
19	158
20	159
21	165
22	167
23	172
24	173



# Apéndice B

## Apéndice: Código

En este apéndice se recoge el código empleado en la obtención de las soluciones

### B.1. Código AMPL

```
reset; #para poner a 0 variables que estaban asignadas

# =====
# PARAMETROS DEL PROBLEMA
# =====

# Definimos las estaciones
# Vamos a denotar los depositos como nodos negativos y las estaciones como positivos
# n numero estaciones
param n >=0 integer;

# n_0 el numero de depositos, el *2 es por los depositos auxiliares
param n_0 >=0 integer;
set V := {-n_0*2..-1} union {1..n};

# Matriz de distancias
param C {V, V} >= 0;

# No va asi.
# param C2{EST2, EST2} >= 0;

# Los vehiculos
param m > 0 integer;
set K := 1..m;

# Plazas y bicicletas.
param P {V} >= 0 integer;
param b {V} < 200 >= 0 integer;

# Demanda. Igual
param d {V} integer;

# conjunto estaciones, depositos y de estaciones con demanda no nula
set Vest := {i in V: i >0};
set Vdep := {i in V: i<0};
set Vdem := {i in V: d[i]<>0};
```

Figura B.1: Código común a ambas formulaciones

```

# los depositos sin los auxiliares
set VdepT := -n_0*2+1..-1 by 2;

# Conjunto con el deposito de cada vehiculo
set H2 within {K, VdepT};

# capacidad vehiculos
param Q {K} >0 integer;

# Limite de tiempo
param T >0;

# =====
# VARIABLES
# =====

var x {V,V,K} binary ;
var f {V,V,K} integer;
var u {Vest,K} >=1 <=n integer;
var y {Vest} >=0 integer;

```

Figura B.2: Código común a ambas formulaciones

```

# =====
# F.OBJ
# =====

minimize fobj1:
    sum{i in V, j in V, k in K} C[i,j]*x[i,j,k];

# =====
# RESTRICCIONES
# =====

subject to
    #atendidas 1 vez maximo
    r1{j in Vest}: sum{i in V, k in K} x[i,j,k]<=1;
    r2{j in Vest}: sum{i in V, k in K} x[j,i,k]<=1;
    # flujo dep entrante=saliente
    r3{k in K, i in V}: sum{j in V} x[i,j,k]=sum{j in V} x[j,i,k];
    # rutas con el comienzo
    r4{(k,i) in H2}: sum{j in V} x[i,j,k] >= 1;
    # no pasa por otros depositos
    r5{(k,i) in {K,VdepT}: (k,i) not in H2}: sum{j in V} x[i,j,k]= 0;
    r6{(k,i) in {K,VdepT}: (k,i) not in H2}: sum{j in V} x[j,i,k]= 0;
    # MTZ
    r7{k in K, i in Vest, j in Vest: i <> j}: u[i,k]-u[j,k]+ n*x[i,j,k] <= n-1;
    # Se satisface la demanda
    r8{j in Vest}: sum{i in V, k in K} f[i,j,k] - sum{i in V, k in K} f[j,i,k]=d[j];
    # acotando f para cumplir demanda
    r10_1{i in V, j in V, k in K}: 0 <= f[i,j,k];
    r10_2{i in V, j in V, k in K}: -d[i]*x[i,j,k] <= f[i,j,k];
    r10_3{i in V, j in V, k in K}: d[j]*x[i,j,k] <= f[i,j,k];
    r10_4{i in V, j in V, k in K}: f[i,j,k] <= Q[k]*x[i,j,k];
    r10_5{i in V, j in V, k in K}: f[i,j,k] <= (Q[k]-d[i])*x[i,j,k];
    r10_6{i in V, j in V, k in K}: f[i,j,k] <= (Q[k]+d[j])*x[i,j,k];
    # acotando f para el deposito
    r11_1{j in VdepT}: sum{i in V, k in K} f[i,j,k] - sum{i in V, k in K} f[j,i,k] <= (P[j]-b[j]);
    r11_2{j in VdepT}: -b[j] <= sum{i in V, k in K} f[i,j,k] - sum{i in V, k in K} f[j,i,k];

```

Figura B.3: Código particular de la primera formulación

```

# =====
# F.OBJ
# =====

minimize fobj2:
    sum{j in Vest} y[j];

# =====
# RESTRICCIONES
# =====

subject to
    #atendidas 1 vez maximo
    r1{j in Vest}: sum{i in V,k in K} x[i,j,k]<=1;
    r2{j in Vest}: sum{i in V,k in K} x[j,i,k]<=1;
    r14{i in V,k in K}: x[i,i,k]=0;
    # flujo dep entrante=saliente
    r3{k in K, i in V}: sum{j in V} x[i,j,k]=sum{j in V} x[j,i,k];
    # rutas con el comienzo
    r4{(k,i) in H2}: sum{j in V} x[i,j,k] >= 1;
    # no pasa por otros depositos
    r5{(k,i) in {K,VdepT}: (k,i) not in H2}: sum{j in V} x[i,j,k]= 0;
    r6{(k,i) in {K,VdepT}: (k,i) not in H2}: sum{j in V} x[j,i,k]= 0;
    # MTZ
    r7{k in K, i in Vest, j in Vest: i <> j}: u[i,k]-u[j,k]+ n*x[i,j,k] <= n-1;
    # Se satisface la demanda
    r8{k in K}: sum{i in V, j in V} C[i,j]*x[i,j,k] <= T;
    # Restricciones a f
    r10_1{i in V, j in V, k in K}: 0 <= f[i,j,k];
    r10_4{i in V, j in V, k in K}: f[i,j,k] <= Q[k]*x[i,j,k];
    r10_5{j in V}: sum{i in V, k in K} f[i,j,k] - sum{i in V, k in K} f[j,i,k] <= (P[j]-b[j]);
    r10_6{j in V}: -b[j] <= sum{i in V, k in K} f[i,j,k] - sum{i in V, k in K} f[j,i,k];
    # Acotando y
    r12{j in Vest}: sum{i in V, k in K} f[i,j,k] - sum{i in V, k in K} f[j,i,k]- d[j] <= y[j];
    r13{j in Vest}: -sum{i in V, k in K} f[i,j,k] + sum{i in V, k in K} f[j,i,k]+ d[j] <= y[j];

```

Figura B.4: Código particular de la segunda formulación

```

# =====
# F.OBJ
# =====

param beta >0;
param alfa >0;

minimize fobj2:
    beta*(sum{j in Vest} y[j]) + (1-beta)*alfa*(sum{i in V,j in V, k in K} C[i,j]*x[i,j,k]);

```

Figura B.5: Código particular de la variante 6

## B.2. Código R

```
# Librerías utilizadas

library(jsonlite)
library(RJSONIO)
library(httr)
library(tibble)
library(readxl)
library(sets)

library(tidyverse)
library(lubridate)
library(dplyr)
library(cluster)
library(tidyjson)
library(magrittr)
library(tidyr)
library(osmdata)
library(sf)
library(ggmap)

# Instalación de la librería rAMPL

library(devtools)
#install.packages("Rcpp", type="source")
#install.packages("https://ampl.com/dl/API/rAMPL.tar.gz", repos=NULL)
library(rAMPL)
setwd(paste(find.package("rAMPL"), "examples", sep="/"))
```

### B.2.1. Preparación de los datos, obtención de información del uso de BiciMAD

En primer lugar se mostrará el código necesario para el uso de la API de BING MAPS y la obtención de la matriz de duraciones de trayectos al volante entre estaciones.

```
# Aviso: Find a driving route that minimizes the
# distance and specifies that the
# first turn must be at least 500 meters
#from the starting point.

# install.packages("remotes")
remotes::install_github("gsk3/taRifx.geo")
```

```

remotes::install_github("kavetinaveen/CWVRP")

library(geosphere)
library(stringr)
library("tidyverse")
library("readxl")

library(RCurl)
library(XML)
library(RJSONIO)

library(taRifx.geo)

estaciones_lon_lat <- read_excel("2018_Julio_Bases_Bicimad_EMT.xlsx",
                                range = "Hoja1!E1:F173",
                                col_names = TRUE)
# Se eliminan las estaciones 162 y 163 en el orden de
#lectura, ya que no tienen latitud
estaciones_lon_lat =estaciones_lon_lat[-c(162,163),]
# Se leen todos los datos de las estaciones,
#incluidas sus identificaciones
estaciones<- read_excel("2018_Julio_Bases_Bicimad_EMT.xlsx",
                        range = "Hoja1!A1:G173",
                        col_names = TRUE)
estaciones =estaciones[-c(162,163),]
identificador_estacion=as.integer(estaciones$id)
estaciones_ordenadas=estaciones[order(identificador_estacion),]
# Se usarán las estaciones ordenadas por su identificador,
#y en ese orden se obtendrá la matriz de distancias
estaciones_lon_lat=estaciones_lon_lat[order(identificador_estacion),]

#cambio de orden de longitus y latitud
aux_c=names(estaciones_lon_lat)
auxiliar=estaciones_lon_lat[,1]
estaciones_lon_lat[,1]=estaciones_lon_lat[,2]
estaciones_lon_lat[,2]=auxiliar
names(estaciones_lon_lat)[1]=aux_c[2]
names(estaciones_lon_lat)[2]=aux_c[1]

rowct <- nrow(estaciones_lon_lat)
from_mat=as.matrix(estaciones_lon_lat)
from_mat_in <- apply(from_mat, 1, paste0, collapse=",")
to_mat=as.matrix(estaciones_lon_lat)
to_mat_in <- apply(to_mat, 1, paste0, collapse=",")

```

```

distancias=matrix(c(rep(NA,rowct*rowct)),rowct,rowct)
duracion=matrix(c(rep(NA,rowct*rowct)),rowct,rowct)
distunit <- 'km'

n_estaciones=rowct
# n_estaciones=5
for(i in 1:n_estaciones){
  for(j in 1:n_estaciones){
    fromtourl=""
    bingkey <- 'introducir_clave_bing'
    starturl <- 'https://dev.virtualearth.net/
                REST/v1/Routes/DistanceMatrix?key='
    fromtourl <- str_c(starturl,bingkey,
                      '&travelMode=Driving&startTime=2021-05-04T08:00:00&
                      timeUnit=minute&origins=',
                      from_mat_in[i], '&destinations=',
                      to_mat_in[j], '&distanceUnit=', distunit)
    if (!identical(from_mat_in[i], to_mat_in[j])){
      u1 <- RCurl::getURL(fromtourl)
      u2 <- RJSONIO::fromJSON(u1, simplify=FALSE)
      distancias[i,j]= u2$resourceSets[[1]]$resources[[1]]$
        results[[1]]$travelDistance
      duracion[i,j]= u2$resourceSets[[1]]$resources[[1]]$
        results[[1]]$travelDuration
    }
  }
}

# Grabar resultados en RData -----
save(distancias,tiempo,file="Distancias_Estaciones_Bicimad.RData")
# Grabar resultados en excel -----
result_distancias=as.matrix(distancias)
result_duracion=as.matrix(duracion)
excel_salida = "distancias_estaciones.xlsx"
library("openxlsx")
if (file.exists(excel_salida)) {
  file.remove(excel_salida)
}
wb = openxlsx::createWorkbook()
openxlsx::addWorksheet(wb,sheetName = "distancias")
openxlsx::writeData(wb, x = result_distancias,
                   sheet = "distancias",colNames = FALSE)
openxlsx::addWorksheet(wb,sheetName = "duracion")
openxlsx::writeData(wb, x = result_duracion,
                   sheet = "duracion",colNames = FALSE)

```

```
openxlsx::saveWorkbook(wb,file = excel_salida,overwrite = TRUE );
rm(wb)
```

A continuación se mostrará el código empleado para analizar el uso por parte de los usuarios de las estaciones de BiciMAD

```
# Leemos el documento json donde esta la informacion
# de uso de BiciMAD:
# Cada uno de los viajes efectuados, con la estacion
# de entrada y salida, el tiempo de viaje etc

raw <- readLines("data/201804UsageBicimad.json")
dat <- iconv(raw, "latin1", "utf8")
dat <- sapply(dat, RJSONIO::fromJSON)
datos=dat %>% spread_all

estaciones=read_csv2("data/bases_bicimad.csv",
                    col_types = cols(
                      .default = col_number(),
                      Distrito= col_factor(),
                      Barrio= col_factor(),
                      Calle= col_factor(),
                      `Nº Finca`=col_character(),
                      `Tipo de Reserva`=col_skip(),
                      Latitud=col_character(),
                      Longitud=col_character(),
                      Direccion=col_character()
                    ))

datos2 = datos %>%
  select(document.id,`_id`,
         idplug_base,user_type,travel_time,
         idunplug_base,idunplug_station,
         idplug_station,unplug_hourTime)

ncol(datos2)
datos2_2=datos2[1:50,-c(10)]

datos3=datos2 %>%
  separate(unplug_hourTime,
          into=c("fecha_hora2","delete"),sep = ".000") %>%
  separate(fecha_hora2,into = c("fecha","hora"),sep = "T")

datos3 %<>% mutate(fecha_hora=ymd_hms(paste(fecha,hora)),
```

```

        fecha=ymd(fecha),
        hora=hms(hora))

datos_unplug= datos3 %>% select(idunplug_station,
                              idunplug_base,
                              document.id, `_id`,
                              fecha,hora,
                              fecha_hora,user_type) %>%
  inner_join(estaciones,c("idunplug_station"="Número"))

```

```

# Obtención de algunas graficas descriptivas
# Que se muestran en la sección 4.1

```

```

datos_unplug %>%
  group_by(hora,Barrío,Distrito) %>%
  count() %>%
  mutate(hora=hour(hora)) %>%
  ggplot(aes(hora,n,
             colour=fct_reorder2(Barrío,hora,n)))+
  geom_line()+
  geom_point()+
  labs(colour = "Barrío")+
  facet_wrap(~Distrito,nrow = 2)+
  theme(legend.position = "none")

```

```

datos_unplug %>% filter(idunplug_station==80)%>%
  group_by(idunplug_station, hora) %>%
  count() %>%
  mutate(hora=hour(hora))%>%
  ggplot(aes(hora,n))+
  geom_point()+
  geom_line()+
  ggtitle("Desenganches de BiciMAD por hora",
          subtitle = "Estación de Atocha")

```

```

# Obtención de la tabla 6
# Que proporciona una estimación de la demanda por
# horas de cada estacion

```

```

# Así eliminamos los movimientos de los empleados
# de los datos

```

```

datos3 %<>% filter(user_type != 3)

```

```

# Hallamos los datos de desenganche
#de cada estacion

```

```

datos_salida= datos3 %>%

```



```

count(hora,idunplug_station) %>%
complete(hora,idunplug_station) %>%
select(hora,estacion=idunplug_station, salidas=n)

datos_salida %<>%
mutate(hora=hour(hora),
        salidas=replace_na(salidas,0)) %>%
arrange(hora,estacion)

# Hallamos los datos de enganche
# de cada estacion, es algo más
#

datos_entrada= datos3 %>%
  select(-c(idplug_base,user_type,
            idunplug_base,idunplug_station,
            delete,fecha,fecha_hora))
datos_entrada %<>%
mutate(hora_entrada=hour(as_datetime(
  as.duration(hora)+dseconds(travel_time))))

datos_entrada %<>%
count(hora_entrada,idplug_station) %>%
complete(hora_entrada,idplug_station) %>%
select(hora=hora_entrada,
        estacion=idplug_station, entradas=n)

datos_entrada %<>%
mutate(entradas=replace_na(entradas,0)) %>%
arrange(hora,estacion)

t_demanda1=datos_entrada %>%
full_join(datos_salida)

t_demanda1 %<>%
mutate(omega= round((salidas-entradas)/30))

write_csv(t_demanda1,"data/t_demanda1.csv")

```

## B.2.2. Clustering y resolución del problema lineal

En primer lugar leemos los datos de estaciones y distancias entre ellas y dividimos el

problema en clusters.

```
#Lectura de la información sobre las estaciones

estaciones<- read_excel("2017_Julio_Bases_Bicimad_EMT.xlsx",
                        range = "Hoja1!A1:G173",col_names = TRUE)
estaciones =estaciones[-c(162,163),]

# Ordenación de la tabla y obtención de
#los identificadores de las estaciones
identificador_estacion=as.integer(estaciones$id)
estaciones_ord=estaciones[order(identificador_estacion),]
id_estaciones=estaciones_ord$id

# Lectura de la matriz de disimilaridad

dist_BiciMAD_C= read_excel("distancias_estaciones.xlsx",
                           sheet = "duracion",
                           col_names = as.character(id_estaciones))
class(dist_BiciMAD_C)="data.frame"
rownames(dist_BiciMAD_C)=colnames(dist_BiciMAD_C)
for (i in 1:ncol(dist_BiciMAD_C)) {
  dist_BiciMAD_C[,i]=replace_na(dist_BiciMAD_C[,i],0)
}

# Clusterización, análisis gráfico.

plot(2:10,
     sapply(2:10,
           function(x) pam(dist_BiciMAD_C,diss = TRUE,k=x)$silinfo$avg.width ),
     type="l",
     xlab="Número de medioides (k)",
     ylab="anchura media")

clusterEst=pam(dist_BiciMAD_C,diss = TRUE,k=5)

# Tabla que relaciona las estaciones con sus clusters

cluster5=as.data.frame(clusterEst$clustering)
cluster5 %<>%
  mutate(estacion=rownames(cluster5),
         cluster=`clusterEst$clustering`)
cluster5= cbind.data.frame(estacion=cluster5$estacion,
                          cluster=cluster5$cluster)
rownames(cluster5)=NULL

# En este chunk se prepararán las matrices de distancia para
# ser incorporadas al objeto ampl en el formato adecuado
```

```

dist_BiciMAD_C=cbind.data.frame(desde=colnames(dist_BiciMAD_C),
                                dist_BiciMAD_C)
dist_BiciMAD_Clong = dist_BiciMAD_C %>%
  pivot_longer(-desde,names_to="hacia",values_to="value")

dist_BiciMAD_Clong %<>%
  inner_join(cluster5,c("desde"="estacion")) %>%
  select(desde,hacia,value,clusDesde=cluster) %>%
  inner_join(cluster5,c("hacia"="estacion")) %>%
  select(desde,hacia,value,clusDesde,clusHacia=cluster)

# Separamos en clusters las estaciones

clus1=dist_BiciMAD_Clong %>%
  filter(clusDesde==1 & clusHacia==1) %>%
  select(desde,hacia,value)
clus2=dist_BiciMAD_Clong %>%
  filter(clusDesde==2 & clusHacia==2) %>%
  select(desde,hacia,value)
clus3=dist_BiciMAD_Clong %>%
  filter(clusDesde==3 & clusHacia==3) %>%
  select(desde,hacia,value)
clus4=dist_BiciMAD_Clong %>%
  filter(clusDesde==4 & clusHacia==4) %>%
  select(desde,hacia,value)
clus5=dist_BiciMAD_Clong %>%
  filter(clusDesde==5 & clusHacia==5) %>%
  select(desde,hacia,value)

# C1

C1=as.data.frame(pivot_wider(clus1,names_from = hacia,
                             values_from = value))
n1=nrow(C1)
rownames(C1)=C1$desde
C1=C1[,-1]
# El deposito está de las estaciones a la misma
#distancia que el medioide
m1=which(rownames(C1)==clusterEst$medoids[1])

C1=cbind.data.frame(`-2`=rep(240,n1),`-1`=C1[,m1],C1)
C1=rbind.data.frame(`-2`=c(0,0,rep(240,n1)),`-1`=t(C1)[,m1],C1)
C1[2,1]=0

# C2, C3, C4 y C5 son análogos a C1

```

```
#Lectura de las plazas y generacion aleatoria de las bicicletas
```

```
set.seed(123)
```

```
p_i=c(estaciones_ord$total_bases)
b_i=c(rep(NA,length(id_estaciones)))
for (i in 1:length(p_i)) {
  b_i[i]=sample(p_i[i]+1,1)-1
}
```

```
# Fucion que obtiene la demanda de una estación a partir
#de las plazas totales y las bicicletas
```

```
ob_demanda= function(alpha=0.25,p,b){
  if (ceiling(alpha*p)>b| floor((1-alpha)*p)<b){
    d=round(0.5*p)-b
    return(d)}
  else {return(0)}}
```

```
d_i=c(rep(NA,length(id_estaciones)))
for (i in 1:length(p_i)) {
  d_i[i]=ob_demanda(0.25,p_i[i],b_i[i])
}
```

A continuación expondremos el código empleado para resolver el problema lineal asociado al primer cluster.

Para los demás clusters el código es análogo, únicamente se deben cambiar los nombres de los objetos.

```
# Creacion problema ampl asociado al primer cluster
# Lectura de los ficheros ampl
```

```
env1 <- new(Environment, "C:\\Datos\\ampl")
ampl1 <- new(AMPL, env1)
ampl1$setOption("solver", "cplex")
```

```
ampl1$read("C:/Datos/5curso/TFG/ampl_carmarcar5/fobj4.mod")
ampl1$readData("C:/Datos/5curso/TFG/ampl_carmarcar5/fobj4.dat")
```

```
# Creación y modificación de variables necesarias
```

```
n_0=1
p_i_c1=c(40,40,p_i[clusterEst$clustering==1])
b_i_c1=c(0,20,b_i[clusterEst$clustering==1])
d_i_c1=c(0,0,d_i[clusterEst$clustering==1])
```

```

C_1=C1
colnames(C_1)=c(-2,-1,seq(1,n1,by=1))
rownames(C_1)=colnames(C_1)
C_1=cbind(desde=rownames(C_1),C_1)
C_1 %<>% pivot_longer(-desde,names_to="hacia",values_to="value")
C_1 %<>% mutate(desde=as.integer(desde),hacia=as.integer(hacia))

maxdurC1= sum(C1$`-1`)+sum(C1[2,])
maxdurC1
maxdemC1=sum(sapply(d_i_c1,abs))
maxdemC1

coefobj1=maxdemC1/maxdurC1
coefobj1

```

*# Incluimos en el modelo los nuevos datos*

```

n_ampl1=ampl1$getParameter("n")
n_ampl1$setValues(n1)
n_ampl1$getValues()
n_0_ampl1=ampl1$getParameter("n_0")
n_0_ampl1$setValues(1)
C_ampl1=ampl1$getParameter("C")
C_ampl1$setValues(C_1)
C_ampl1$getValues()

m_ampl1=ampl1$getParameter("m")
m_ampl1$setValues(1)
H_ampl1=ampl1$getSet("H2")
H_ampl1$getValues()

p_ampl1=ampl1$getParameter("P")
p_ampl1$setValues(p_i_c1)
p_ampl1$getValues()
b_ampl1=ampl1$getParameter("b")
b_ampl1$setValues(b_i_c1)
d_ampl1=ampl1$getParameter("d")
d_ampl1$setValues(d_i_c1)
d_ampl1$getValues()

Q_ampl1=ampl1$getParameter("Q")
Q_ampl1$setValues(c(30))

```

```

T_ampl1=ampl1$getParameter("T")
T_ampl1$setValues(240)

alpha_ampl1=ampl1$getParameter("alfa")
alpha_ampl1$setValues(coefobj1)

# Ejecucion

ampl1$solve()

# Obtencion variables "x" y "f"

varx1=ampl1$getVariable("x")
tablax1=varx1$getValues()
tablax1=as_tibble(tablax1)
colnames(tablax1)=c("i","j","k","x")
#tablax %>% filter(k==1)
xc1=tablax1[tablax1$k==1,]
xc1 =xc1[,-3]
xc1$j
xc1 %<>% pivot_wider(names_from=2,values_from=3)
xc1

varf1=ampl1$getVariable("f")
tablaf1=varf1$getValues()
tablaf1=as_tibble(tablaf1)
colnames(tablaf1)=c("i","j","k","f")
#tablax %>% filter(k==1)
fc1=tablaf1[tablaf1$k==1,]
fc1 =fc1[,-3]
fc1 %<>% pivot_wider(names_from=2,values_from=3)
fc1

```

Veamos cómo se ha obtenido la duración total de la ruta:

```

solxc1=as.data.frame(xc1)
rownames(solxc1)=as.character(solxc1$i)
solxc1=solxc1[,-1]

(tiempo1=sum(C1*solxc1))

```

Por último exponemos el código empleado para representar las estaciones de BiciMAD en el mapa de Madrid, clasificándolas por colores según el cluster al que pertenezcan y el empleado para representar la ruta óptima del cluster 1.

```

estaciones_cluster= estaciones_ord %>%
  mutate(id=as.character(id)) %>%
  inner_join(cluster5,c("id"="estacion"))

estaciones_cluster %<>%

```

```

mutate(cluster=as.factor(cluster))

mad_map <- get_map(getbb("Madrid"))
ggmap(mad_map,
      extent = "none",
      darken = c(0.8,"white"),
      base_layer = ggplot(aes(x=longitudo,
                              y=latitudo,
                              color= cluster),
                          data = estaciones_cluster))+
geom_point()+
geom_point(data =filter(estaciones_cluster,
                        id==56|id==13|id==95|id==53|id==146),
           shape=18,size = 5, alpha=0.8)+
coord_cartesian(ylim=c(40.39,40.46),
                xlim=c(-3.745,-3.645))+
labs(title = "Estaciones BiciMAD", subtitle = "2018")

orden=1:14
estaciones=as.character(c(56,36,35,29,39,48,2,1,33,32,31,47,46,56))

ruta1=cbind.data.frame(orden,id=estaciones)
estaciones_cluster1=estaciones_cluster %>% filter(cluster==1)
ruta1com=ruta1 %>% left_join(estaciones_cluster1) %>% arrange(orden)

estaciones_cluster1 %>%
  ggplot(aes(x=longitudo, y=latitudo))+
  #geom_point()+
  geom_point(data = filter(estaciones_cluster,id==56),
            shape=18, size = 5, alpha=0.8,color="red")+
  annotate("text",x=estaciones_cluster1$longitudo, y=estaciones_cluster1$latitudo,lab
  geom_segment(color="red",
              aes(x=c(ruta1com$longitudo[-14],rep(NA,10)),
                  y=c(ruta1com$latitudo[-14],rep(NA,10)),
                  xend=c(ruta1com$longitudo[-1],rep(NA,10)),
                  yend=c(ruta1com$latitudo[-1],rep(NA,10))),
              arrow=arrow(length=unit(0.3,"cm")))+
  labs(title = "Ruta Óptima", subtitle = "Primer Cluster")

```





# Bibliografía

- [1] Brinkmann, J., Ulmer, M.W. and Mattfeld, D.C. 2015. Short-term strategies for stochastic inventory routing in bike sharing systems. *Transportation Research Procedia*. 10, (2015), 364–373.
- [2] Clarke, G. and Wright, J.W. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*. 12, 4 (1964), 568–581.
- [3] Cordeau, J.F., Laporte, G. and Salvendy, M. 2007. *Transportation on demand, handbooks in operations research and management science*. Elsevier.
- [4] Dantzig, G.B. and Ramser, J.H. 1959. The truck dispatching problem. *Management Science*. 6, 1 (1959), 80–91.
- [5] Dell’Amico, M., Hadjiconstantinou, E., Iori, M. and Novellani, S. 2013. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega*. 45, (Jan. 2013).
- [6] Dell’Amico, M., Hadjiconstantinou, E., Iori, M. and Novellani, S. 2014. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega*. 45, (2014), 7–19.
- [7] Eilon, Watson-Gandy and Christofides 1971. *Distribution management: Mathematical modelling and practical analysis*. Hafner Publication Co.
- [8] Flood, M.M. 1956. The traveling-salesman problem. *Operations Research*. 4, 1 (1956), 61–75.
- [9] Hemert, J. van and la, J. 2004. Dynamic routing problems with fruitful regions: Models and evolutionary computation. (Sep. 2004).
- [10] Larsen, A. 2001. *The dynamic vehicle routing problem*. Technical University of Denmark (DTU).
- [11] Larsen, A., Madsen, O. and Solomon, M. 2007. Classification of dynamic vehicle routing systems. *Operations Research/Computer Science Interfaces Series*. 38, (2007), 19–40.
- [12] Larsen, A., Madsen, O. and Solomon, M. 2002. Partially dynamic vehicle routing-models and algorithms. *The Journal of the Operational Research Society*. 53, 6 (2002), 637–646.
- [13] Lund, K., Madsen, O. and Rygaard, J. 1996. *Vehicle routing problems with varying degrees of dynamism. technical report*. IMM Institute of Mathematical Modelling.
- [14] Luque-Calvo, P.L. 2019. *Cómo crear tablas de información en r markdown*. Disponible en <http://destio.us.es/calvo>.

- [15] Luque-Calvo, P.L. 2017. *Escribir un trabajo fin de estudios con r markdown*. Disponible en <http://destio.us.es/calvo>.
- [16] Powell, W.B., Jaillet, P. and Odoni, A. 1995. Network routing. *Handbooks in Operations Research and Management Science*. 8, (1995), 141–295.
- [17] Psaraftis, H. 1980. A dynamic-programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*. 14, 2 (1980), 130–154.
- [18] R Core Team 2016. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing.
- [19] Ramos, T.R.P., Gomes, M.I. and Póvoa, A.P.B. 2020. Multi-depot vehicle routing problem: A comparative study of alternative formulations. *International Journal of Logistics Research and Applications*. 23, 2 (2020), 103–120.
- [20] RStudio Team 2015. *RStudio: Integrated development environment for r*. RStudio, Inc.
- [21] Tagmouti, M., Gendreau, M. and Potvin, J.-T. 2011. A dynamic capacitated arc routing problem with time-dependent service costs. *Transportation Research Part C: Emerging Technologies*. 19, 1 (2011), 20–28.
- [22] Taniguchi, E., Thompson, R.G., Yamada, T. and Van Duin, R. 2001. *City logistics: Network modelling and intelligent transport systems*. Pergamon.
- [23] Yang, J., Jaillet, P. and Mahmassani, H. 2004. Real-time multivehicle truckload pickup and delivery problems. *Transportation Science*. 38, (May 2004), 135–148.
- [24] Zang, Zhenzhen, Liu, M. and Lim, A. 2015. A memetic algorithm for the patient transportation problem. *Omega*. 54, (2015), 60–71.
- [25] Catálogo de datos abiertos, ayuntamiento de madrid. <https://datos.madrid.es/portal/site/egob>.
- [26] European cyclist’ federation. <https://ecf.com/what-we-do/urban-mobility/bike-share-schemes-bss>.
- [27] Observatorio de la bicicleta pública en españa. <https://bicicletapublica.es/>.