UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

ESTUDIO DE LA APLICACIÓN DE ALGORITMOS DE ENRUTADO AL BALANCEO DE VEHÍCULOS EN SISTEMAS DE COMPARTICIÓN DE BICICLETAS

AUTOR: IGNACIO CEBRIÁN MARTÍNEZ

TUTOR: VÍCTOR SÁNCHEZ ANGUIX

COTUTOR: JUAN MIGUEL ALBEROLA OLTRA

Curso Académico: 2020-21

Índice

1. Normativa.

- Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos personales y garantía de los derechos digitales.

Esta ley es la adaptación de la legislación española a la normativa europea. Hace referencia a las exigencias en el tratamiento de información entre el usuario y las empresas. Esta ley tiene como finalidad el proteger la intimidad, privacidad e integridad del individuo y del mismo modo, regula todo proceso en el que se de lugar una transferencia de datos, buscando garantizar la seguridad de esta.

Se considera relevante para nuestro caso debido al trabajo que se realiza con los datos de uso de las diferentes estaciones del servicio de compartición de bicicletas de Valenbisi. Los datos provienen de la empresa JCDecaux [1] y en todo momento se protege la información de sus usuarios, ya que no se ofrece ningún dato identificativo ni acerca de los movimientos realizados.

[1] https://www.jcdecaux.es/

2. Introducción.

Los problemas de enrutado de vehículos se encuentran presentes en el día a día del mundo de la industria debido a la gran importancia que está cogiendo en este tiempo el ámbito de la logística. Es de gran interés para las empresas ser capaces de optimizar o al menos mejorar sus servicios todo lo posible para poder ser competitivos en el mercado en el momento en el que nos encontramos.

2.1. Objetivos.

Este trabajo se centra en los sistemas de compartición de bicicletas, sistema en el que se define a la perfección este tipo de problemas, y más exactamente en el balanceo de los recursos de estos sistemas, ya que debido al uso de este se producen descompensaciones entre las diferentes estaciones para ciertos intervalos del día, empeorando la calidad del servicio para sus usuarios ya que puede dar lugar a la falta de bicicletas en estaciones cuando sean necesarias. Explorar mejoras en este proceso puede conllevar grandes ventajas y beneficios desde el punto de vista económico, medioambiental y temporal.

El objetivo es plantear una solución para este problema través de un algoritmo de Google denominado OR Tools, en el que se introducirán una serie de casos, planteados para siete días seleccionados, mediante dos métodos de asignación entre las estaciones que necesitan y las que tienen bicicletas de sobra, uno aleatorio y otro de mínima distancia. Mediante el algoritmo desarrollado se buscará obtener la ruta más corta para poder satisfacer la demanda de esas estaciones que requieren bicicletas dejando el sistema balanceado y listo para poder ofrecer un mejor servicio.

3. Estado del arte:

En este apartado se busca ofrecer una descripción de la problemática tratada por este trabajo final de grado, desde una visión general, centrado tanto en los sistemas de compartición de bicicletas como en el problema de enrutado de vehículos, como a los métodos de optimización específicos utilizados para resolver problemas de enrutado de vehículos. Además, se hace referencia a una serie de artículos que permiten tener un mayor conocimiento de los problemas y herramientas que se utilizan hoy en día por parte de la comunidad científica.

3.1. Sistemas de compartición de bicicletas:

Los sistemas de compartición de bicicletas son sistemas en los que los usuarios tienen acceso a una serie de bicicletas ubicadas en diferentes estaciones repartidas por toda la ciudad. Están compuestas por una serie limitada de plazas (i.e., aparcamientos para las bicicletas) y bicicletas, estando los usuarios obligados a, después de su utilización, dejarlas en las estaciones más próximas a su destino.

Estos últimos años se ha podido observar cómo está aumentando la implantación de estos servicios públicos a lo largo de todo mundo (Espegren et al., 2016). Actualmente se encuentran implantadas y en uso unos 948 sistemas, y además, otras 273 están en proceso de construcción, siendo datos de 2014 y que mostramos en la Figura 1.

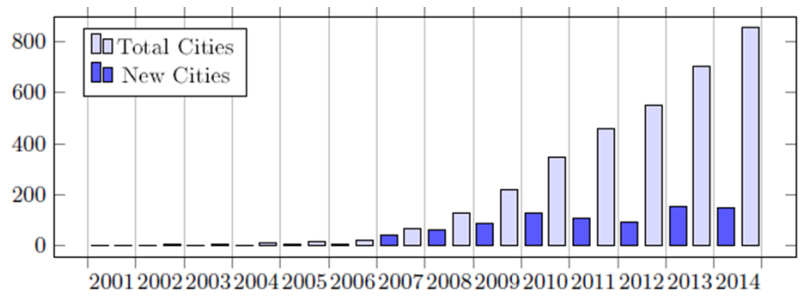


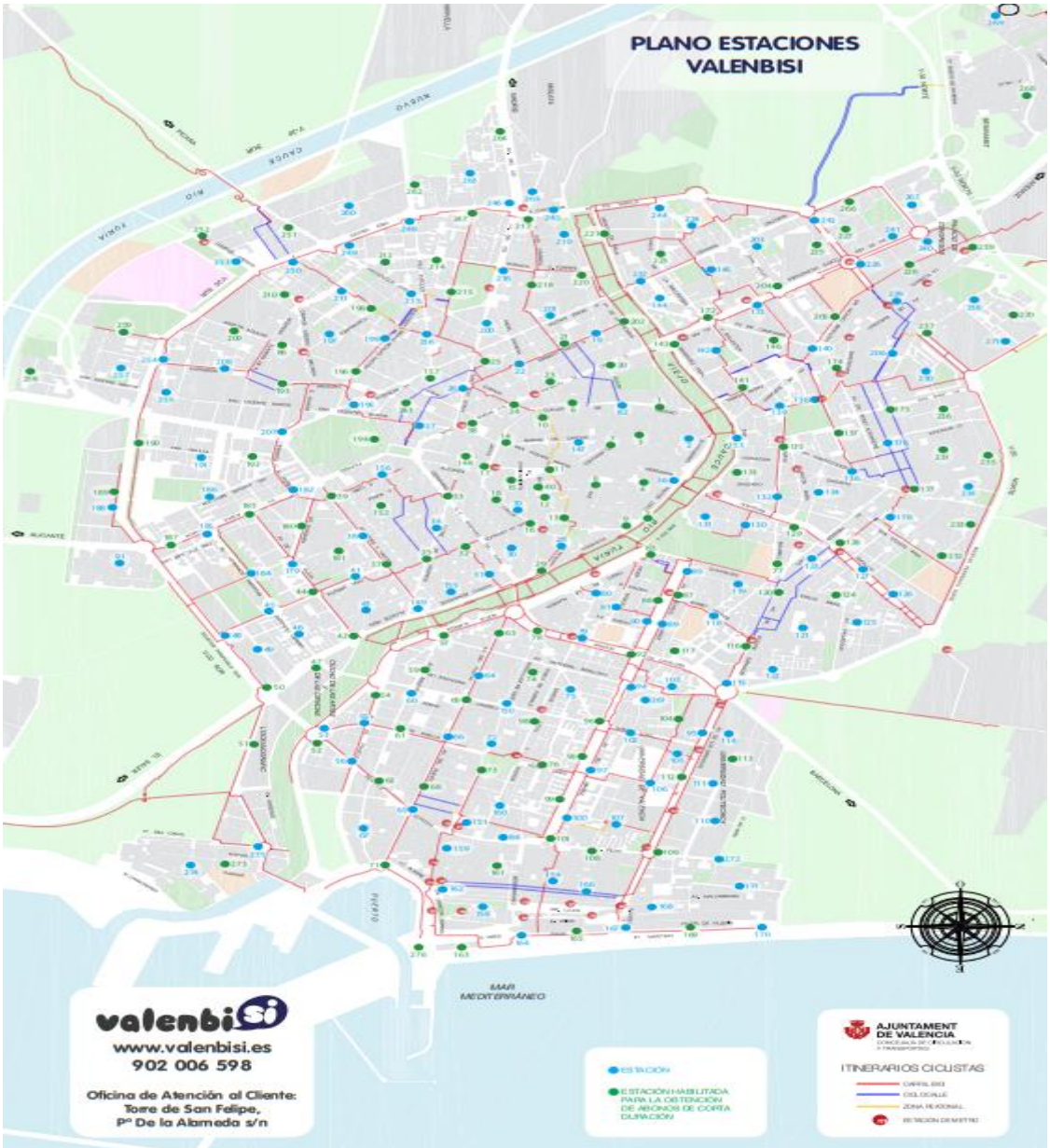
Figura 1: Gráfica acumulada total de ciudades con sistemas de bicibletas compartidas

(Espregen et al., 2016)

El sistema instalado en la ciudad de Valencia está formado por 276 estaciones y más de 2750 bicicletas, y recibe el nombre de Valenbisi[4]. En la Figura 2 se observa el plano en el que se ubican las 276 estaciones con su respectiva numeración, indicando las estaciones en las que se puede obtener acceso al servicio, además de información de la empresa. En

[4] http://www.valenbisi.es/

este sistema, a través de un pago anual, se tiene acceso a todo el servicio. Es decir, puede utilizar las bicicletas de todas las estaciones disponibles durante un tiempo (30 minutos) y, si se necesita, utilizar más tiempo dicha bicicleta pagando más en función del tiempo extra que la hayas usado. Este sistema fue implantado en el año 2010, impulsado por el Ayuntamiento de Valencia y encargándose de su gestión y explotación la empresa francesa JCDecaux[5].



Fuente: cas.valenbisi.es

Figura 2: Plano de estaciones Valenbisi

[5] https://www.jcdecaux.es/

Por desgracia, como consecuencia de la conducta y uso de los usuarios, se producen desequilibrios en el sistema. En muchas ocasiones se producen o falta de huecos en estaciones para poder dejar las bicicletas o a la inversa, faltan bicicletas en otras estaciones para que los usuarios puedan acceder al sistema (García-Palomares et al. 2016). A esta problemática se le conoce como el problema del balanceo en sistemas de compartición de bicicletas.

Más concretamente, en Espregen et al. 2016 se indica que este tipo de sistemas (y en general los servicios de transporte) presentan hasta tres tipos de niveles de problemas: estratégico, táctico y operacional. El primero de ellos hace referencia a los problemas que surgen a la hora de diseñar el sistema, por ejemplo, el número de bicicletas por estación, el número de estaciones. En cuanto al nivel táctico, se centra más en la propia distribución de las bicicletas entre las estaciones a lo largo de un día, por último, el nivel operacional consiste en buscar las rutas óptimas que deben seguir la flota de vehículos para balancear el sistema. Estos dos últimos problemas están directamente relacionados con la problemática de balancear en sistemas de compartición de bicicletas. Primero se debe decidir cuál es la distribución óptima de bicicletas a lo largo de las estaciones, y para ello debe emplearse una flota de vehículos que permita transportar bicicletas de una estación a otra a coste mínimo.

Fuera de esta clasificación de problemas también encontramos el factor económico. De acuerdo con algunos estudios (Van Essen et al. 2011) los costes externos producidos por atascos en el tráfico alcanzan entre el 1% y el 2% del PIB europeo. Esto hace que sea cada vez más importante, sobre todo para las pequeñas y medianas empresas, el tener en consideración a la hora de tomar decisiones este factor, que se ve afectado directamente por la búsqueda de rutas óptimas para la flota de vehículos.

El problema del balanceo de bicicletas puede ser clasificado, de acuerdo a su naturaleza, en función de diferentes parámetros:

- Estático o dinámico: Esta primera clasificación se centra en el momento en el que se produce el balanceo de los recursos por parte de la flota de vehículos. En el caso estático se produce a horas en los que no se hace uso del servicio por parte de los clientes, lo que suele ser por la noche o a horas muy tempranas de la mañana. Si se trata de un problema de balanceo dinámico, este rebalanceo se produce a lo largo del servicio. Este último es una versión del problema bastante más compleja por toda la serie de restricciones que se deben tener en cuenta, como el propio uso de las bicicletas o el tráfico de las ciudades, además de la cantidad de tiempo que se le tiene que dedicar y sobre todo el coste económico de mantener un servicio durante todo el día.

- Simétrico o asimétrico: Esta segunda clasificación hace referencia al coste de las rutas entre estaciones. El caso simétrico establece que el coste de ir de un nodo i a un nodo j es igual que el coste del camino inverso, es decir, del nodo j al nodo i. El caso asimétrico muestra una diferencia de coste entre esas dos rutas.
- Según objetivo: Aquí entra la intención del desarrollador del estudio. En algunos estudios (Espregen et al. 2016) buscan minimizar la desviación del número de bicicletas en cada estación con el número óptimo en el menor tiempo posible. También tenemos artículos (Cáceres-Cruz et al. 2014, Dell' Amico et al. 2013) donde se busca minimizar el coste total del recorrido satisfaciendo la necesidad de todos los clientes o minimizar el tiempo que se tarda en satisfacer a los clientes además del coste (Erdogan et al. 2014) o simplemente la longitud del recorrido de la ruta más larga (Schuijbroek et al. 2013).

Por estos motivos, con este proyecto buscamos analizar estos sistemas de compartición de bicicletas, aunque la aplicación y el caso de estudio se centrará en el sistema implantado en la ciudad de Valencia. Más concretamente, nos centraremos en el balanceo estático de bicicletas. Es decir, las bicicletas serán redistribuidas por la ciudad de Valencia de acuerdo a las necesidades de viaje en horas de uso mínimo del sistema de Valenbisi.

3.2. Optimización y programación matemática:

En este apartado queremos ofrecer una introducción al campo de la optimización, ya no solo del balanceo estático de recursos, dentro de una empresa o servicio, si no por los problemas de logística y transporte en general, o fuera también del ámbito de la industria. En cualquier servicio o sistema es de interés general desarrollar la mejor versión que se pueda ofrecer.

Podríamos definir como optimización al proceso de búsqueda de la mejor solución posible dados una serie de parámetros, restricciones, objetivos y variables para un problema que se plantee. Para comenzar, los principales elementos que componen los problemas de optimización se pueden clasificar en los siguientes:

- Variables (o alternativas de) decisión: Son las incógnitas o decisiones que deben determinarse conforme se vaya resolviendo el problema.
- Restricciones: Se tratan de las limitaciones que presenta el sistema y que deben tenerse en cuenta. Tecnológicas, legales, económicas y otras que van a restringir las variables decisión en un rango de valores que resulte factible.

- Función objetivo: Define la medida de efectividad que obtiene el sistema cuando los valores de las variables decisión con sus respectivos parámetros y restricciones, dan como resultado una mejora del sistema.

Dentro de la optimización, como se ha mencionado en la sección anterior, en algunos artículos se plantean soluciones óptimas (Raviv et al. 2013, Espegren et al. 2016) de diferentes parámetros para tratar de minimizar la desviación de la solución con respectos a estos.

Pero para ser más específicos a la hora de hablar de los diferentes métodos de optimización, tenemos que hacer una primera división entre dos métodos, cuya taxonomía queda reflejada en la Figura 3 (Cáceres-Cruz et al. 2016):

- Métodos exactos: Este tipo de métodos son utilizados para resolver problemas de menor tamaño, aunque también se está consiguiendo resolver problemas de mayor tamaño en un tiempo razonable, a pesar de ser más costosos. Como ejemplos de este tipo de métodos podemos encontrar el *Branch and Bound*, que se utiliza para resolver problemas de programación lineal entera pura y programación lineal entera mixta, o *Dynamic Programming,* que se centra en resolver problemas complejos mediante su división en problemas más simples. Luego tenemos el *Constraint programming,* el cual mediante el uso de restricciones desarrolla relaciones entre las diferentes variables. Una de las principales diferencias del *Constraint programming* con otros métodos de programación es que en este no hace falta definir una serie de pasos para ejecutar la solución del problema. Los modelos que utilizan este método están compuestos por tres elementos: variables, sus dominios o rangos y las restricciones que mencionamos anteriormente y que se encargarán de relacionar estas variables; y el funcionamiento se basa en hacer cada vez más reducidos los dominios y cada vez más restrictivas las restricciones o generar restricciones nuevas.
  Otros ejemplos de métodos exactos incluyen el método gráfico, que a partir de las funciones objetivo y las restricciones se genera una resolución visual del problema, pudiendo existir una única solución, infinitas soluciones, no tener una solución acotada o directamente no existir ninguna solución. Este método únicamente es empleado para problemas con dos variables de decisión. Por último, hacer referencia al algoritmo Simplex, este algoritmo se basa en la resolución de sistemas de ecuaciones lineales con el procedimiento de Gauss-Jordan apoyado con criterios para el cambio de solución básica. Se trata de un procedimiento iterativo que se aplica hasta que se cumple la condición de optimalidad. Este último método supone el algoritmo clásico para la resolución de problemas de programación lineal con variables continuas.

- Métodos aproximados: En este grupo encontramos los referidos a la heurística y meta-heurística. Estos dos métodos permiten encontrar soluciones aceptables para problemas específicos, pero no pueden asegurar la existencia de una solución óptima. Incluyen varias de las técnicas más reconocidas como la *Variable Neighborhood Search (VNS)*, que se basa en una sucesiva exploración de un conjunto de colonias/barrios predefinidos para encontrar la mejor solución para cada etapa, el *Large Neighborhood Search (LNS)* (Guimarans, 2012), una versión específica del VNS que trata de tener en cuentas varios de estas colonias o barrios a la vez, o el *Savings (CWS)* (Clarke y Wright, 1964), que es considerado uno de los métodos heurísticos más utilizados para la resolución de problemas de enrutado de vehículos con restricción de capacidad, que se centra en la estimación del ahorro que se produce mediante la fusión de rutas. Por otra parte, con respecto a la metaheurística, se puede hablar del *Ant Colony Optimization*, que refleja el comportamiento y comunicación de las hormigas en la búsqueda de la ruta más corta hacia fuentes de recursos, *Tabu Search*, en el que tras cada iteración se establece como solución la mejor solución de una parte del estudio (existiendo varias) o *Simulated Annealing* que se basa en el principio físico usado en el proceso de calentamiento y luego enfriamiento progresivo de una sustancia para producir una fuerte estructura cristalina (Nikolaev and Jacobson 2010).

  Como consecuencia de la alta complejidad de los problemas combinatorios como lo son el TSP y el VRP, en general se suelen emplear heurísticas y metaheurísticas para su resolución.

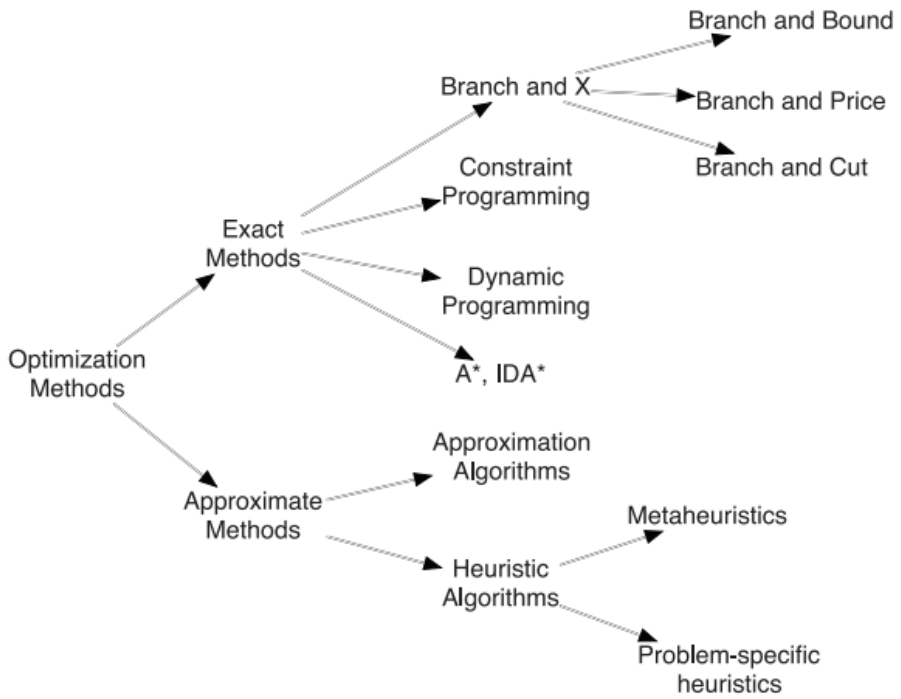


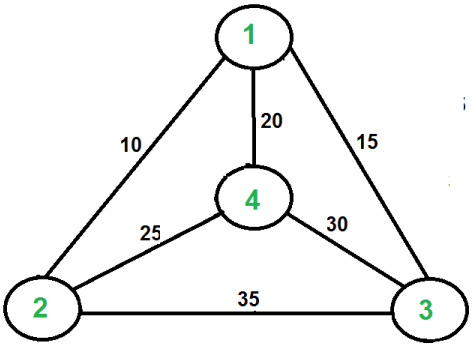
Figura 3: Clasificación de los métodos de clasificación clásicos

Fuente: Cáceres-Cruz et al. 2014

3.3. Problemas de enrutado de vehículos.

Una de las muchas aplicaciones en el mundo de la optimización es la del enrutado de vehículos (Cáceres-Cruz et al. 2015). En estos problemas se busca hallar la mejor ruta posible para un vehículo o una flota de vehículos que deben de visitar una serie de localizaciones o clientes.

Dentro de este tipo de problemas dominan dos tipos: *Travelling Salesman Problem (TSP)*, que busca la ruta más corta que pase una vez por cada nodo, comenzando y finalizando en el mismo nodo, a su vez que minimiza el coste total de los desplazamientos para un único vehículo como podemos observar en la Figura 4 (a), en el que se muestran los costes de cada trayecto, y *Vehicle Routing Problem (VRP),* que se trata de una generalización del TSP en el que se tienen disponibles múltiples vehículos que deben visitar una serie de localizaciones, como se ve en la Figura 4 (b), en la que de un nodo inicial se recorren, en este caso por varios vehículos, todos los nodos para luego regresar al nodo de origen. En la Figura, la ruta realizada por cada vehículo es diferenciada con un color.



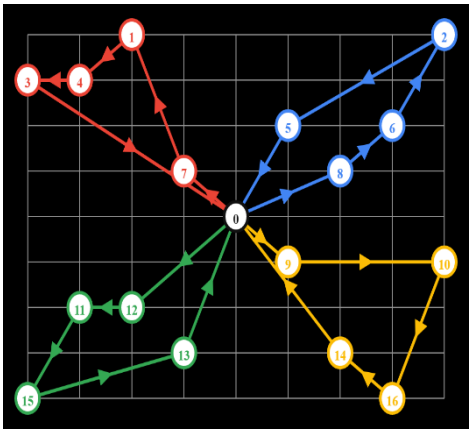


Figura 4:

(a) Ejemplo de TSP

Fuente: *geeksforgeeks.com*

(b) Ejemplo VRP

Fuente: *developers.google.com*

Después de ofrecer una primera clasificación, queremos desarrollar el VRP, el cual ha sido muy investigado o explotado estos últimos años (como hemos mencionado en anteriores apartados) y que aún se seguirá trabajando.

Algunos artículos (Golden et al. 2008) definen que el objetivo de los problemas de enrutado de vehículos es optimizar el diseño de las rutas, entre depósitos y clientes, de forma que las demandas queden satisfechas sin violar ninguna de las restricciones específicas del problema que se impongan.

Dentro de este grupo de problemas (VRP) podemos observar diferentes tipos, basándose estas diferencias principalmente en las restricciones empleadas (Cáceres-Cruz et al. 2014):

- *Assymetric cost matrix VRP:* Como mencionamos anteriormente, es el caso en el que ir del punto a al punto b no tiene el mismo coste que ir del punto b al punto a.
- *Distance-Constrained VRP:* La distancia de los arcos de las rutas no puede superar un valor máximo.
- *Heterogeneous fleet VRP (HVRP):* El caso en el que se utilizan vehículos de diferentes capacidades, por ello, las rutas tendrán que considerar la capacidad individual de cada uno de los vehículos. Dentro de los problemas de flota heterogénea podemos encontrar otras diferenciaciones:
    - *Fleet Size and Mix VRP:* Se considera un número ilimitado de vehículos.
    - *Site-Dependent VRP:* En este tipo de problema, se establece que un vehículo no puede visitar alguna ubicación por un motivo determinado.
    - *HVRP with Multiple use of vehicles:* Los vehículos pueden hacer más de un trayecto.
- *Multiple Depots VRP:* La compañía posee diferentes depósitos, esto también conlleva que algunos vehículos tengan diferentes lugares de origen o final.
- *Open VRP:* Las rutas pueden terminar en puntos diferentes a los depósitos.
- *Periodic delivery VRP:* El proceso no se realiza en un único día, lo que conlleva que no todos los clientes serán visitados en un día, aunque la planificación de las rutas si que sea diaria.
- *Pickup-and-delivery VRP:* A los clientes se les asocian dos cantidades, una de demanda para ser entregada y otra una cantidad que tiene que ser recogida de ese punto. Esto implica una restricción más en la capacidad de los vehículos, que se verán obligados a tener en cuenta estas dos cantidades para no sobrepasar en ningún momento la capacidad máxima del vehículo.
- *Split-delivery VRP:* Un cliente puede ser visitado por diferentes vehículos, siendo muy práctico para los casos cuyas demandas sean superiores a la capacidad de los vehículos,
- *Stochastic VRP:* Se trata de los casos en los que se tiene en cuenta alguna situación realista y aleatoria. Suelen hacer referencia al cliente, su demanda, el tiempo de servicio o el tiempo del trayecto entre clientes.
- *VRP with Backhauls:* Similar al PDVRP, los clientes se clasifican en dos tipos, unos a los cuales sólo se les podrá entregar una cantidad demandada (*linehaul costumers)* y los otros sólo podrán entregar una cantidad determinada *(backhaul costumers).* Una característica de este problema es que,

obligatoriamente, el primer grupo de clientes tienen que ser visitados antes que los segundos.

- *VRP with Time Windows:* En este problema se introduce la dimensión del tiempo, ya que obliga a que los clientes sean visitados a ciertas horas. Afecta tanto al tiempo de trayecto como al tiempo de servicio para cada cliente en su ubicación.
- *Green VRP:* En esta clasificación busca introducir componentes medioambientales en el proceso de optimización.
- *Rich VRP:* Una versión que trata de mostrar los principales atributos que forman un sistema de distribución de enrutado de vehículos (dinamismo, heterogeneidad, factores ambientales, legales y contractuales…) por lo que las soluciones obtenidas de este tipo de problemas pueden ser aplicables a un escenario real.
- *General VRP:* Se trata de una versión del PDVRP en el que las demandas no tienen porqué ser asignadas a un vehículo, entra en juego la toma de decisión de asignar a un vehículo de la flota o contratar a una empresa externa.

Analizando esta clasificación, se puede asociar los sistemas de compartición de bicicletas al grupo de problemas de enrutado de vehículos, ya que las características de estos sistemas pueden tener restricciones mencionadas en esta clasificación. Por ejemplo, las estaciones pueden ser de recogida o de entrega de bicicletas en función de su situación, el sistema puede tener diferentes depósitos, y el sistema puede estar formado por vehículos de diferentes tamaños, siendo una flota heterogénea. Como hemos mencionado anteriormente, en los sistemas de compartición de bicicletas tenemos que hacer frente a ese desequilibrio de recursos que se produce como consecuencia de la utilización del propio sistema. Este problema recibe concretamente el nombre de problemas de balanceo. En estos problemas una flota de vehículos perteneciente a este sistema será la responsable de corregir este desequilibrio. En resumen, presenta muchas restricciones y necesidades que suelen aparecer en los problemas de enrutado de vehículos, y por lo tanto se puede estudiar como uno.

Para estudiar algo más en profundidad el uso de las técnicas de optimización en el problema del enrutado de vehículos, hemos querido hacer referencia y analizar a tres artículos que plantean sus respectivas formulaciones para tres problemas diferentes. El primero (Asvin Goel and Volker Gruhn, 2006) desarrolla un problema de enrutado para una empresa de transporte aéreo. Una característica de este trabajo es el hecho que establece que no todos los requerimientos de transporte han de ser asignados a un vehículo, además que la flota de vehículos puede tener capacidades diferentes, así como tiempos y costes de trayecto diferentes. Entre otras características del modelo formulado por los autores, destaca que los vehículos no tienen que regresar a un depósito central, sino que a cada vehículo se le asigna una ubicación final. Por otro lado, todas las ubicaciones han de ser visitadas en una franja de tiempo, si visitan esa ubicación antes de tiempo les tocará

esperar y estarán obligadas a visitar las diferentes ubicaciones en una determinada secuencia/orden. El objetivo es encontrar diferentes rutas factibles buscando maximizar el beneficio (establecido como la diferencia entre las ganancias por los servicios prestados menos los costes de los trayectos de estas rutas). Una se considerará factible si para todas las órdenes se cumplen las restricciones de compatibilidad, de franjas horarias y de capacidad. La técnica utilizada para la obtención de estas soluciones es a partir de una serie de enfoques de mejora iterativas basadas en un LNS (*Large Neighborhood Search)*.

En el segundo artículo (Espegren et al., 2016) se desarrolla la formulación para un problema de balanceo estático de recursos de un sistema de compartimiento de bicicletas. El objetivo del sistema es minimizar la desviación en el número de bicicletas en cada estación con respecto a un valor óptimo en un tiempo límite. En lo referido a las características del problema, plantean una flota heterogénea de vehículos (de diferentes capacidades), el hecho de que un vehículo puede visitar un nodo varias veces, así como el hecho de que varios vehículos pueden visitar un mismo nodo. Además, se establece un tiempo de conducción entre las diferentes estaciones constante e independiente de la hora, y se establece un tiempo de aparcamiento en las diferentes estaciones. El tiempo de carga y descarga de bicicletas es proporcional al número de bicicletas que tenga el vehículo y se le sumará un tiempo de aparcamiento. Por último, se establece que cada nodo o estación será de recogida o descarga. Para la resolución del caso que plantean utilizan el *lower bound method* y el *upper bound method* concluyendo con la recomendación de este primer método ya que ofrece soluciones próximas a la óptima con un esfuerzo computacional menor.

Por último, el tercer artículo (Cáceres-Cruz et al., 2015) desarrolla una formulación que tiene como objetivo minimizar el coste total de la distancia del recorrido para satisfacer a todos los clientes además de una revisión bibliográfica acerca de los diferentes problemas de enrutado de vehículos. Las restricciones que establecen son las siguientes: la flota de vehículos es homogénea (misma capacidad todos los vehículos), el coste de las rutas es simétrico, cada nodo verá satisfecha su demanda por un único vehículo, toda la flota de vehículos del servicio comenzará y acabará la ruta en el depósito, que es denominado como nodo 0, los vehículos no podrán parar dos veces en un mismo nodo, y ningún vehículo podrá superar la capacidad máxima de este.

3.4. Modelos de optimización para el balanceo estático de sistemas de compartición de bicicletas.

En este apartado se muestran algunos ejemplos de cómo se ha ido resolviendo el problema del balanceo estático de bicicletas. En concreto, hemos querido mostrar los desarrollados por Dell'Amico et al. 2013, por Chemla, Meunier y Wolfler, 2012 y Cruz et al.

2016; se tratan de 3 artículos que muestran modelos de optimización de estos sistemas en tres casos diferentes, Reggio Emilia, en el norte de Italia, París, Francia y en el último artículo se simulan casos creados por los propios autores.

En el primer caso, se trata de una ciudad de 170.000 habitantes que posee un sistema de compartición de bicicletas formado por 13 estaciones, repartidas por la ciudad de la forma mostrada en la Figura 5, que muestra la ubicación de estas, y un número de bicicletas de aproximadamente 100. Funciona durante todo el día y queda cerrado por la noche, momento en el que se produce el rebalanceo de los recursos por un vehículo que visitará cada estación una sola vez.

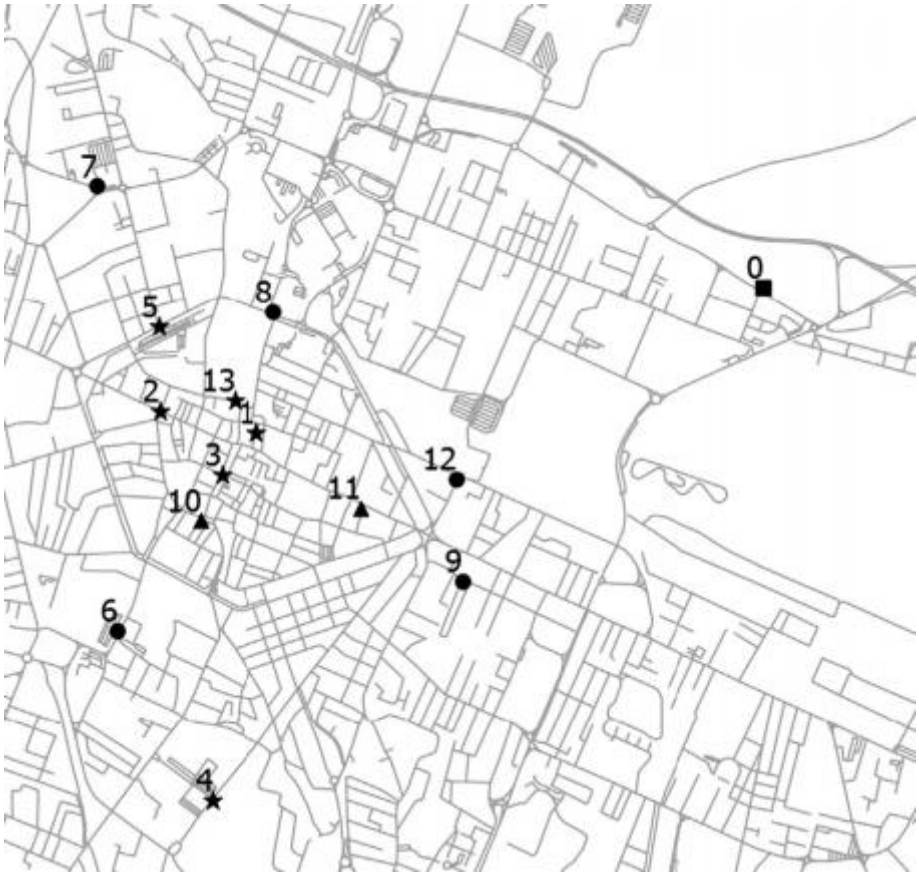


Figura 5: Sistema de compartición de bicicletas de Reggio Emilia.

Fuente: Dell'Amico et al. 2013

El problema planteado por los autores pertenece al grupo de los *Pickup and Delivery Vehicle Routing Problems (PDVRP)* en el que se nos da una serie de vértices que representan las estaciones, siendo el 0 la que representa al depósito, y una demanda que puede ser positiva o negativa en función de si en ella se tienen que retirar bicicletas o depositar (respectivamente). Está compuesto por una flota de vehículos de capacidad homogénea

que parten desde el depósito y, además, se les a asocia un determinado coste a los arcos entre los diferentes nodos o vértices. Para este problema se analizó el caso de que el coste fuera simétrico y asimétrico. El modelo tiene como objetivo hallar rutas para esa flota de vehículos minimizando el coste económico total teniendo que cumplir una serie de restricciones:

1. Todos los vehículos tienen una ruta que comienza y termina en el depósito.
2. Cada vehículo empieza la ruta vacío o con una carga determinada.
3. Cada estación es visitada una única vez y su demanda es satisfecha por el vehículo que la visite.
4. La cantidad total de bicicletas que requiera una ubicación y la posible carga inicial que pueda tener el vehículo nunca podrá ser negativa ni superior a la capacidad establecida de dicho vehículo.

Para la resolución del problema se decidió utilizar un modelo lineal entero mixto o *Mixed Integer Linear Programming* (MILP). Estos son modelos más complejos de resolver que los de programación lineal continua aunque, a diferencia de estos últimos, los modelos de programación entera tienen muchas menos soluciones que considerar. En este caso utiliza el método de *Branch and Cut.* Esta metodología permite resolver problemas lineales con un número muy elevado de restricciones. A partir de una solución inicial en el primer nodo, se inicia una iteración que selecciona un subproblema de las diferentes divisiones que se van realizando, buscando mejorar la cota superior que sustituirá a esa solución inicial.

En el segundo artículo (Chemla, Meunier y Wolfler, 2012), se plantea como un caso estático tanto para varios vehículos como para un único vehículo en una zona determinada. En este artículo se marcan como objetivo el de encontrar nuevas rutas con el mínimo coste posible, y como restricciones las siguientes:

1. Las bicicletas no se mueven, es decir, no están en uso al tratarse de un caso estático.
2. Dentro de una ruta, las estaciones pueden ser visitadas más de una vez.
3. Las bicicletas solo pueden ser recogidas en estaciones de recogida y entregadas en estaciones que requieran que se entreguen. Además, las estaciones que estén ya balanceadas son saltadas.
4. El tamaño de la ruta, referido al número de paradas, incluirá la salida y entrada del depósito.
5. Los vehículos saldrán y regresarán vacíos.

En este artículo, utilizan para su resolución también se utiliza el algoritmo de *branch-and-cut.*

En el último (Cruz et al. 2016), para la resolución utilizan un modelo híbrido de un ILS (*Iterated Local Search),* los cuales se ha demostrado que son muy efectivos a la hora de resolver problemas de enrutado de vehículo de gran tamaño incluyendo aquellos que solo utilizan un vehículo (referenciar artículo). El objetivo es encontrar la ruta de menor coste que comience y termine en el depósito, que visite las estaciones que tengan una demanda diferente de 0 al menos una vez y por último las estaciones pueden ofrecer servicios temporales como depósito, por ejemplo, aportando más bicicletas que las que marcan su demanda inicial o guardando más bicicletas (siempre que no se supere su capacidad máxima) y en ambos casos sus demandas serán satisfechas en visitas posteriores.

3.5. Herramientas de optimización: OR Tools

En esta última parte del estado del arte, hemos querido hacer referencia a las herramientas de optimización utilizadas en este tipo de problemas. La que hemos utilizado en nuestro caso ha sido ORTools de Google[6], que ofrece un paquete, o suite, de software público para tratar ya no solo problemas de enrutado de vehículo, también abarca problemas de optimización en redes, problemas de programación lineal, y problemas de programación entera.

En lo que respecta a la programación de enrutado de vehículos y el *Travelling Salesman Problem,* se muestra un ejemplo práctico y su codificación, explicando paso a paso la resolución de este tipo de problemas y proponiendo soluciones en diferentes lenguajes: Python, C++, Java y C#.

Además, nos permite tratar el problema de enrutado de vehículos con algunas restricciones como son las de capacidad de los vehículos, las ventanas de temporales, las de los propios recursos como espacio o personal para cargar o descargar los vehículos en el depósito o con la posibilidad de saltarse alguna visita, en este caso no es obligatorio que los vehículos pasen por todos los nodos o clientes de la ruta, pero tendrán que pagar una penalización por cada nodo/cliente que no visiten.

En este tipo de problemas, la cantidad de tiempo que puede tardar en resolverse el problema crece exponencialmente conforme crece el problema, por lo que el software puede tardar demasiado tiempo en devolver una solución óptima, por lo que normalmente

[6] https://developers.google.com/optimization

ORTools ofrece una solución buena para este tipo de problemas de grandes dimensiones, por todo ello usa diferentes métodos heurísticos para la resolución de los problemas de enrutado de vehículos. Para la programación y resolución del modelo se utilizará la API de Python.

Para este tipo de problemas se pueden utilizar otras herramientas como es Concorde[7]. Su uso principal es la resolución del problema del comerciante y además con la condición de ser simétrico. No obstante, con la configuración apropiada, puede ser utilizado en otro tipo de problemas de optimización. Presume de haber llegado a resolver problemas de hasta 85900 ciudades (David L. Applegate, Robert E.Bixby, Vasek Chvátal y William J.Cook, 2006). En este caso, se ha utilizado OR Tools debido a que el problema es un *Vehicle Routing Problem (VRP),* que no es de grandes dimensiones y cuyas restricciones no pueden ser incluidas en herramientas como Concorde, ya que en nuestro caso las distancias no son simétricas.

[7] https://www.math.uwaterloo.ca/tsp/concorde.html

4. Análisis del Sistema Valenbisi

El objetivo de esta sección es el de mostrar un primer acercamiento al problema del balanceo estático de bicicletas de Valenbisi, y para el cual plantearemos un modelo para su optimización,. Se trata de un primer análisis de la situación del sistema de Valenbisi y su uso, estudiando el comportamiento de los usuarios para poder entender cómo funciona y particularidades que puedan diferenciarlas de otros sistemas o particularidades que haya que tener en cuenta a la hora de preparar las instancias o nuestro modelo de optimización.

4.1. Introducción a la base de datos:

A partir de un análisis en profundidad de los datos de las estaciones se puede justificar el interés en desarrollar estrategias de balanceo de estos recursos por parte de los responsables de estos sistemas. En esta parte del trabajo se lleva a cabo un análisis del uso de las diferentes estaciones que componen Valenbisi (276) centrándose en el número de entradas y salidas de bicicletas según el día de la semana y la hora. El objetivo de este análisis es determinar qué posibles aspectos influyen en la demanda y uso del sistema de Valenbisi. Estos aspectos serán tenidos en consideración cuando se creen las instancias del problema del balanceo de bicicletas. Más concretamente, el objetivo de este análisis es conseguir predecir el estado de las diferentes estaciones a ciertas horas del día y según el día de la semana para estimar la forma en la que se tiene que producir el balanceo de las bicicletas por parte de la flota de camiones que tiene la compañía. Debe considerarse que el objetivo de este trabajo final de grado no es el de desarrollar algoritmos predictivos exactos, sino estudiar el impacto de diferentes algoritmos de optimización sobre las rutas recorridas por los camiones de la compañía. No obstante, para poder generar instancias razonablemente reales de balanceo es necesario tener una estimación aproximada de qué tipos de desbalances pueden ocurrir en el sistema.

4.2. Descripción de los datos y objetivo del análisis:

Los datos analizados fueron recopilados para el desarrollo del artículo Kull et al. 2015. En el artículo se ofrece información sobre las estaciones de compartición de bicicletas de 27 ciudades y 11 países, alcanzando un total de 3584 estaciones analizadas, acompañándolo además con información de la climatología de estas ciudades y busca analizar la demanda teniendo en cuenta factores como el día de la semana, la estación del año, el tiempo, la ubicación y los eventos para poder llevar una planificación precisa del proceso de balanceo de recursos.

Se ha tenido acceso a estos datos utilizados en el artículo a partir del servicio de datos públicos de la UPV [8] y en estos se recoge la información, hora a hora desde el 26 de septiembre de 2014 hasta el 2 de julio de 2015, del estado y uso de las 276 estaciones que componen este sistema. Para ser exactos, la información que nos ha sido de interés ha sido el número de bicicletas que han salido en esa determinada hora y el número de bicis que se han dejado en esa hora en una determinada estación, como se muestra en la Figura 6, en el que la variable *totinc* hace referencia al número de bicicletas que se han dejado en la estación, *totdecr* al número de bicicletas que han abandonado la estación y *lastbikes* hace referencia al número de bicicletas que encontramos en la estación al final de la hora. Los campos de *day*, *month*, *year*, y *hour* hacen referencia a la fecha y hora en la que se tomó la instantánea del estado de la estación.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	day	month	year	hour	houred	totinc	totdecr			lastbikes			
2	26	9	2014	10	392145	0	0			13			
3	26	9	2014	11	392146	2	5			11			
4	26	9	2014	12	392147	3	4			10			
5	26	9	2014	13	392148	3	3			10			
6	26	9	2014	14	392149	3	3			10			
7	26	9	2014	15	392150	1	5			6			
8	26	9	2014	16	392151	1	1			6			
9	26	9	2014	17	392152	4	7			3			
10	26	9	2014	18	392153	7	2			7			
11	26	9	2014	19	392154	4	2			9			
12	26	9	2014	20	392155	4	5			8			
13	26	9	2014	21	392156	4	1			11			
14	26	9	2014	22	392157	2	3			11			
15	26	9	2014	23	392158	1	5			7			
16	27	9	2014	0	392159	1	2			5			
17	27	9	2014	1	392160	1	4			2			
18	27	9	2014	2	392161	0	2			0			
19	27	9	2014	3	392162	0	0			0			
20	27	9	2014	4	392163	0	0			0			
21	27	9	2014	5	392164	0	0			0			
22	27	9	2014	6	392165	0	0			0			

Figura 6, información sobre una de las estaciones en la base de datos de Valenbisi

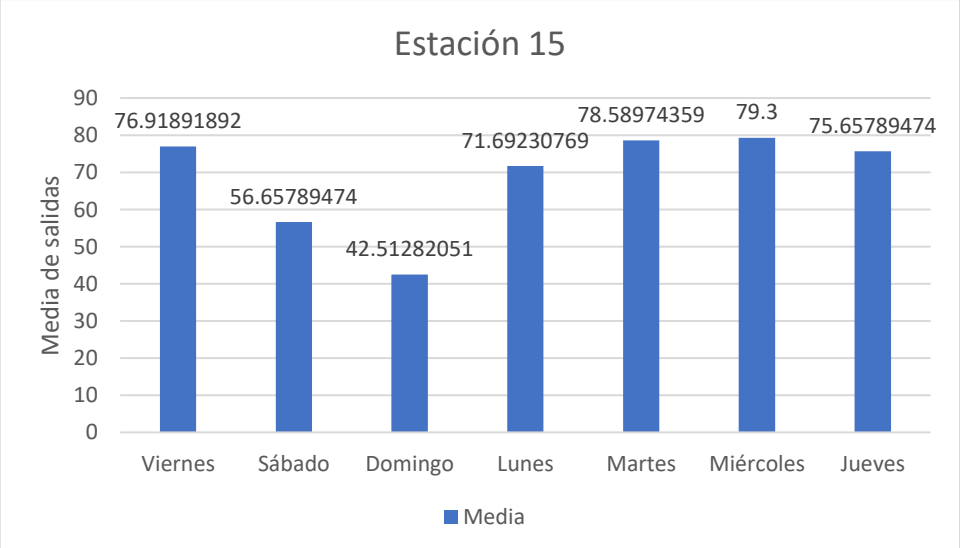
4.3. Análisis de los datos.

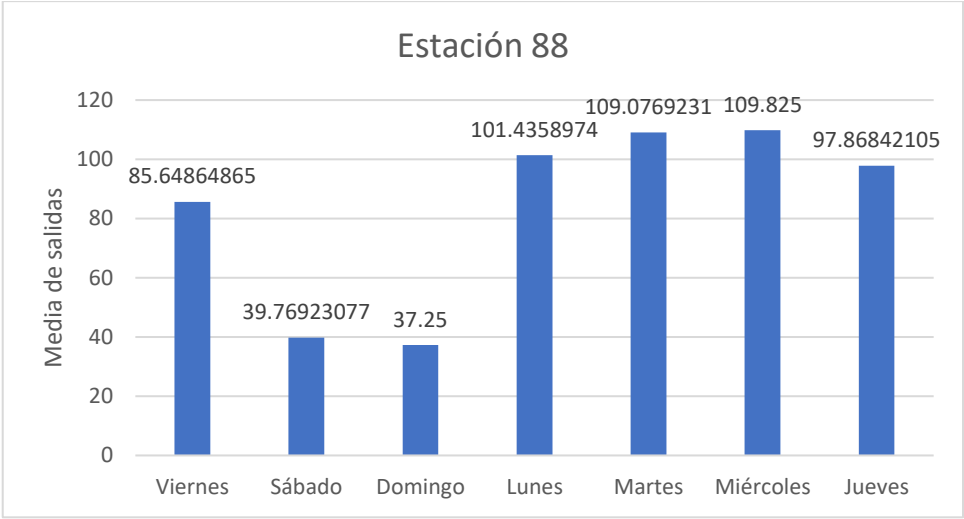
En esta sección, se ofrece un análisis de 12 estaciones seleccionadas en función de la ubicación, tratando de abarcar toda la superficie posible de la ciudad y como muestra, aproximada, de las diferentes estaciones que se encuentran alrededor de estas. El motivo por el que han sido seleccionadas estas doce se basa en que permiten representar el uso que se les da a las diferentes estaciones según se encuentren en zonas residenciales, zonas estudiantiles o académicas, y también zonas turísticas o centro de negocios de la ciudad. Dentro de la clasificación de zona residencial permiten comprobar que tendencia se tiene en cada zona, más o menos alejadas del centro de la ciudad o de zonas de interés turístico o empresarial. Las estaciones, identificadas numéricamente, son las siguientes:

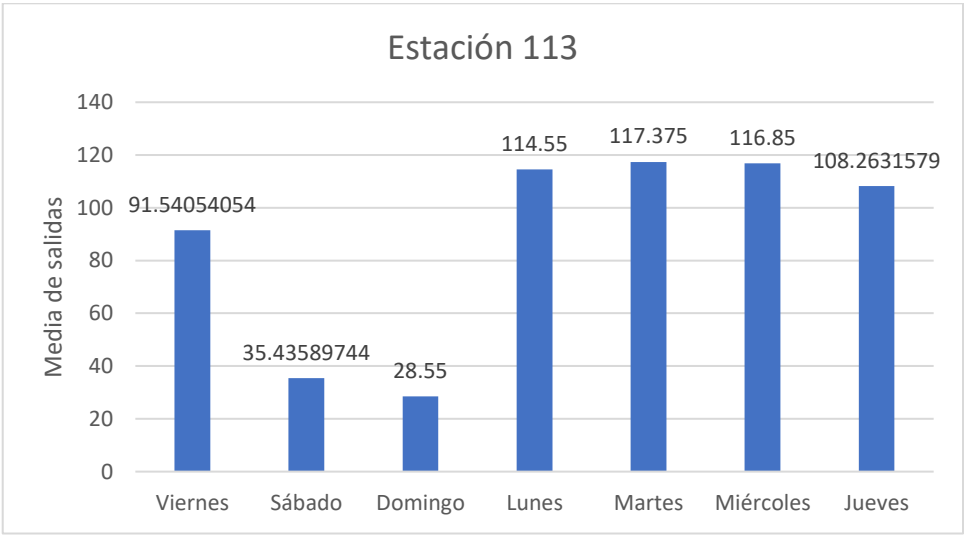
[8] https://dataupv.webs.upv.es/datos-historicos-de-valenbisi/

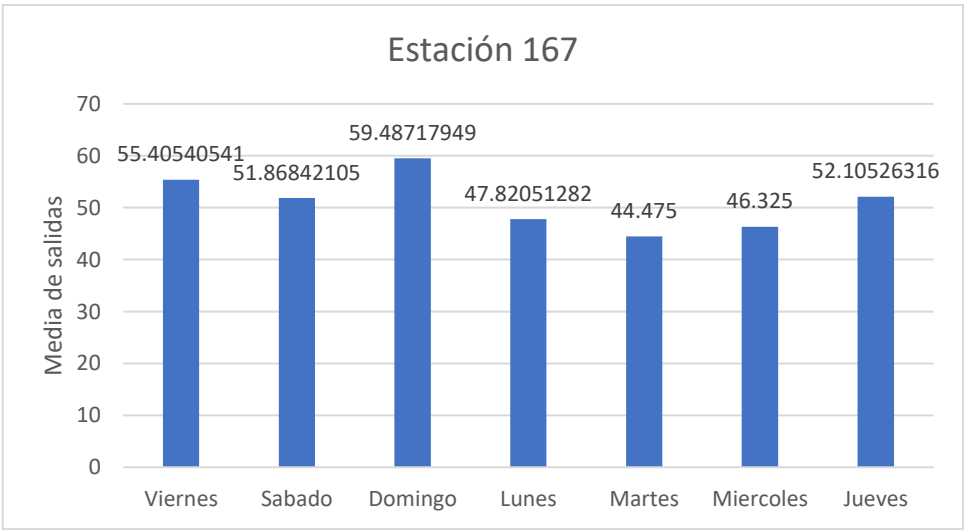
- 15, en la Plaza del Ayuntamiento. (Zona turística y de negocios)
- 21, en el cruce entre la Calle Juan Llorens y Literato Gabriel Miró. (Zona residencial)
- 53, ubicada en el Paseo de Alameda a la altura de la rotonda de El Corte Inglés. (Zona de interés comercial)
- 63, en la Avenida del Puerto. (Zona residencial)
- 73, ubicada en la Calle Jerónimo Monsoriu (zona residencial).
- 88, en la Avenida Blasco Ibáñez, a la altura de la Facultad de Geografía e Historia. (Zona estudiantil)
- 113, dentro de la Universidad Politécnica de Valencia, en la Biblioteca General. (Zona estudiantil)
- 120, entre las calles Doctor Vicent Zaragozá y Emilio Baró, en la zona de Benimaclet. (Zona residencial y estudiantil)
- 135, en el cruce de la Avenida de la Constitución con la Calle Reus (zona residencial).
- 167, en Calle de la Acequia de la Cadena, ubicada en la zona de la Playa la Malvarrosa. (Zona de interés turístico)
- 179, en el cruce entre las calles Zapadores y Avenida de la Plata, barrio de Ruzafa. (Zona residencial y próxima al centro)
- 197, en el cruce entre Carrer de Fontanars dels Alforins y Calle Jacinto Labaila, en la zona de Patraix. (Zona residencial).

El primer paso para el análisis de los datos fue estudiar el efecto del día de la semana sobre la cantidad de salidas de bicicletas de las 12 estaciones seleccionadas. Tras esta clasificación se recogió de forma gráfica los valores medios para cada uno de los días de la semana en cada una de las estaciones. Como ejemplo de este tipo de gráficos, se obtuvieron las siguientes gráficas para las estaciones 15, 88, 113 y 167 (Figura 7).





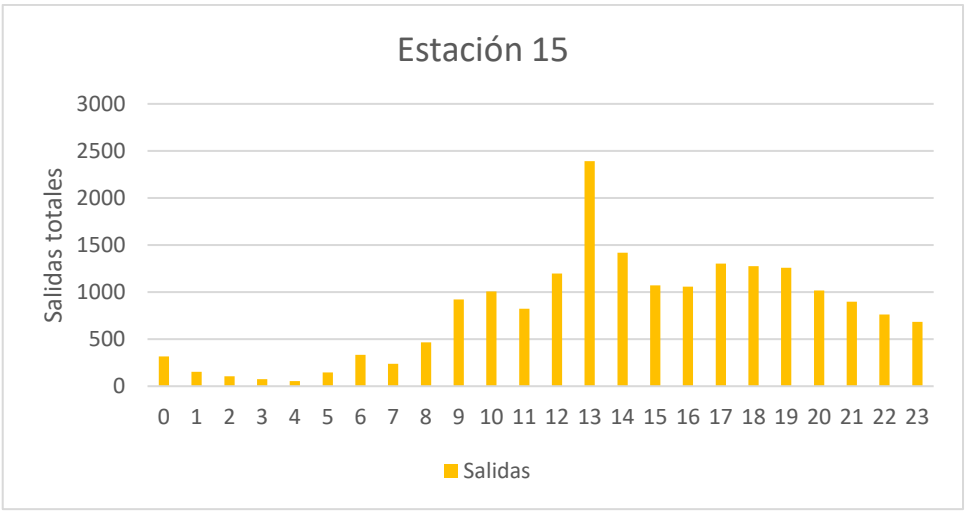


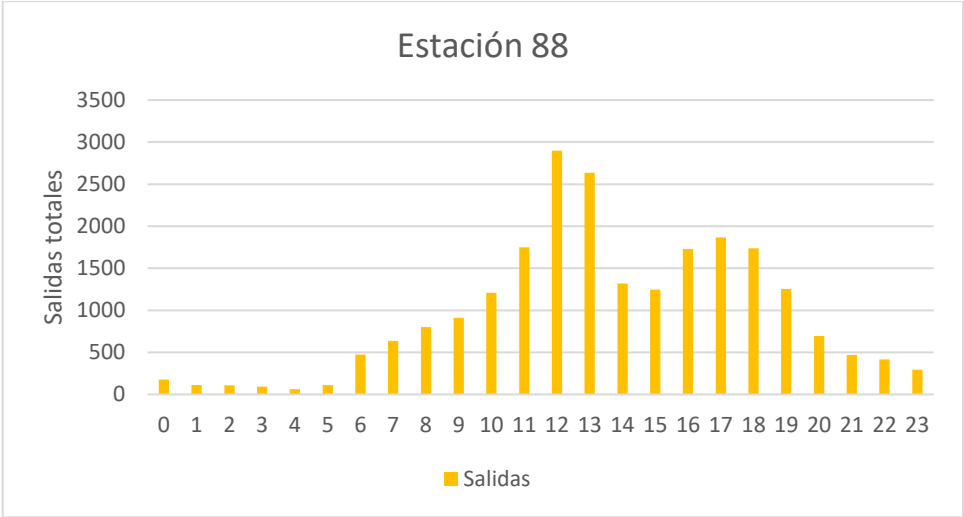


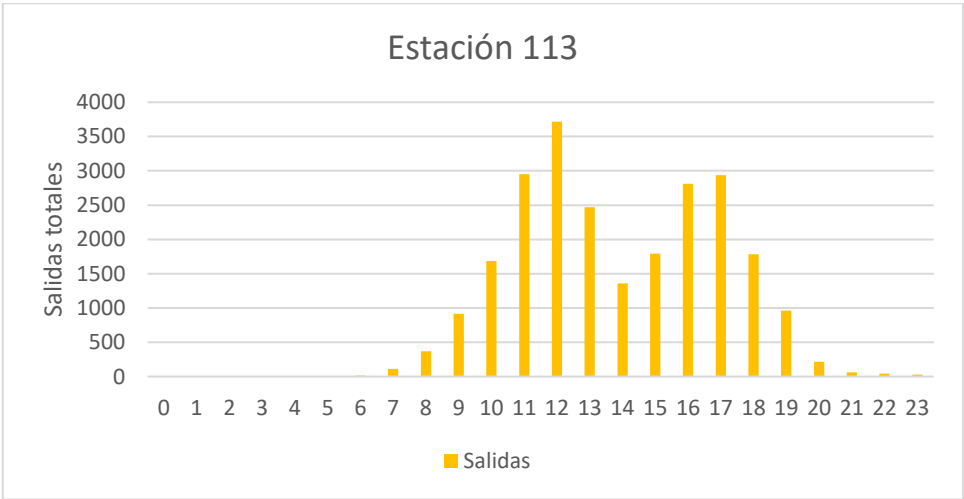
Figuras 7, información según el día de la semana de las estaciones 15, 88, 113 y 167

Como se puede observar en la Figura 7 los resultados muestran una gran influencia, en mayor o menor medida, del día de semana en el uso de las diferentes estaciones. Esto se refleja en la gran diferencia que se puede apreciar entre el número de salidas que se producen los fines de semana con cualquier día de entre semana, incluso entre los días de entresemana se aprecia como hasta los miércoles se produce un aumento diario del número de bicicletas que salen para después, conforme nos acercamos al fin de semana, empezar a disminuir. Encontramos la excepción en la estación 167, ubicada en la zona de la Malvarrosa, que refleja un aumento del uso en los fines de semana, como consecuencia de ser una zona de gran interés turístico y de tratarse de días no laborables que permiten a los usuarios del sistema visitar la zona. También podemos observar como la estación 15, situada en la Plaza del Ayuntamiento, no sufre una caída tan brusca como la estación 113, situada en la Universidad Politécnica de Valencia, ya que se trata de una zona turística a la vez que de negocios. Estos patrones se repitieron en el resto de las estaciones analizadas. Estos resultados sugieren que el día de la semana influye sobre la demanda y uso de las estaciones de bicicleta. Por tanto, a la hora de determinar la demanda de una estación para un instante dado, debería tenerse en consideración el día de la semana en el que nos encontramos.

Además, para completar este estudio y la información que aporta el conjunto de datos, se ha realizado un segundo estudio buscando analizar la influencia de la hora del día en las salidas de bicicletas de cada una de las estaciones estudiadas, en el documento se muestran unos ejemplos en la Figura 8:







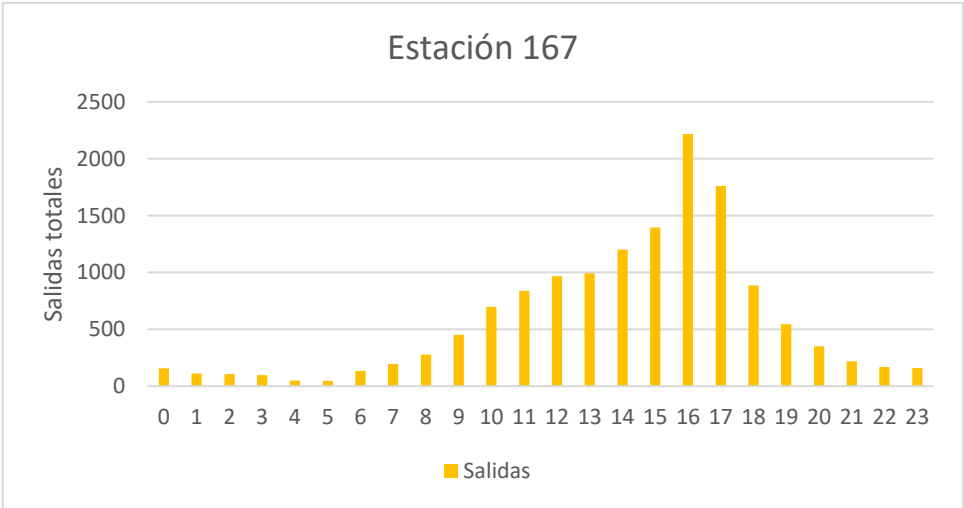


Figura 8, salidas totales por hora en las estaciones 15, 88, 113 y 167

Se puede apreciar con claridad en las figuras 12-15 como afecta la hora del día al uso de las diferentes estaciones, cabe destacar como a mediodía se produce un pico de salidas que podrían estar asociadas a salidas tanto de centros educativos como de puestos de trabajo, y este proceso se repite a media tarde, coincidiendo con los mismos grupos, es decir, con salidas de centros educativos o de puestos de trabajo. En el caso de la estación 167 (figura 15), ubicada en la zona de la Malvarrosa, es característico que solo se produce un único pico a las 16 horas, reflejando una tendencia de visita a la ubicación.

Este análisis permite estimar a qué hora se debe realizar el rebalanceo para cada una de las estaciones. Queda claro que el sistema es utilizado en su mayoría a partir de media mañana y por las tardes, teniendo que quedar preparado con antelación, por ello es de interés llevar a cabo este rebalanceo por las mañanas temprano o a últimas horas de la tarde, incluso se podría admitir por las noches ya que apenas se producen salidas en ninguno de los casos. Por este motivo, la franja horaria seleccionada para llevar a cabo este proceso es entre las 22:00 y las 00:00.

4.4. Interpretación del análisis:

Estos dos estudios tienen una gran influencia a la hora de la estimación de la demanda que se realiza en este trabajo. Reflejan la gran influencia que tienen el día de la semana y la hora dentro del uso de cada una de las estaciones, y por lo tanto se tratan de factores que se deben tener en cuenta. Ya no solo por separado, también conjunto, es decir, como es la demanda para ciertas horas determinadas de ciertos días de la semana determinados. A partir de esta información de salidas y entradas de las bicicletas, se puede llegar a establecer un valor objetivo para cada una de las estaciones que nos indique si están preparadas para atender las necesidades del sistema, además de permitirnos establecer la mejor franja horaria en la que llevar a cabo el proceso de balanceo como se ha mencionado en el apartado anterior (4.3. Análisis de los datos.). Esta información será posteriormente empleada para la creación de instancias, o lo que es lo mismo, escenarios con tintes realistas para probar las soluciones propuestas para el balanceo estático de bicicletas. Esto será cubierto en la próxima sección.

5. Preparación de problemas y datos.

Recordemos que el objetivo de este trabajo es el del desarrollo de soluciones para el problema del balanceo estático de bicicletas en el sistema de Valenbisi. Con el fin de poder evaluar las soluciones propuestas, es necesario llevar a cabo una simulación lo más realista posible del sistema de compartición de bicicletas de Valencia, Valenbisi.. Para llevar a cabo esta simulación, es necesario crear ejemplos de problemas en los que sea necesario balancear las bicicletas en el sistema de Valenbisi. A cada uno de estos problemas las llamaremos instancia, y describirán el problema que debe resolverse. Más concretamente, la instancia definirá qué estaciones necesitan bicicletas (y cuántas) y qué estaciones pueden dar bicicletas (y cuántas). Con el fin de que estas instancias sean realistas y puedan representar situaciones reales en el sistema de Valenbisi, se ha llevado a cabo una creación de instancias apoyada en el análisis de datos llevado a cabo en la sección anterior. Se ha empezado por proponer un modelo para determinar el flujo medio de bicicletas para cada una de las estaciones, atendiendo tanto al número de entradas y salidas de bicicletas, como al número de estas que se encuentran en las estaciones. Nótese, que el objetivo de este trabajo no es el de realizar una predicción completamente fidedigna y exacta del uso de sistemas de bicicletas ya que se trataría de un tema que podría abarcar otro trabajo completo. Simplemente se busca la creación de instancias con ciertos tintes de realidad.

5.1. Modelo propuesto para describir el flujo de bicicletas en una estación

Para esta parte se asume que tanto la tasa de salidas como de llegadas de bicicletas a las estaciones sigue una distribución de Poisson, ya que se trata de una observación de hechos que suceden cada cierto período de tiempo y tienen naturaleza aleatoria, entre otros motivos. Es decir, una distribución de Poisson para las llegadas de bicicletas y una distribución de Poisson para las salidas de bicicletas de dicha estación.

Como se ha demostrado ya, el día de semana y la hora del día influyen en el uso de bicicletas en la ciudad de Valencia. Esto implica que la tasa de llegada y salida de bicicletas para una estación puede ser influida tanto por el día de la semana como por la hora del día. Por ello, se asume una distribución de Poisson diferente tanto para entradas como para salidas, para cada combinación de hora y día de la semana.

Una distribución de Poisson es una distribución de probabilidad discreta que se aplica a las ocurrencias de un suceso durante un intervalo determinado, a las probabilidades de que ocurra un determinado suceso durante un tiempo determinado. La función de probabilidad de la distribución es la siguiente:

$$P(x) = \frac{e^{-\lambda} * \lambda^{x}}{x!}$$

Siendo λ el promedio de ocurrencias en un intervalo, x representa el número de ocurrencias y P(x) se definiría como la probabilidad de ocurrencia cuando la variable x toma un valor finito.

Atendiendo a lo comentado, definimos $X_{i,j,k}$ como una variable aleatoria que representa el flujo de bicicletas para una estación dada en un día de la semana y hora determinados, donde $i = \{E, S\}$ representa un flujo entrante o saliente de bicicletas, $j = \{1,2,3, \dots ,7\}$ representa el día de la semana, y $k = \{0,1,2, \dots ,23\}$ representa la hora del día. Como se ha comentado, asumimos que esta variable sigue una distribución de Poisson $X_{i,j,k} \sim Ps(\lambda_{i,j,k})$. Por tanto, $\lambda_{i,j,k}$ es el parámetro que representa la tasa de entrada/salida de bicicletas en una hora para dicha estación en el día de la semana y hora especificada.

Como resultado del análisis elaborado en la sección anterior, se ha observado que el menor número de salidas se produce entre las 22:00 y las 05:00 por lo que sería de gran interés llevar a cabo este proceso de balanceo estático durante la madrugada (00:00-02:00), ya que sería la franja horaria en la que se produciría menor diferencia entre lo estimado en los análisis y la realidad como consecuencia del escaso número de salidas que se producen en todas las estaciones, habiendo casos en los que no se llega a producir ninguna, por lo que no se estorba al usuario ni se paraliza el sistema. Además se puede aprovechar un menor tráfico para poder hacer los desplazamientos en el menor tiempo posible.

A partir de la situación de las estaciones a las 00:00, se determinan dos variables aleatorias, la del flujo de entradas de bicicletas hasta las 07:00, que se trata de una variable aleatoria que es la suma de los flujos de entrada desde las 00:00 hasta las 07:00, y la de flujo de salidas de bicicletas hasta las 07:00, que se trata de una variable aleatoria que es la suma de flujos de salida desde las 00:00 hasta las 07:00. En ambos casos, ambas variables aleatorias son el resultado de la suma de variables de Poisson. En principio, con el fin de simplificar el modelo, asumiremos independencia entre las variables aleatorias de Poisson que componen las sumas. De forma formal, definimos estas variables como:

$$X_{i,j,0-7} = X_{i,j,0} + X_{i,j,1} + \cdots + X_{i,j,6}$$

donde $i = \{E, S\}$, hace referencia a si el flujo es de entrada o salida. Al asumir independencia, el resultado es una nueva variable de Poisson $X_{i,j,0-7} \sim Ps(\lambda_{i,j,0-7})$ cuyo parámetro se puede calcular como:

$$\lambda_{i,j,0-7} = \lambda_{i,j,0} + \lambda_{i,j,1} + \cdots + \lambda_{i,j,6}$$

El flujo total entre las 00:00 y las 07:00 para un día de la semana j es una nueva variable aleatoria:

$$X_{j,0-7} = X_{E,j,0-7} - X_{S,j,0-7}$$

De nuevo, con el fin de simplificar el modelo, asumimos independencia entre ambas variables aleatorias. La resta entre dos variables aleatorias de Poisson independientes sigue una distribución de Skellam $X_{j,0-7} \sim Sk(\lambda_{E,j,0-7}, \lambda_{S,J,0-7})$, siendo la media de esta la resta entre el parámetro de la primera Poisson y el de la segunda. Es decir, $E(X_{j,0-7}) = \lambda_{E,j,0-7} - \lambda_{S,J,0-7}$.

La distribución de Skellam consiste en una distribución de probabilidad que gobierna la diferencia entre dos variables aleatorias independientes de Poisson de distinto valor esperado. En nuestro caso, se ha utilizado como consecuencia de llevar a cabo la diferencia de dos variables aleatorias como son el flujo de entradas y salidas de bicicletas de las estaciones entre las 00:00 y las 07:00 para poder determinar la situación de cada una de las estaciones.

Por tanto, el modelo propuesto para definir el flujo de bicicletas depende los parámetros $\lambda_{i,j,k}$ de cada una de las distribuciones de Poisson que participan en el modelo. Para estimar estos parámetros, se ha optado por una estimación por máxima verosimilitud para cada combinación de estación de bicicletas, tipo de flujo (entrada o salida), día de la semana y hora del día. De acuerdo con la estimación por máxima verosimilitud, $\lambda_{i,j,k}$ no es más que la media muestral de las observaciones para cada combinación de estación de bicicletas, tipo de flujo (entrada o salida), día de la semana y hora del día[9].

5.2. Selección de casos de estudio:

Posteriormente, se llevó a cabo una selección de días aleatorios de entre todos los que había en la base de datos para observar la creación de instancias. Se buscó crear una serie de instancias que recogieran el comportamiento de los usuarios esos días y el estado

[9] Taboga, Marco (2017). "Poisson distribution - Maximum Likelihood Estimation", Lectures on probability theory and mathematical statistics, Third edition.

real del sistema, así como representar diferentes casuísticas y escenarios que se puedan dar dentro del sistema de Valenbisi. Los días seleccionados fueron:

- Sábado 27 de septiembre de 2014.
- Martes 11 de noviembre
- Lunes 15 de diciembre
- Domingo 1 de febrero de 2015.
- Viernes 27 de marzo.
- Jueves 21 de mayo.
- Miércoles 1 de julio.

El criterio utilizado para la selección de estos días fue que al final se pudiese representar el comportamiento en los siete días de la semana. Como se vio con anterioridad, el día de la semana influía en el uso de este servicio. Además, se decidió que hubiese cierta distancia entre las fechas, pudiéndose reflejar factores como la meteorología, factor clave en una ciudad como Valencia, y la influencia de ciertas épocas del año como son las épocas de vacaciones o de exámenes en los diferentes centros universitarios.

5.3. Generación de instancias:

Para la generación de las instancias asociadas a los días mencionados, se siguieron los siguientes pasos:

I. Primero se obtuvo el número de bicicletas que se encontraban en cada una de las estaciones a las doce de la noche, para cada uno de los días seleccionados.

II. A continuación, se calcularon las $\lambda_{E,j,0-7} - \lambda_{S,J,0-7}$ de entradas y salidas para cada una de las estaciones en base al histórico de datos para dicha estación, teniendo en cuenta el día en el que nos encontrábamos en dicha estancia. Concretamente, de acuerdo a la estimación por máxima verosimilitud, las $\lambda_{i,j,k}$ fueron estimadas como la media muestral de las observaciones. Para calcular las $\lambda_{i,j,0-7}$ se sumaron las $\lambda_{i,j,k}$ estimadas. Finalmente, la media o valor esperado para el flujo total de bicicletas se calculó como $E(X_{j,0-7}) = \lambda_{E,j,0-7} - \lambda_{S,J,0-7}$, obteniendo lo que sería la medía de entradas y salidas para el intervalo de tiempo que interesa. Este valor nos indica si a lo largo de esas horas el número de bicicletas aumenta, disminuye o se mantiene, en promedio.

III. Este valor, $E(X_{j,0-7})$, habrá que sumarlo al número de bicicletas que se encuentran a las doce y que habíamos obtenido anteriormente. Este valor

nos da una estimación del número de bicicletas que esperamos tener a las 07:00. Al número de bicicletas disponibles esperadas que podemos encontrar en una estación e la denominamos como $E(B)$.

IV. Ese valor, $E(B) = B_0 + E\left(X_{j,0-7}\right)$, lo comparamos con el valor medio de salidas que se producen a las 7 para determinar si una estación tiene bicicletas de sobra, tiene las bicicletas justas o necesita bicicletas de alguna de estas estaciones que tienen de sobra. A una estación se le asigna que le faltan bicicletas cuando la suma, o resta según el signo del segundo término, del número de bicicletas que tiene a las 00:00 dicha estación y el valor obtenido de la resta entre la suma de las medias de entradas de bicicletas para ese día de la semana hasta las 07:00 y la suma de las medias de salidas de bicicletas para ese día de la semana hasta las 07:00 es menor que la medía de salidas de bicicletas para ese día de la semana a las 07:00.

$$E(B) < \lambda_{S,j,7}$$

Tendrá el número de bicicletas justas cuando este valor sea igual a la media de salidas a las 07:00 de ese día de la semana y tendrá de sobra cuando sea superior.

6. Soluciones propuestas para el balanceo de bicicletas

Para esta parte se ha trabajado con el paquete de software gratuito de Google, ORTools, como ya hemos mencionado anteriormente (3.5. Herramientas de optimización: OR Tools). Este paquete se especializa en la resolución de problemas de optimización, permitiendo resolver problemas de enrutado de vehículos, programación entera o lineal y problemas con restricciones de capacidad entre otros. Además, ofrece la posibilidad de trabajar en diferentes lenguajes informáticos como son C++, Python, Java o C#.

En el caso de los sistemas de compartición de bicicletas, y más en la parte que se corresponde al balanceo estático de estos recursos, se seleccionó la parte referida al problema de enrutado de vehículos, por la fuerte relación comentada anteriormente en la revisión del estado del arte. Dentro de esta clasificación, OR Tools ofrece una gran variedad de posibilidades, desde el *Travelling Salesman Problem (TSP)* o problema del viajero comercial hasta el *Vehicle Routing Problem (VRP)* o problema de enrutado de vehículos, para el que permite tener en cuenta una serie de restricciones en función de lo que se busque:

- *Capacity Constraints*: Hace referencia a imponer una restricción de capacidad en el vehículo o vehículos de transporte. En este caso, a cada nodo se le asigna un valor que hace referencia a una cantidad a dejar o recoger y el vehículo no podrá superar nunca un valor determinado por el usuario.
- *Pickups and Deliveries:* Consiste en establecer nodos de recogida y nodos de entrega. Se crea una lista que empareja un nodo a recoger con uno a entregar y de base devuelve la ruta más corta para pasar por todas, volviendo al nodo que se establezca como de origen.
- *Time Window Constraints:* Para los problemas que requieren un horario de visita para clientes que solo tienen disponibilidad en ciertos momentos del día. En este caso, en vez de utilizar matrices que incluyen las distancias entre los diferentes nodos, se introduce una matriz con los tiempos, también se añade una lista con las horas a las que se puede visitar cada nodo.
- *Resource Constraints:* Se ha hablado de restricciones de tiempo en los nodos/visitas pero también existe la posibilidad de añadir restricciones de tiempo en el propio depósito, simulando el tiempo que se tardaría en cargar y descargar el vehículo, y esta espera también la sufren los diferentes vehículos que se quieran incluir, ya que OR Tools permite introducir más de un vehículo.
- *Penalties and Dropping Visits:* Se trata de una opción para casos para los que no se encuentra una solución factible debido a las restricciones. Se pone de ejemplo un problema de enrutado de vehículos con restricciones de capacidad en el que la demanda total de las ubicaciones supera la capacidad del vehículo, por ello el vehículo tiene que dejar ciertos nodos/estaciones/clientes sin visitar. Esta versión añade un coste como penalización y el solver ofrece una opción que minimice la suma de las penalizaciones.

6.1. Pickups and Deliveries.

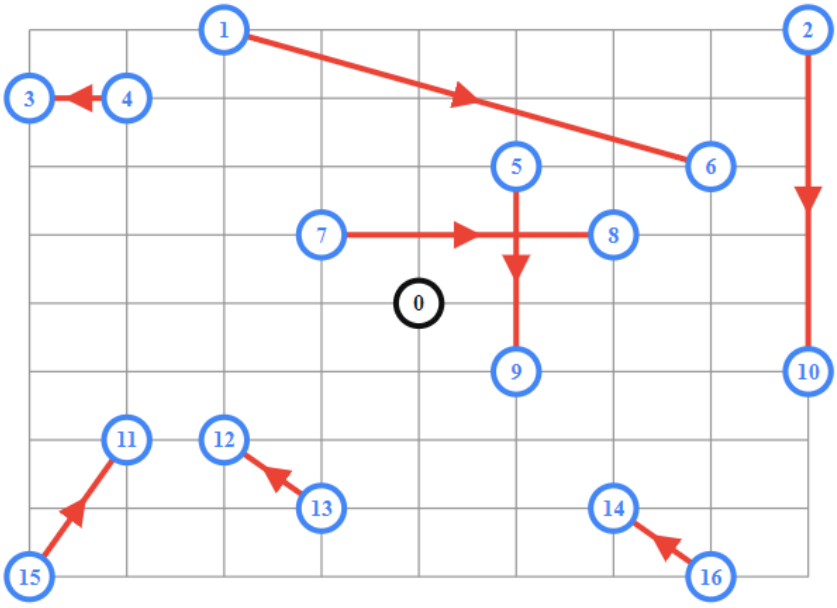


Figura 9, ejemplo representado gráficamente de un problema de Pickups and Deliveries en ORTools

En este trabajo se ha seleccionado la versión de Pickups and Deliveries que ofrece este paquete. El motivo se basa en que se va a partir de un depósito, en el que se encuentran almacenados los vehículos pertenecientes a JCDecaux y Valenbisi, y de allí tendrán que pasar por las estaciones que tienen bicicletas de sobra para llevarlas a las estaciones que se han establecido que requieren. Esto nos permite establecer que estaciones aportan un flujo positivo de bicicletas (se recogen) y qué estaciones aportan un flujo negativo (se dejan), que no sería posible en otras configuraciones que se ofrecen.

En cuanto a las restricciones, OR Tools permite plantear casos tanto simétricos como asimétricos en lo referido a las distancias, con un vehículo o más y también definir la capacidad de dichos vehículos, siendo posible establecer capacidades diferentes. También permite determinar un límite de distancia para la solución, pudiéndose establecer un objetivo a no superar.

En la implementación que hace ORTools del VRP con Pickups and Deliveries, se establecen una serie de asunciones para la entrada del problema. Todo problema que sea resuelto con esta herramienta deberá cumplir con estas asunciones. Por ello, para poder resolver nuestro problema del balanceo de la carga estático en sistemas de bicicletas

compartidas serán necesarios una serie de cambios en la entrada de nuestro problema. Más concretamente:

- Primero que nada, ORTools establece que cada estación solo puede ser asignada una vez. Es decir, en la lista en la que se hace referencia a las parejas de recogidas y entregas no puede aparecer repetido ningún valor. En la Figura 9 queda reflejado las propiedades iniciales que tiene el algoritmo en ORTools. Se muestra que cada nodo del grafo solo tiene asignado otro nodo. Además, los nodos no se repiten. En la configuración del método, el depósito es siempre el nodo 0, al cual regresa siempre al terminar la ruta. En nuestro caso, desde una estación con un número de bicicletas sobrantes podemos enviar a más de una estación a la que le faltan bicicletas. Por ello, en la elaboración de este trabajo se han creado nodos fantasma (con la mismas características que los nodos a los que representan y a distancia cero del nodo original al que representan) siempre que se ha requerido que un nodo sea visitado más de una vez. Esta es la única manera que permite que de una estación se puedan llevar bicicletas a más de una estación, que es lo que sucede en la gran mayoría de los casos. Esto también permite que el número de estaciones a visitar sea menor.
  En la Figura 10 se muestra como resuelve el algoritmo básico de OR Tools el ejemplo mostrado en la Figura 9, en el que se describe el planteamiento inicial del problema, reflejando las restricciones descritas en cuanto a las visitas de las estaciones para un caso con cuatro vehículos.

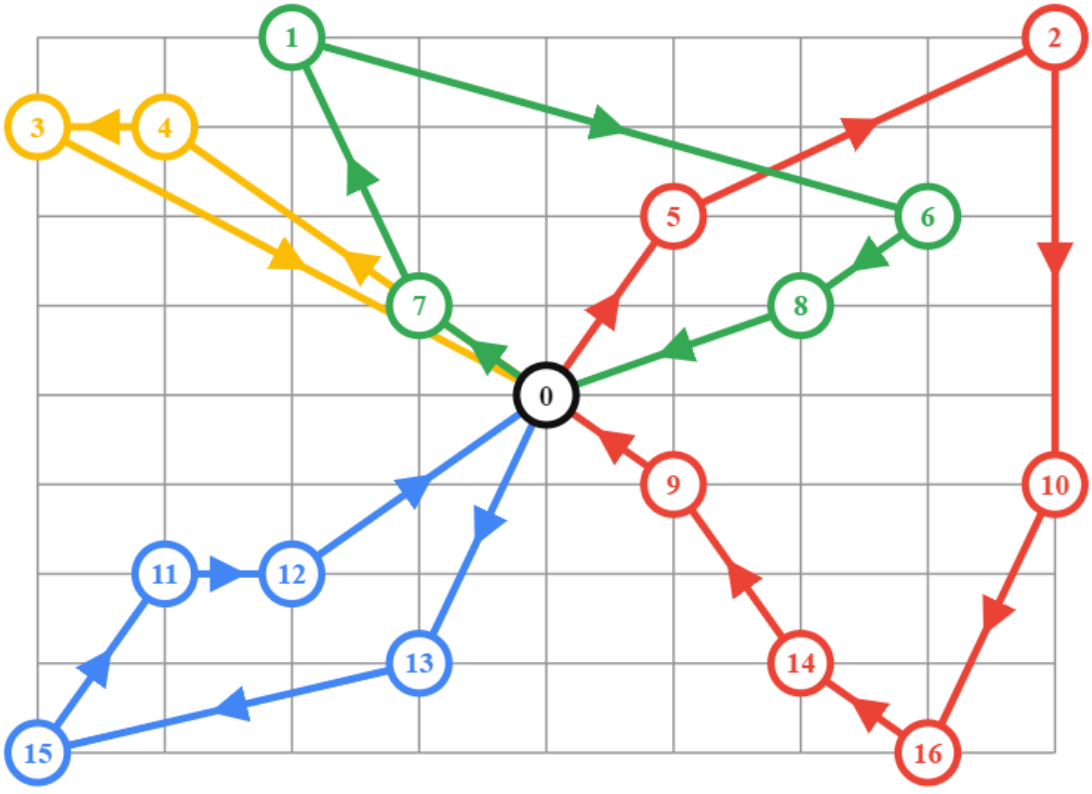


Figura 10, solución ejemplo básico de OR Tools.

- Por defecto, en la implementación de Pickup and Deliveries de ORTools se asume que en los nodos origen únicamente se recoge un ítem, que es llevado al nodo destino. En nuestro caso, en una estación se pueden recoger realmente varias bicicletas, al igual que se pueden dejar varias bicicletas. Para solventar este problema, se realizó una modificación del código de ejemplo proporcionado por ORTools para Pickup and Deliveries.
- Por otro lado, ORTools únicamente acepta como entrada la distancia entre los nodos que forman parte de alguna de las parejas de origen-destino, además del nodo 0 u origen global del problema. Esto conlleva que, a pesar de ser un sistema de 276 estaciones, para preparar la entrada para ORTools, se tienen que eliminar todas aquellas distancias que no participen en la ruta. Es decir, eliminar aquellas que tienen las bicicletas justas o que tienen de más pero no lo suficiente como para ser estaciones que entreguen a otras. Esto conlleva una nueva numeración de las estaciones en funciones de aquellas que aparezcan en el caso, como se muestra en la Figura 11. En este caso, la estación que se encuentre en la fila uno, representará la estación cero, la que se encuentre en la fila dos representará a la estación uno y así sucesivamente.

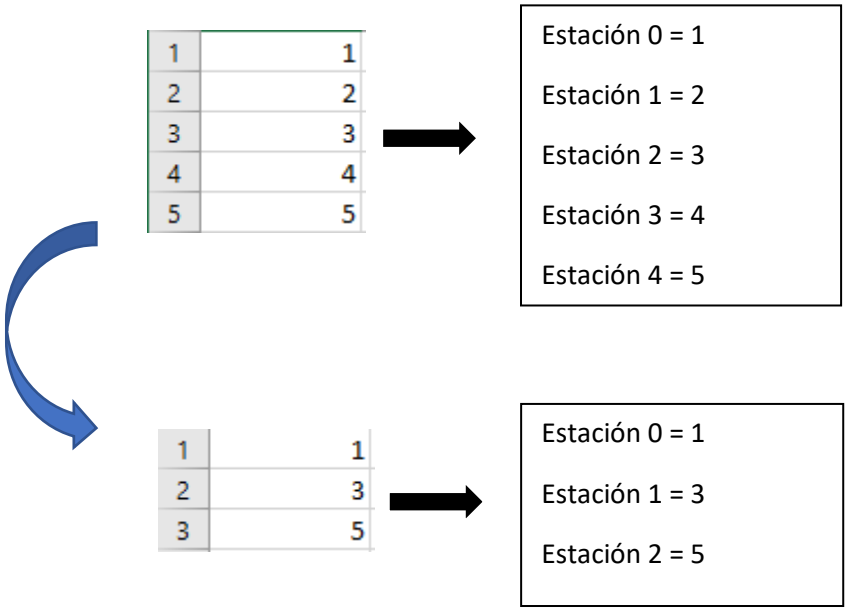
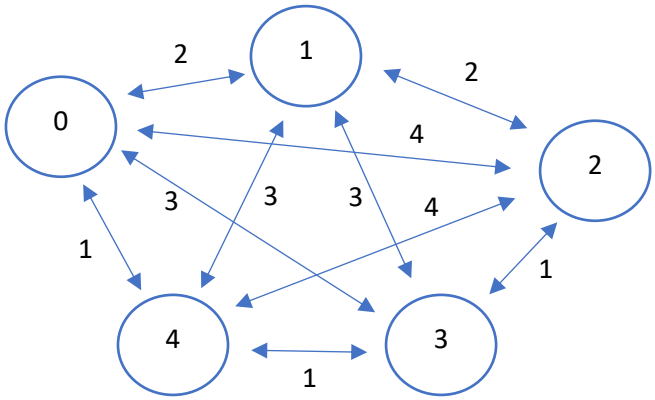


Figura 11, ejemplo para explicación de la nueva numeración

A continuación, se va a ilustrar estas modificaciones en la estructura inicial del problema con un pequeño ejemplo. Se plantea un caso con cinco nodos, con sus respectivas distancias entre ellos como se muestra en las Figuras 12 (a), (b) y (c), en el que

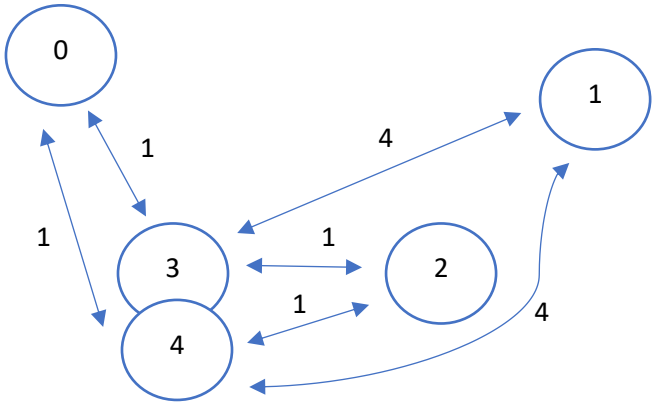
el nodo 2 (que tendrá el valor de 1 en el algoritmo) no participa en la ruta y en el que el nodo 5 (asignado al valor 4) tiene que visitar tanto al nodo 3 como al nodo 4.

1.



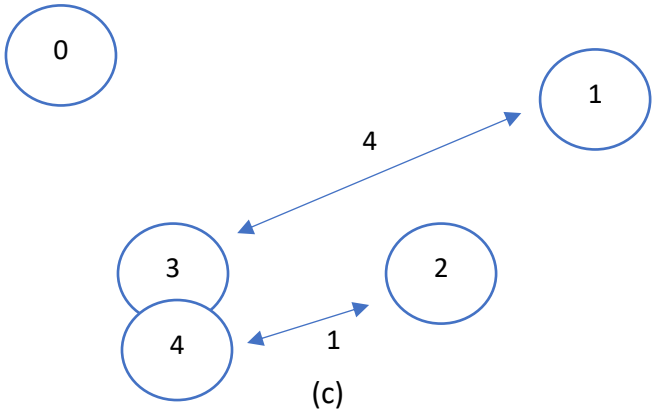
(a)

2.



(b)

3.



(c)

Figura 12 ejemplo con pasos para preparar los datos y asignaciones: (a) Red inicial con distancias; (b) Red con nodos fantasma; (c) Selección de pares origen-destino.

Como habrá podido observar el lector, ORTools es capaz de encontrar rutas de distancia mínima (o cercana a la mínima) con las heurísticas implementadas internamente. No obstante, requiere que los datos sean preparados apropiadamente y, lo más importante, que se decidan los pares de origen-destino apropiados para los Pickups and Deliveries. Por tanto, la elección de estos pares es clave para la calidad de la solución encontrada por ORTools, ya que influyen directamente sobre la ruta que se realizará. Debido a esto, en este TFG hemos propuesto varios métodos de asignación de pares origen-destino con el fin de resolver el problema con ORTools y estudiar el desempeño de cada uno de los métodos de asignación. Estos algoritmos de asignación de origen-destino se describirán en detalle en la Sección 5.2 y 5.3.

Antes de describir con los métodos de asignación, se debe determinar qué estaciones de entre todas las que tienen bicicletas de sobra se van a recoger las bicicletas. El criterio escogido fue:

- Seleccionar las estaciones que tuvieran el mayor número total de bicicletas de sobra. Estableciendo un mínimo de entre nueve y diez bicicletas de sobra con el objetivo de ser capaces de abastecer a las estaciones que necesitan y sin que fuese necesario llevarse todas las bicicletas sobrantes de estas primeras estaciones.
- Buscando que el número de estaciones a visitar para extraer las bicicletas fuera la mínima posible, para ello se sumó el total de bicicletas que se requería para cada día y se buscó un número total de estaciones de las cuales se pudiera extraer esa cantidad. Es decir, si para un día eran necesarias 150 bicicletas, las estaciones seleccionadas debían ser capaces de aportarlas mientras se cumple el primer punto.

Este proceso de pre-filtrado de las estaciones para la recogida de bicicletas se realizó manualmente para cada instancia generada. La Figura 13 muestra para cada uno de los siete días aquellas estaciones de las 276 estaciones que se filtraron como estaciones con bicicletas para recoger, estaciones con bicicletas para recibir y estaciones con un número de bicicletas justas. El color rojo indica que esa estación requiere ser visitada y que necesita el número de bicicletas que marca cada celda para estar preparada. El color verde indica que esa estación está preparada pero que no entrega ninguna bicicleta. Las de color azul son aquellas que tienen las bicicletas justas. Por último, las de color naranja son aquellas estaciones que tienen bicicletas de sobra, el valor muestra el número total de bicicletas que les sobra, no es el número de bicicletas que entregan.

Estación	15/12/2014	11/11/2014	01/07/2015	21/05/2015	27/03/2015	27/09/2014	01/02/2015
1	2	9	3	-1	-4	2	7
2	-4	2	0	2	-1	3	-1
3	-2	-2	-5	-1	2	7	-1
4	-3	-1	-2	0	-4	-1	-1
5	2	-3	-2	0	-2	0	7
6	4	-2	-3	2	2	5	2
7	-1	-1	-4	-4	-4	4	-1
8	-2	-2	-1	0	0	1	-1
9	12	0	0	0	6	-1	17
10	-1	-2	-4	-3	-4	1	2
11	1	6	3	7	3	-1	-1
12	5	2	4	4	3	10	2
13	12	2	2	2	9	13	1
14	2	6	3	4	2	10	-1
15	27	6	7	5	7	13	-1
16	17	2	1	2	1	1	10
17	22	3	4	1	0	18	13
18	3	0	0	2	-1	2	5
19	6	4	4	-4	0	1	0
20	13	-3	0	1	0	-1	-1

Figura 13, análisis de la situación que refleja el estado de todas las estaciones así como las bicicletas que necesita cada estación.

6.2. Método de asignación aleatorio

En esta sección se busca que el lector comprenda el proceso que se ha llevado a cabo a la hora de realizar los diferentes emparejamientos entre las estaciones para el primero de los dos métodos elaborados:

1. En primer lugar, se realiza los emparejamientos entre las estaciones, a partir del análisis realizado anteriormente se conoce que estaciones requieren bicicletas y de cuales pueden venir. Para ello, se ordenan de forma aleatoria las estaciones que requieren bicicletas, mientras que las estaciones que tienen bicicletas de sobra mantienen su orden numérico (entre las estaciones seleccionadas).
2. Con ese nuevo orden, se empiezan a asignar (se crea par origen destino) de una en una a la primera estación que tenga bicicletas de más en función del orden numérico que se ha mantenido en su caso.
3. Se empiezan a asignar a esa estación que tiene bicicletas de sobra hasta alcanzar las diez definidas como capacidad del vehículo, o en el caso de que la siguiente estación a incluir hiciese superar esas diez bicicletas, esa estación pasaría a ser asignada a la siguiente estación con bicicletas de sobra.

ESTACIÓN 66(23)	93(33)	101(29)	160(20)
1. 214(1)	1. 23(1)	1. 24(2)	1. 78(2)
2. 4(1)	2. 141(3)	2. 9(1)	2. 60(2)
3. 57(2)	3. 33(1)	3. 26(1)	3. 82(1)
4. 20(1)	4. 30(1)	4. 36(2)	4. 249(1)
5. 94(2)	5. 27(1)	5. 156(1)	
6. 243(1)	6. 92(1)	6. 25(2)	
7. 247(2)	7. 116(1)		
	8. 11(1)		

Figura 14, ejemplo asignación aleatoria para uno de los casos

Una vez han quedado asignadas las estaciones, se lleva a cabo la preparación de la matriz de distancias de la forma que se ha explicado anteriormente, eliminando de las 276 aquellas que no participen y replicando las estaciones de recogida que se repitan, en el ejemplo mostrado en la figura 14 como se puede observar se tendrían que crear 6 nodos fantasma para la estación 66, 7 nodos fantasma para la estación 93, 5 nodos fantasma para la estación 101 y 3 nodos fantasma para la estación 160. El número de nodos fantasmas será el número de estaciones asignadas menos uno, ya que se mantiene la original, es decir, consiste en "crear" nuevas estaciones a partir de una ya existente. Para ello, a través de la herramienta Excel para agilizar el proceso, se lleva a cabo el duplicado de las filas y columnas, así como la eliminación de las que no participan. Este proceso a su vez genera la nueva numeración como se refleja en la Figura 11.

A continuación, toca preparar la entrada en Python para ORTools. En la lista para ello se ha denominada como "Flow" se anota el flujo de bicicletas en cada nodo (incluidos fantasma). Es decir, un valor es positivo significa que de esa estación se recogen bicicletas, y su cantidad, y si es negativo indica que se dejan, y su cantidad. En la Figura 15 se muestra para el pequeño ejemplo desarrollado antes.

```
data['flow'] = {
    0:0,
    1:2,
    2:1,
    3:-2,
    4:-1,

}
```

Figura 15, lista "Flow" para el ejemplo desarrollado

Se le asigna cero al valor que represente el depósito, en el ejemplo sería el 1 (con valor 0) pero en nuestro caso el depósito ha sido asignado a la estación 268.

Por último, se introduce en ORTools la nueva matriz de distancias, con los emparejamientos entre estaciones y el flujo de bicicletas establecido. Una vez ejecutado

ORTools con la entrada apropiada, nos devolverá la ruta que ha seguido y la distancia recorrida. Para el pequeño ejemplo creado (6.1. Pickups and Deliveries.) nos devuelve lo mostrado en la Figura 16.

```
Objective: 3838
Route for vehicle 0:
 0 ->  2 ->  1 ->  3 ->  4 -> 0
Distance of the route: 38m

Total Distance of all routes: 38m
```

Figura 16, solución devuelta por OR Tools para el ejemplo.

6.3. Método de asignación por mínima distancia posible

Al igual que para el método aleatorio, en este punto se busca desarrollar el proceso que se ha seguido para la creación de los diferentes emparejamientos entre estaciones.

El primer paso, el de emparejamiento de estaciones, en este método se realiza de forma manual a partir de una vista de la situación en Google Earth, como es la Figura 17 en la que las etiquetas hacen referencia al identificador de la estación.

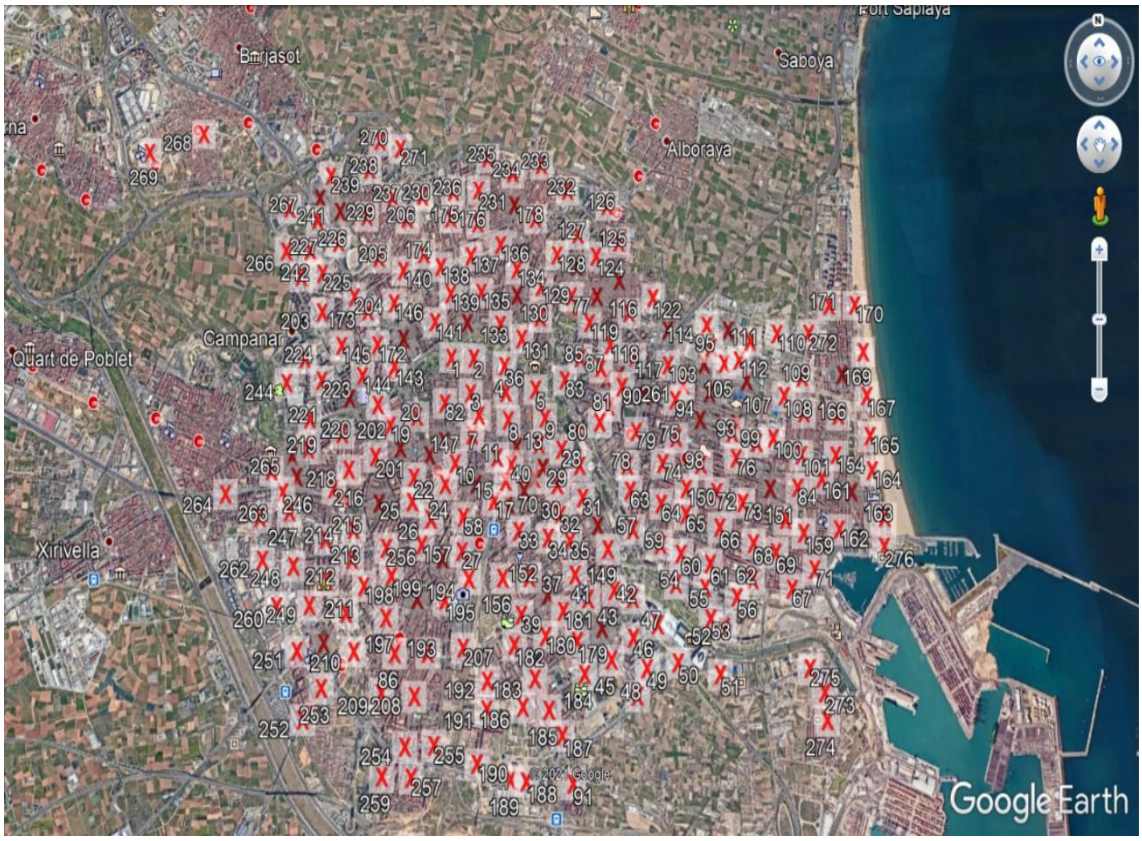


Figura 17, plano de las estaciones de Valenbisi

En este caso, hay que tener en cuenta que la asignación no se ha podido hacer de forma directa, es decir, una estación con bicicletas de más no tiene por qué tener asignada la estación que requiera bicicletas inmediatamente más cercanas y esto se debe a que habría conflicto si para dos estaciones la más próxima es la misma, y a que diversas estaciones, las más alejadas del centro, sobre todo, se verían emparejadas con las estaciones que "sobraran" pudiéndose dar emparejamientos poco prácticos.

1. Se ha creado grupos de estaciones que requieren ser visitadas y que se encuentran próximas entre ellas (entre las que requieren ser visitadas), estableciendo una distancia máxima entre cualquier estación y al menos una de las estaciones que formen dicho grupo de un kilómetro (< 1 km), buscando que sus demandas sumen diez, aunque en algunos casos se admiten grupos menores de diez para situaciones en los que no haya estaciones próximas o se requiera ir a distancias muy grandes para poder formar los grupos.
2. Se recuadran las estaciones que van a entregar bicis para obtener toda la información necesaria para poder llevar a cabo la asignación. Esto queda reflejado en la Figura 18, donde los grupos de estaciones se marcan con un color mientras que las estaciones con bicicletas de más quedan recuadradas.
3. Una vez clasificados se comienzan a emparejar de forma manual basándose en un criterio de mínima distancia entre la estación de carga y el grupo de estaciones formado, buscando evitar parejas muy descompensadas en cuanto a distancia. Se podría empezar a emparejar por la estación que el usuario desee o por un grupo de bicicletas que requieran visita determinado.

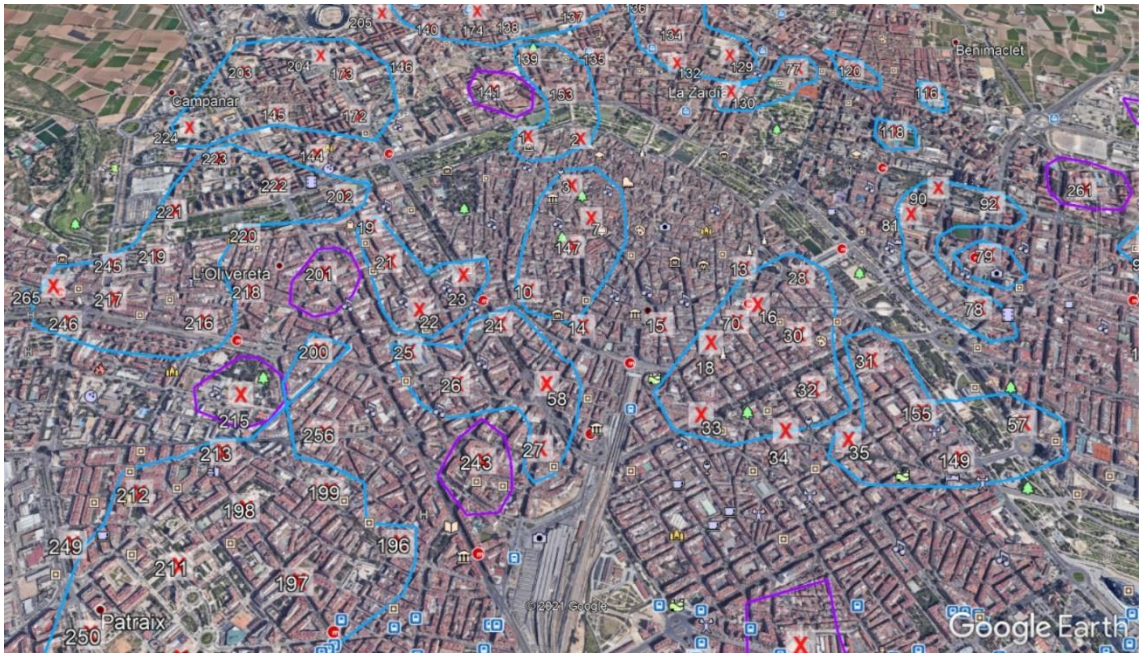


Figura 18, ejemplo de grupos de estaciones.

El resto del proceso para este método sería igual al método aleatorio, tocaría preparar las matrices de distancias, con sus nodos fantasma, y el flujo de bicicletas para posteriormente introducirlo en OR Tools. Para el caso de la Figura 18, la asignación mediante el método de la mínima distancia quedaría como muestra la Figura 19, en este caso no se ha indicado entre paréntesis el valor de las bicicletas entre paréntesis debido a que como están agrupadas de tal manera que suman diez o menos no se ha considerado oportuno.

ESTACIÓN 66(23)	93(33)	101(29)	160(20)
25	4	9	24
26	11	92	30
27	20	94	33
156	23	116	57
214	36		60
243	82		78
247	141		
249			

Figura 19, ejemplo asignación mínima distancia

7. Implementación del VRP con Pickup and Deliveries en Python:

7.1. Descripción del código para el VRP:

El primer paso en la descripción del código es el de la creación de los datos. En nuestro caso se trata de la matriz de distancias, los emparejamientos entre los nodos de recogida y los nodos de entrega, siendo el primer valor el identificador de la estación de recogida y el segundo el identificador de la estación de entrega y el flujo de bicicletas que se dejan o recogen en cada estación.

A continuación, se añade la función que nos devuelve la solución, donde se indica que devuelva la ruta y su distancia, además de la suma de las rutas en los casos con más de un vehículo, como se observa en la Figura 20.

```
def print_solution(data, manager, routing, solution):
    """Prints solution on console."""
    print(f'Objective: {solution.ObjectiveValue()}')
    max_route_distance = 0
    for vehicle_id in range(data['num_vehicles']):
        index = routing.Start(vehicle_id)
        plan_output = 'Route for vehicle {}:\n'.format(vehicle_id)
        route_distance = 0
        while not routing.IsEnd(index):
            plan_output += ' {} -> '.format(manager.IndexToNode(index))
            previous_index = index
            index = solution.Value(routing.NextVar(index))
            route_distance += routing.GetArcCostForVehicle(
                previous_index, index, vehicle_id)
        plan_output += '{}\n'.format(manager.IndexToNode(index))
        plan_output += 'Distance of the route: {}m\n'.format(route_distance)
        print(plan_output)
        max_route_distance = max(route_distance, max_route_distance)
    print('Maximum of the route distances: {}m'.format(max_route_distance))
```

Figura 20, función que devuelve la solución

Una vez hecho esto, el siguiente paso es el de la creación del modelo de enrutado dentro de la función principal, en el que se lee el número de filas de la matriz de las distancias, el número de vehículo y el nodo que se establece como depósito. En la Figura 21 lo vemos reflejado en la parte referida a “manager”.

```
data = create_data_model()
manager = pywrapcp.RoutingIndexManager(len(data['distance_matrix']),
                                       data['num_vehicles'], data['depot'])
routing = pywrapcp.RoutingModel(manager)
```

Figura 21, creación del modelo de enrutado

La siguiente parte del código hace referencia a la función que devuelve la distancia entre un par de ubicaciones, también nos determina el coste del arco, del trayecto, que en este caso se trata de la distancia (Figura 22).

```
def distance_callback(from_index, to_index):
    """Returns the distance between the two nodes."""
    # Convert from routing variable Index to distance matrix NodeIndex.
    from_node = manager.IndexToNode(from_index)
    to_node = manager.IndexToNode(to_index)
    return data['distance_matrix'][from_node][to_node]

transit_callback_index = routing.RegisterTransitCallback(distance_callback)
routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index)
```

Figura 22, función distancia

Para resolver los VRP es necesario crear una dimensión para la distancia, que irá acumulando la distancia recorrida por el vehículo a lo largo de la ruta. Como observamos en la Figura 23, se puede establecer una distancia máxima.

```
dimension_name = 'Distance'
routing.AddDimension(
    transit_callback_index,
    0,  # no slack
    3000,  # vehicle maximum travel distance
    True,  # start cumul to zero
    dimension_name)
distance_dimension = routing.GetDimensionOrDie(dimension_name)
distance_dimension.SetGlobalSpanCostCoefficient(100)
```

Figura 23, creación de la dimensión distancia.

7.2. Descripción de la parte de Pickups y Deliveries:

Seguidamente, se definen las órdenes de recogida y de entrega utilizando las localizaciones en *data['pickups_delivieries']* (Figura 24). También se añaden restricciones para que el vehículo que haga la recogida y la entrega para un par de origen-destino sea el mismo. Además, se añade también una restricción para que la entrega se realice después de la recogida.

```
for request in data['pickups_deliveries']:
    pickup_index = manager.NodeToIndex(request[0])
    delivery_index = manager.NodeToIndex(request[1])
    routing.AddPickupAndDelivery(pickup_index, delivery_index)
    routing.solver().Add(
        routing.VehicleVar(pickup_index) == routing.VehicleVar(
            delivery_index))
    routing.solver().Add(
        distance_dimension.CumulVar(pickup_index) <=
        distance_dimension.CumulVar(delivery_index))
```

Figura 24, creación de las órdenes de recogida y entrega

8. Experimentos y análisis de resultados.

En lo referido a la parte experimental, el objetivo que se ha marcado ha sido el de obtener el algoritmo de asignación que garantice la menor distancia recorrida por los vehículos encargados de balancear el sistema. Para ello se va a analizar, a través de los dos métodos explicados, las distancias recorridas en las rutas proporcionadas por los algoritmos propuestos, así como el tiempo de respuesta que se requiere obtener la solución.

Las distancias que se han evaluado para determinar el algoritmo que garantiza nuestro objetivo provienen de la creación de una serie de casos para los siete días seleccionados (5.2. Selección de casos de estudio:). Para cada día/instancia/problema se han llevado a cabo 20 simulaciones con cada método, en el caso del método aleatorio las 20 simulaciones, y se ha llevado a cabo una comparación entre los resultados obtenidos por ambos métodos tanto a nivel global como a nivel de cada instancia/problema/día. El análisis está apoyado por técnicas de contraste de hipótesis.

Para dichas simulaciones, se ha considerado un único vehículo cuya capacidad era de 10 bicicletas y el depósito, debido a que la ubicación del depósito oficial de la compañía no es pública, se ha establecido en una estación que en ninguno de los siete casos requería visita y que además se encontraba en las afueras de la ciudad, tratando de asemejarlo lo máximo posible a la realidad. Esta decisión se ha tomado ya que se ha considerado que un depósito de estas características no se debe de encontrar en una zona céntrica, en este caso la estación seleccionada está asignada al número 268 de las 276 que componen el sistema. Además, no se ha establecido ningún límite de distancia a los vehículos, lo que ha dado lugar a casos con largas distancias.

8.1. Análisis del desempeño global.

Primero que nada, analizamos el desempeño de ambos algoritmos de asignación de pares de origen-destino a nivel global. Es decir, teniendo en cuenta todas las simulaciones llevadas a cabo e independientemente de a qué instancia pertenezcan.

Para el análisis global de las soluciones se ha realizado una prueba de significancia a los resultados obtenido en las 140 simulaciones (7 días x 20 repeticiones=140 simulaciones) realizadas para cada algoritmo de asignación, es decir, en nuestro caso la variable aleatoria ha sido la distancia recorrida por los vehículos en la solución propuesta por cada uno de los algoritmos.

El primer paso ha sido realizar una comprobación de la normalidad de los datos, para ello se ha realizado la prueba K-cuadrado de D'Agostino, prueba que tiene como objetivo determinar si una muestra dada proviene o no de una población distribuida normalmente, basándose en transformaciones de la curtosis y la asimetría de la muestra. Aplicándole dicha prueba a nuestras muestras se obtiene que no siguen una distribución normal, por ello, para valorar posibles diferencias entre ambas muestras se tendrá que recurrir a la prueba "U" de Mann-Whitney, un test de datos no paramétricos que nos permite determinar las diferencias de dos grupos independientes. En nuestro caso la hipótesis nula H_0 será que las medianas de las distancias recorridas por cada uno de los métodos son iguales, y la hipótesis alternativa será que ambas medianas son diferentes.

Para la prueba "U" de Mann-Whitney se han establecido los siguientes parámetros:

- Nº de colas: Dos.
- Nivel de significancia (α) = 0.05

Con estos parámetros y nuestras muestras obtenemos los siguientes resultados:

- P-value= 4.219e-15
- Z= -7.8475, que se encuentra fuera de la región de aceptabilidad del 95% [-1.96:1,96].
- U=4490, que no se encuentra dentro de la región de aceptabilidad del 95% [8473.9105:1126.0895]

Siendo: $U_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1$

$U_2 = n_1 n_2 + \frac{n_2(n_2+1)}{2} - R_2$

$U = mín(U_1, U_2)$

$Z = \frac{U-m}{\sigma}$

Siendo R la suma de los rangos, es decir la suma de la posición relativa de cada individuo en la muestra, y n los tamaños respectivos de las muestras. Para Z, m es la media y σ es la desviación estándar.

Estos valores nos indican que H_0 se rechaza (p-value<α) y que la mediana de las distancias recorridas por ambos métodos es estadísticamente diferente. Lo podemos observar en la Figura 25, que refleja las medianas de las simulaciones resueltas por cada método, quedando a la vista como es el método de asignación de mínima distancia el que ofrece mejor resultado.

Mediana	
Mínima	Aleatorio
168985 m	252780 m

Figura 25, medianas de los dos métodos utilizados

Se observan una mejora del 33.15% en la distancia media entre ambos métodos.

8.2. Análisis del desempeño a nivel de instancias:

Al igual que en el caso global, para cada una de las siete instancias se ha llevado a cabo una prueba de significancia en las muestras obtenidas para cada una de las instancias. Es decir, buscamos determinar si existen diferencias significativas en el desempeño de cada algoritmo de asignación a nivel de cada instancia (recordar que hemos creado 7 instancias diferentes). Para ello, primero se ha determinado que ninguna de las muestras recogidas para cada una de las instancias y algoritmos no siguen una distribución normal, con la misma prueba K-cuadrado de D'Agostino. Por ello, de nuevo, se ha pasado a realizar una prueba "U" de Mann-Whitney para determinar si existen diferencias significativas entre las medianas las distancias recorridas por los vehículos. Para las siete instancias se ha obtenido p-valores inferiores a α, por lo que existen evidencias para pensar que las medianas obtenidas por cada método a nivel de instancia son estadísticamente diferentes.

Después de los resultados arrojados por las pruebas de Mann-Whitney anteriores, se procede a realizar una comparativa entre los distintos resultados. A continuación, en la Figura 26, se muestra una comparativa entre las medianas de los siete casos estudiados para ambos métodos:

27/09/2014		11/11/2014		15/12/2014	
Mínima	Aleatorio	Mín	Al	Mín	Al
65533	71812.5	176813	257914.5	232224	327232.5

01/02/2015		27/03/2015		21/05/2015	
Mín	Al	Mín	Al	Mín	Al
61723	68915	168985	217483.5	161741	260753

01/07/2015	
Mín	Al
183940	285342

Figura 26, comparativa de las medianas para los casos estudiados

Se puede observar una diferencia considerable, produciéndose reducciones en las distancias de hasta el 35.53%, en el caso del 01/07/2015, en las soluciones excepto en los casos del 27/09/2014 y el 01/02/2015, que se producen caídas de 8.74% y de 10.45% respectivamente, esto coincide con los casos que menor número de bicicletas requerían visita, reduciendo así las posibilidades en las combinaciones.

También es destacable que se tratan de distancias demasiado largas, alcanzo en algunos casos más de 200 kilómetros, esto hace pensar que sería interesante aumentar el número de vehículos o la capacidad de estos, que ahorraría el número de estaciones a visitar y el número de visitas a las estaciones que tienen bicicletas de más. También podría ser interesante analizar el caso de no visitar ciertas estaciones y centrarse que se encontrasen en una situación más crítica.

De estas simulaciones también se ha realizado el análisis de los tiempos de ejecución por parte del programa, obteniendo los resultados reflejados en la Figura 27, que representan el tiempo medio que le ha requerido al algoritmo obtener la solución.:

27/09/2014		11/11/2014		15/12/2014	
Mínima	Aleatorio	Mín	Al	Mín	Al
< 1 min	< 30 s	12 min	10 min	48 min	18 min

01/02/2015		27/03/2015		21/05/2015	
Mín	Al	Mín	Al	Mín	Al
< 1 min	< 1 min	7 min	5 min	12 min	9 min

01/07/2015	
Mín	Al
11 min	10 min

Figura 27, tiempos medios de ejecución para cada caso.

Destaca que en los casos aleatorios se reducen los tiempos de simulación, de forma considerable en algunos casos como es el del 15/12/2014. El motivo de estas diferencias en el tiempo puede ser debido a que, al ser peores soluciones en el caso aleatorio, conlleva una menor repetición en la búsqueda posibles mejores soluciones.

Se tratan de tiempos moderados en relación con el proceso que se quiere llevar a cabo, que es el del balanceo que se realiza una vez al día.

Para la representación gráfica de las soluciones se ha utilizado Google Earth como método de visualización y la estrategia seguida para mostrar de la forma más clara posible las rutas ha sido a través de un etiquetado en el que se indica el orden en el que se visita cada estación y se diferencia entre las estaciones de las cuales se recogen bicicletas y en las cuales se entregan mediante el identificador de las estaciones, que en el caso de las estaciones en las cuales se recogen aparece entre paréntesis. Se ha seleccionado este método debido a que a la hora de trazar líneas que conectasen las diferentes estaciones no quedaba claro debido a que el vehículo es capaz de visitar una misma estación de recogida varias veces, produciendo muchos cruces entre todas las líneas.

El orden de aparición es en primer lugar el caso de "mínima distancia" (Figura 28) y después el caso aleatorio (Figura 30). Debajo de cada plano aparece las asignaciones en formato de tabla por si no quedase claro con los planos (Figuras 29 y 31 respectivamente).

- 27/09/2014:

Figura 28, ruta de la solución para el 27/09/2014 (mín distancia)

ESTACIÓN 66(23)	93(33)	101(29)	160(20)
25	4	9	24
26	11	92	30
27	20	94	33
156	23	116	57
214	36		60
243	82		78
247	141		
249			

Figura 29, asignaciones solución mínima distancia para el 27/09/2014

Figura 30, ruta de la solución para el 27/09/2014 (aleatoria)

ESTACIÓN 66(23)	93(33)	101(29)	160(20)
1. 214(1)	1. 23(1)	1. 24(2)	1. 78(2)
2. 4(1)	2. 141(3)	2. 9(1)	2. 60(2)
3. 57(2)	3. 33(1)	3. 26(1)	3. 82(1)
4. 20(1)	4. 30(1)	4. 36(2)	4. 249(1)
5. 94(2)	5. 27(1)	5. 156(1)	
6. 243(1)	6. 92(1)	6. 25(2)	
7. 247(2)	7. 116(1)		
	8. 11(1)		

Figura 31, asignaciones solución aleatoria para el 27/09/2014

En las simulaciones se observa que el vehículo hace movimientos de todo tipo, desde cargarse en una única estación las diez bicicletas, en el caso del 21/05/2015 del depósito va directamente la estación 141 y recoge todas las bicicletas que tiene para entregar, a pasar por varias estaciones de carga diferentes de forma consecutiva, como sucede en la solución del caso aleatorio del 01/02/2015, visita de forma consecutiva las estaciones 95-93-112 (en ese orden), e incluso en algunos cargarse de una estación al principio de la ruta y no volver a pasar hasta prácticamente el final por lo que no es posible sacar un patrón. Por lo general se observa un orden coherente de las visitas, cargándose con bicicletas para estaciones que están próximas entre sí y siempre el trayecto de regreso al depósito es aprovechado para visitar las estaciones que están más próximas a este. Se acondiciona de forma correcta a los diferentes emparejamientos que se hayan establecido.

9. Conclusiones:

En este trabajo se ha querido explorar el problema del enrutado de vehículo para un caso muy específico como lo es un sistema de compartición de bicicletas, se tratan de sistemas en los que los usuarios tienen acceso a unas bicicletas en estaciones repartidas por toda la ciudad a un precio determinado. Uno de los principales problemas es que debido al mismo uso del sistema se producen acumulaciones o faltas de bicicletas en ciertas estaciones a determinadas horas del día, por ello se ha buscado analizar su uso y plantear una mejora de este sistema en lo referido al balanceo de sus recursos (las bicicletas) a través de un algoritmo de optimización y una serie de simulaciones en este a partir de dos métodos de asignación entre las estaciones, uno aleatorio y uno de mínima distancia. Este primero se centra en una creación completamente aleatorio de emparejamiento de estaciones, sin importar la ubicación de los mismos, la única restricción que presenta es la de no poder superar en las estaciones de carga el total de diez bicicletas entre los diferentes emparejamientos, mientras que en el método aleatorio se han buscado de forma individualizada los emparejamientos para evitar casos muy poco prácticos.

Los resultados muestran con claridad que las simulaciones con los casos asignados a una hipotética mínima distancia ofrecen mejores resultados, confirmándose esa hipótesis y ofreciendo una posibilidad de mejora para la actualidad. Se puede observar que ORTools es una herramienta práctica y que permite al usuario una gran cantidad de posibilidades a la hora de optimización de procesos. En el caso del rebalanceo en los sistemas de compartición de bicicletas y más concretamente en el caso de Valenbisi, a pesar de no ser pública información acerca del funcionamiento de este proceso por parte de JCDecaux, puede ofrecer una gran ayuda a la hora de reducir tiempos, distancias y a la vez costes en estos trayectos.

En futuros trabajos se podrían realizar más comparaciones con otros casos, además del aleatorio, que permitieran constatar al caso de la mínima distancia como mejor solución o, si se diese el caso, encontrar una alternativa mejor. Quizá un mayor número de casos con los que compararlo, como puede ser una asignación por orden numérico, o se podrían modificar las restricciones, como un mayor número de vehículo o de la capacidad de estos, que permitiese no tener que visitar tantas estaciones de carga.

10. Presupuesto.

En lo referido al presupuesto, se ha valorado tanto la parte de implementación del software como de la parte del balanceo para las características que se han establecido en las simulaciones, es decir, un único vehículo.

En la parte referida al personal, debido a que obtenemos soluciones con grandes distancias, sería necesario tener a dos operarios para poder rotar entre ellos, ya que puede ser un proceso que lleve varias horas.

Se contratará a una empresa externa, en este caso una consultoría, para el desarrollo del software, para el coste se ha hecho una estimación por horas aplicando precios horarios[10].

Al tener en cuenta el proceso de balanceo se ha considerado incluir los costes del vehículo, el precio se ha obtenido de un artículo que hace referencias a las mejores furgonetas eléctricas en relación calidad-precio[11]. Esto también hace tener en cuenta el gasto de electricidad del vehículo, cuyo precio de carga se ha obtenido de una página web de movilidad eléctrica[12] y el número de kilómetros indicados es la media diaria según los resultados de las simulaciones.

PARTIDA	Subpartida	Total	%
Amortización activos fijos (5 años)		**6,000 €**	5.20%
Vehículo transporte bicicletas, incluyendo remolque (PVP 30.000€)	6,000 €		
Costes de personal.		**87,900 €**	76.18%
1 operario encargado de la aplicación del software	37,900 €		
2 operarios con carnet de conducir	50,000 €		
Suministros (Electricidad carga furgoneta 150km/día - 2€/100km)		**1,095 €**	0.95%
Servicios externos.		**11,600 €**	10.05%
Mantenimiento vehículo	2,500 €		
Seguro vehiculo	700 €		
Consultora encargada del desarrollo del software (35€/hora, 8h/día, 30 días)	8,400 €		
Gastos generales e imprevistos		**8,790 €**	7.62%

TOTAL PROYECTO	115,385 €

[10] https://www.masquenegocio.com/2017/07/19/desarrollar-app-espana/#:~:text=As%C3%AD%2C%20el%20precio%20medio%20para,aproximadamente%2035%20euros%20por%20hora.

[11] https://www.autobild.es/noticias/furgonetas-electricas-3-mejores-relacion-calidad-precio-191288

[12] https://movilidadelectrica.com/cargar-coche-electrico/

11. Bibliografía.

- Applegate, D. L., Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2006). *The traveling salesman problem*. Princeton: Princeton University Press.
- Caceres-Cruz, J., Arias, P., Guimarans, D., Riera, D., & Juan, A. A. (2015). Rich vehicle routing problem. *ACM Computing Surveys, 47*(2), 1-28. doi:10.1145/2666003
- Chemla, D., Meunier, F., & Wolfler Calvo, R. (2013). Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization, 10*(2), 120-146. doi:10.1016/j.disopt.2012.11.005
- Cruz, F., Subramanian, A., Bruck, B. P., & Iori, M. (2017). A heuristic algorithm for a single vehicle static bike sharing rebalancing problem. *Computers & Operations Research, 79*, 19-33. doi:10.1016/j.cor.2016.09.025
- Daza, J. M., Montoya, J. R., & Narducci, F. (2009). Resolución del problema de enrutamiento de vehículos con limitaciones de capacidad utilizando un procedimiento metaheurístico de dos fases. *Revista EIA,* (12), 23-38. Retrieved from http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1794-12372009000200003&lng=en&tlng=en
- Dell'Amico, M., Hadjicostantinou, E., Iori, M., & Novellani, S. (2014). The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega (Oxford), 45*, 7-19. doi:10.1016/j.omega.2013.12.001
- Espegren, H. M., Kristianslund, J., Andersson, H., & Fagerholt, K. (2016). The static bicycle repositioning problem - literature survey and new formulation. *Computational logistics* (pp. 337-351). Cham: Springer International Publishing. doi:10.1007/978-3-319-44896-1_22

- García-Palomares, J. C., Gutiérrez, J., & Latorre, M. (2012). Optimizing the location of stations in bike-sharing programs: A GIS approach. *Applied Geography (Sevenoaks), 35*(1-2), 235-246. doi:10.1016/j.apgeog.2012.07.002
- Garey, M. R., & Johnson, D. S. (2008). *Computers and intractability* (27. print. ed.). New York: Freeman.
- Goel, A., & Gruhn, V. (2005). Solving a dynamic real-life vehicle routing problem. *Operations research proceedings 2005* (pp. 367-372). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/3-540-32539-5_58
- *Https://Developers.google.com/optimization/routing/cvrp*.https://developers.google.com/optimization/routing/cvrp
- *Https://Neos-guide.org/content/optimization-taxonomy*.https://neos-guide.org/content/optimization-taxonomy
- *Https://Www.statlect.com/fundamentals-of-statistics/poisson-distribution-maximum-likelihood*.https://www.statlect.com/fundamentals-of-statistics/Poisson-distribution-maximum-likelihood
- Kull, M., Ferri, C., & Martínez-Usó, A. *Bike rental and weather data across dozens of cities*
- Pisinger, D., & Ropke, S. (1005). Large neighborhood search. *Handbook of Metaheuristics, *, 399. doi:10.1007/978-1-4419-1665-5_13
- Raviv, T., Tzur, M., & Forma, I. A. (2013). Static repositioning in a bike-sharing system: Models and solution approaches. *EURO Journal on Transportation and Logistics, 2*(3), 187-229. doi:10.1007/s13676-012-0017-6
- Schuijbroek, J., Hampshire, R., & Willem-Jan Van Hoeve. (2018). *Inventory rebalancing and vehicle routing in bike sharing systems* Figshare. doi:10.1184/r1/6706265.v1

- Tormos, P. y Lova, A. *Investigación operativa para ingenieros* SPUPV-2003.591.
- Tormos, P. y Lova, A. (2014). Diseño de sistemas de optimización basados en hoja de cálculo.
- *The vehicle routing problem: Latest advances and new challenges*. Boston, MA: Springer US.

12. Anexo

12.1. Anexo 1: Gráficos de los análisis.

Figura 32, información según el día de la semana de la estación 21

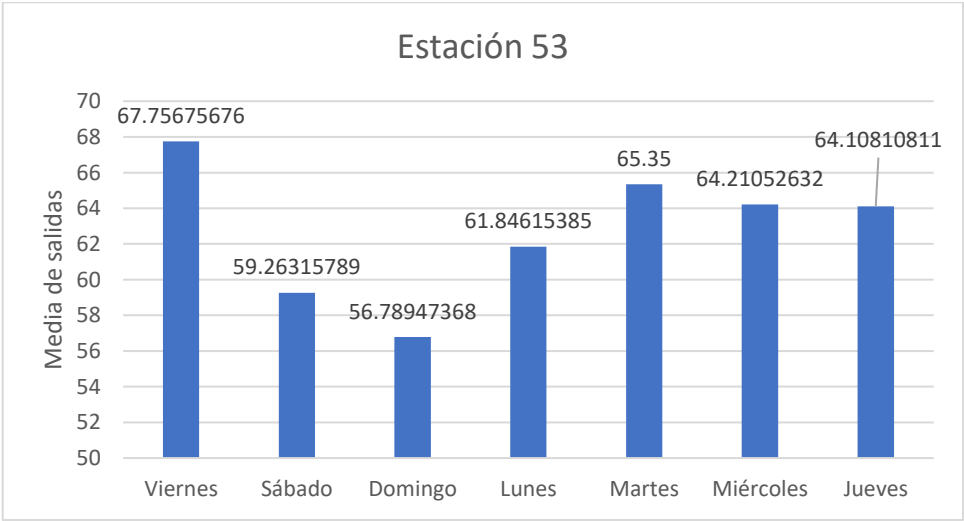


Figura 33, información según el día de la semana de la estación 53

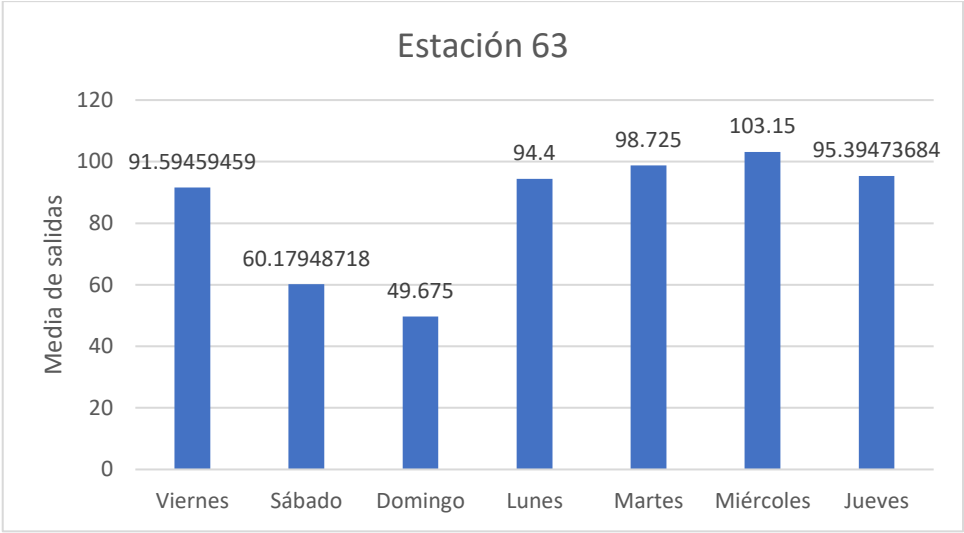


Figura 34, información según el día de la semana de la estación 63

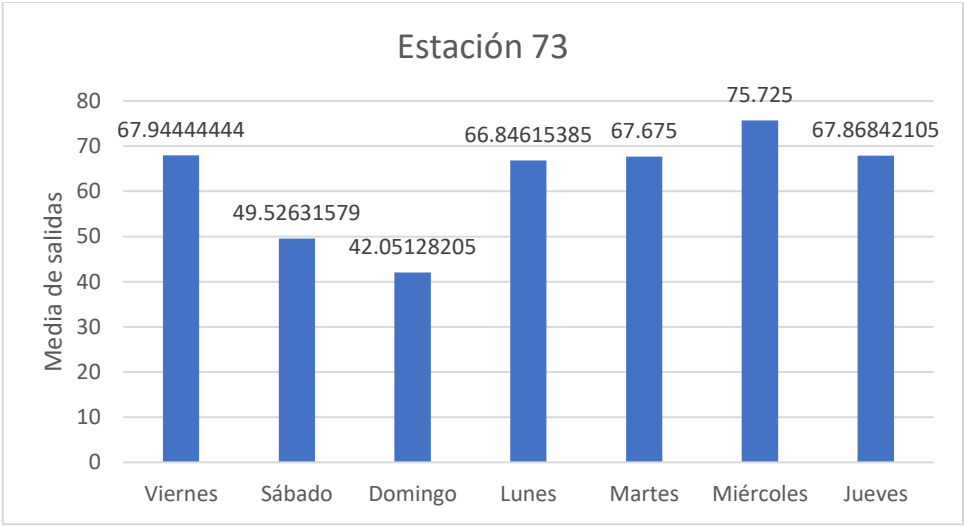


Figura 35, información según el día de la semana de la estación 73

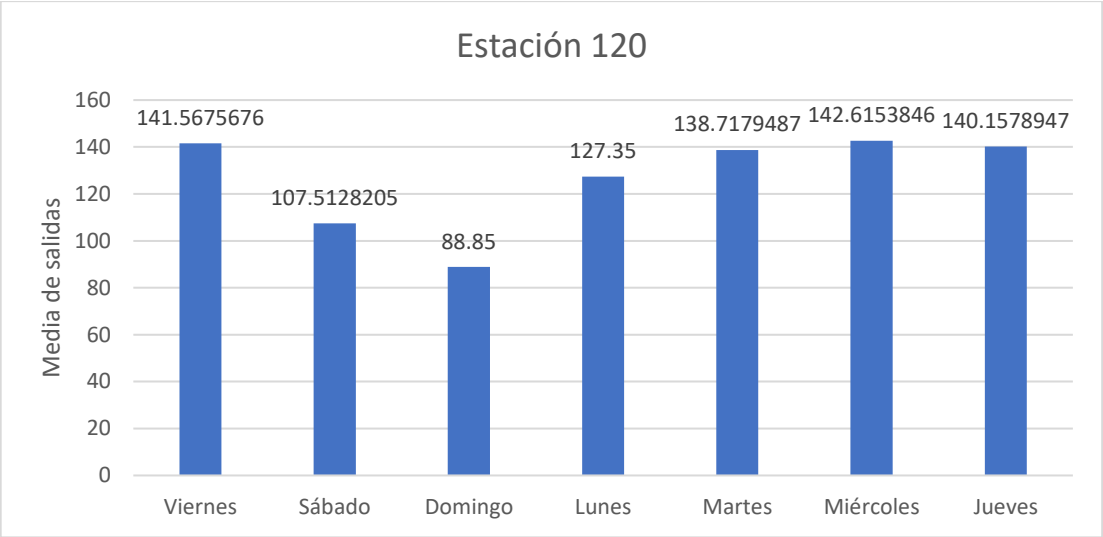


Figura 36, información según el día de la semana de la estación 120

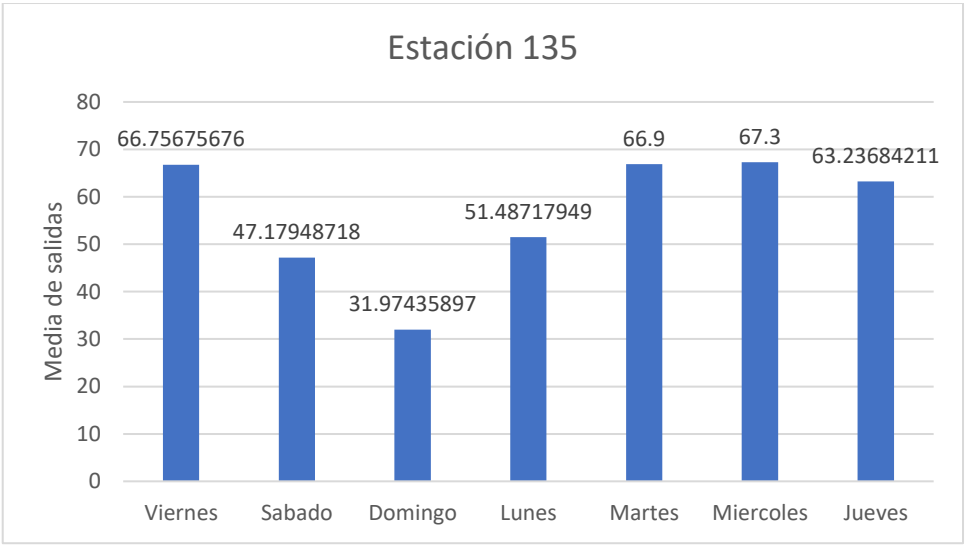


Figura 37, información según el día de la semana de la estación 135

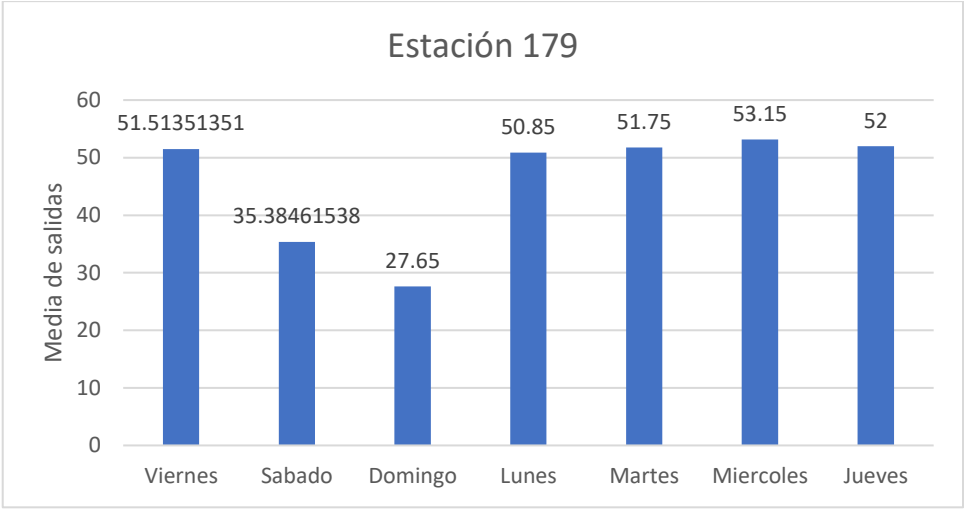


Figura 38, información según el día de la semana de la estación 179

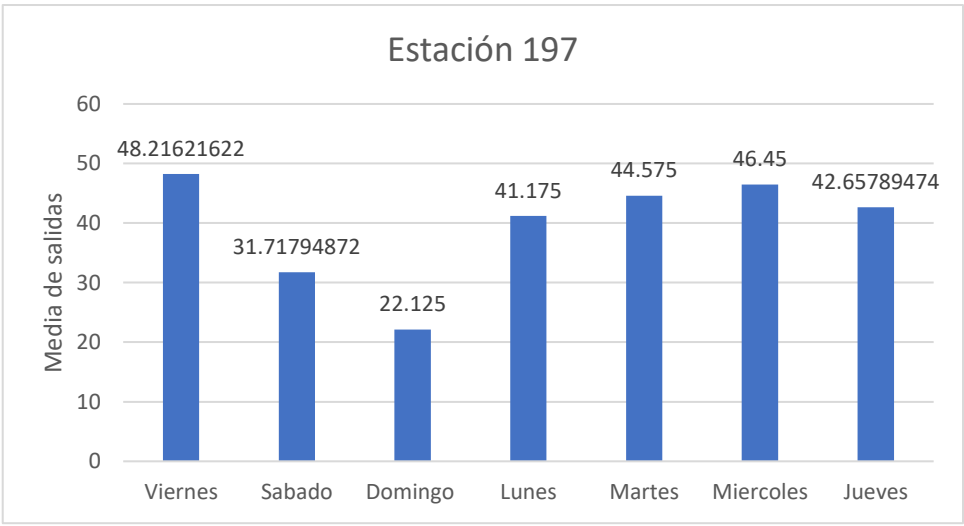


Figura 39, información según el día de la semana de la estación 197

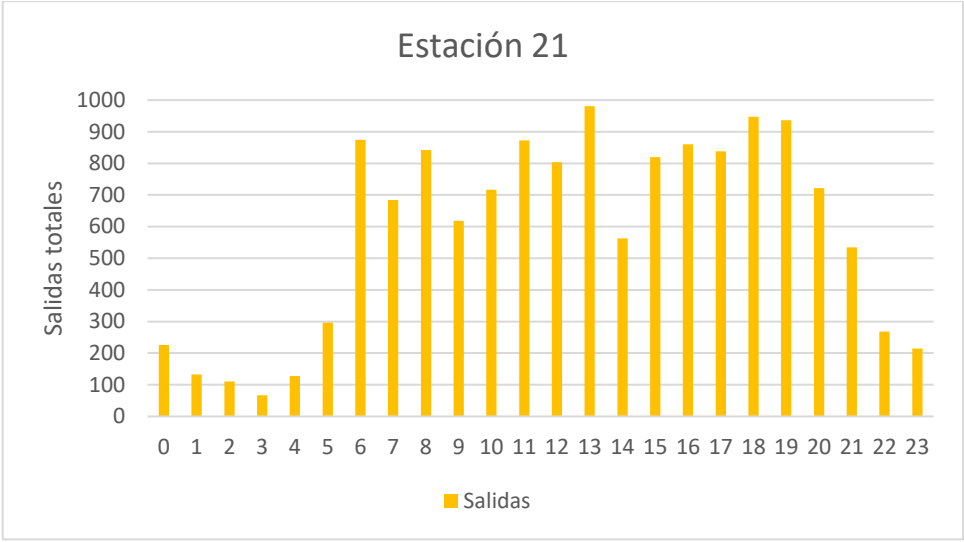


Figura 40, salidas totales por hora en la estación 21

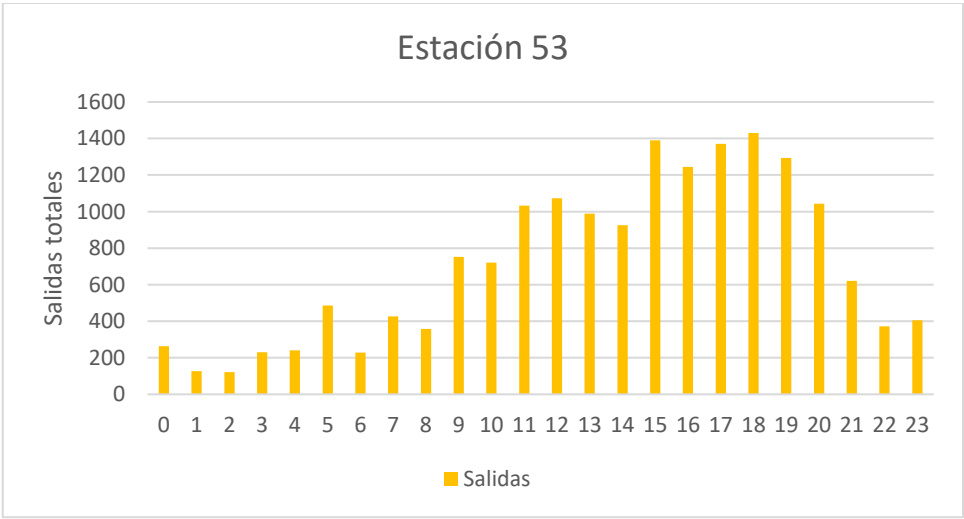


Figura 41, salidas totales por hora en la estación 53

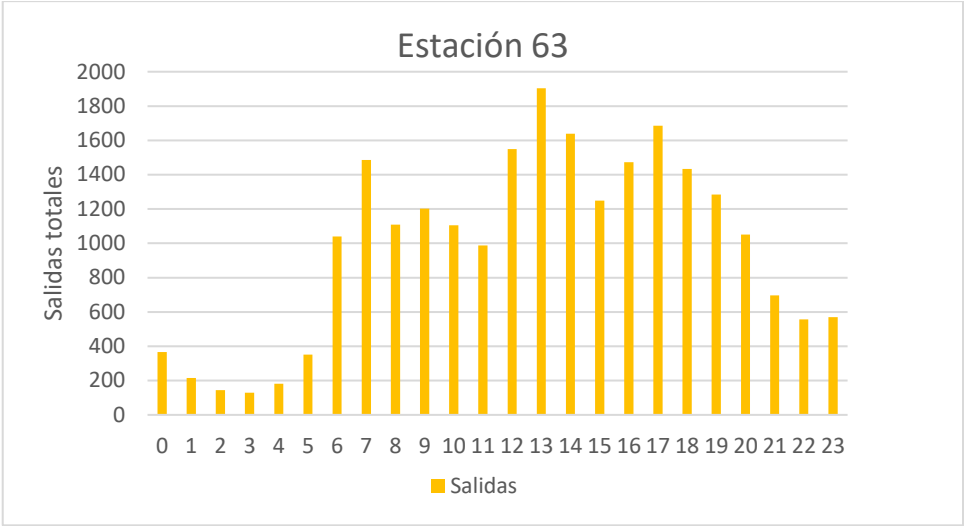


Figura 42, salidas totales por hora en la estación 63

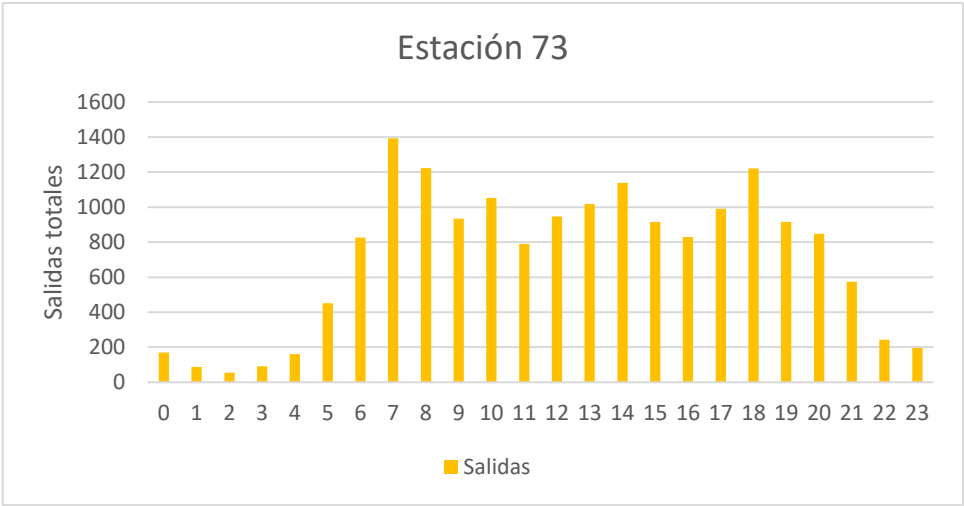


Figura 43, salidas totales por hora en la estación 73

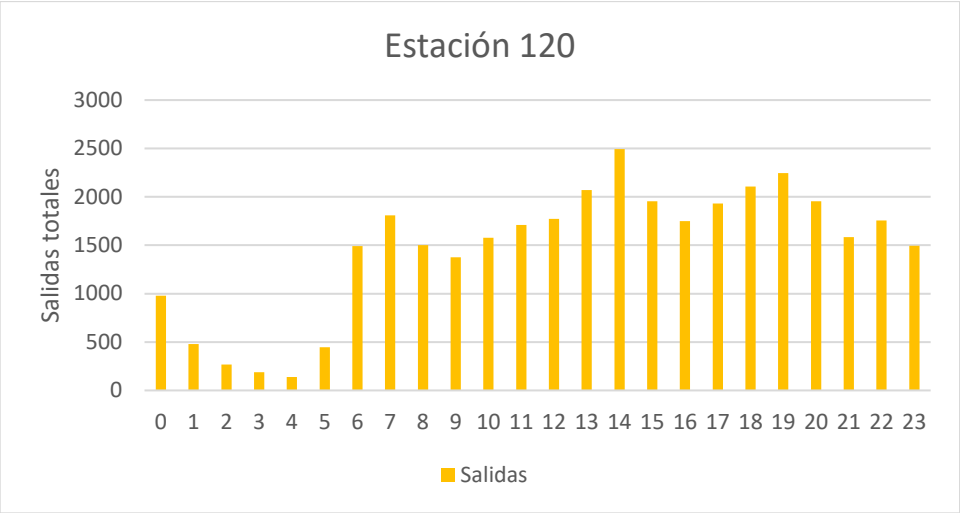


Figura 44, salidas totales por hora en la estación 120

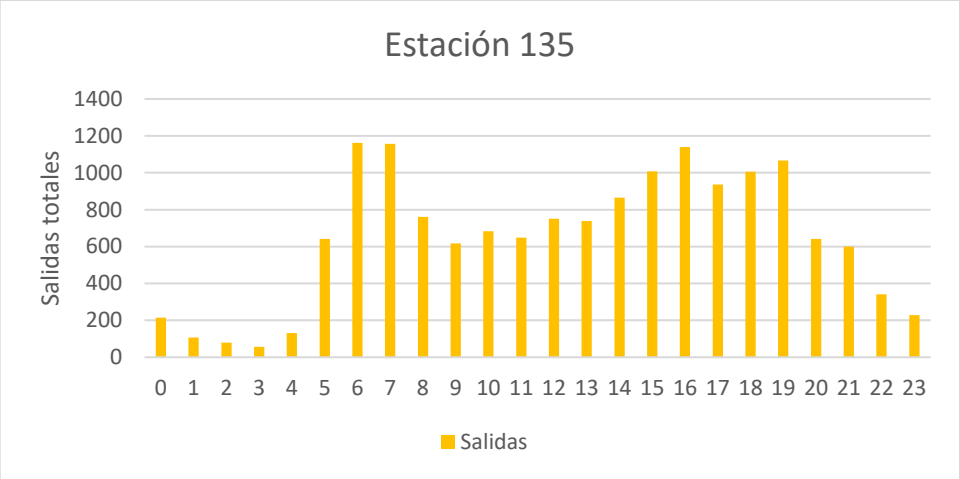


Figura 45, salidas totales por hora en la estación 135

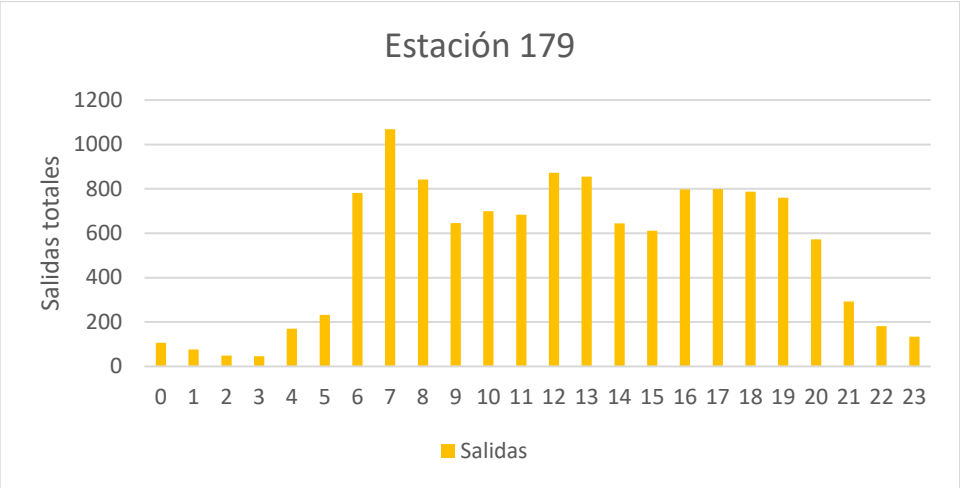


Figura 46, salidas totales por hora en la estación 179

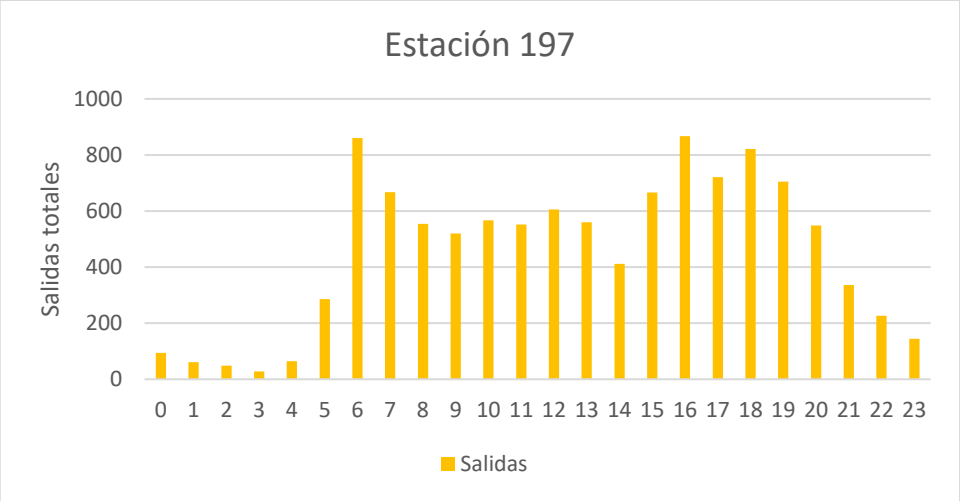


Figura 47, salidas totales por hora en la estación 197

12.2. Anexo 2: Rutas de las soluciones del método de mínima distancia.

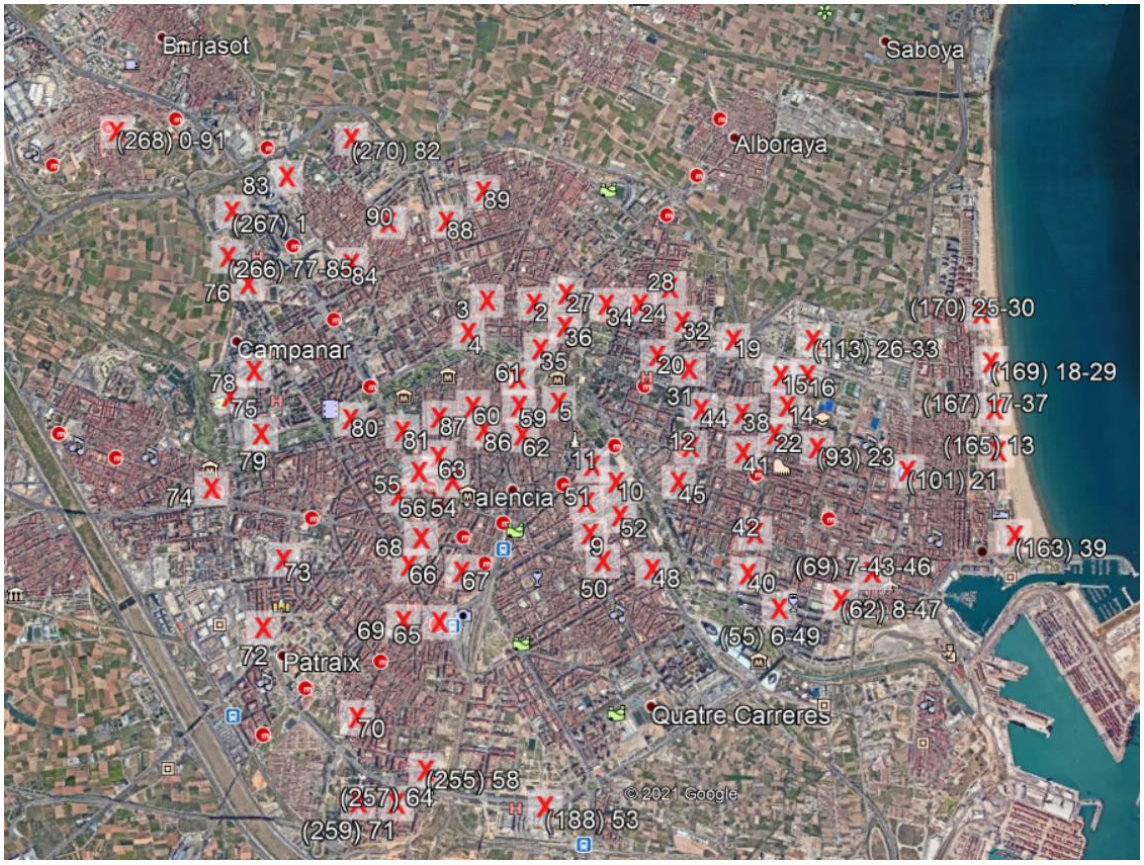


Figura 48, ruta solución 11/11/2014 (mínima distancia)

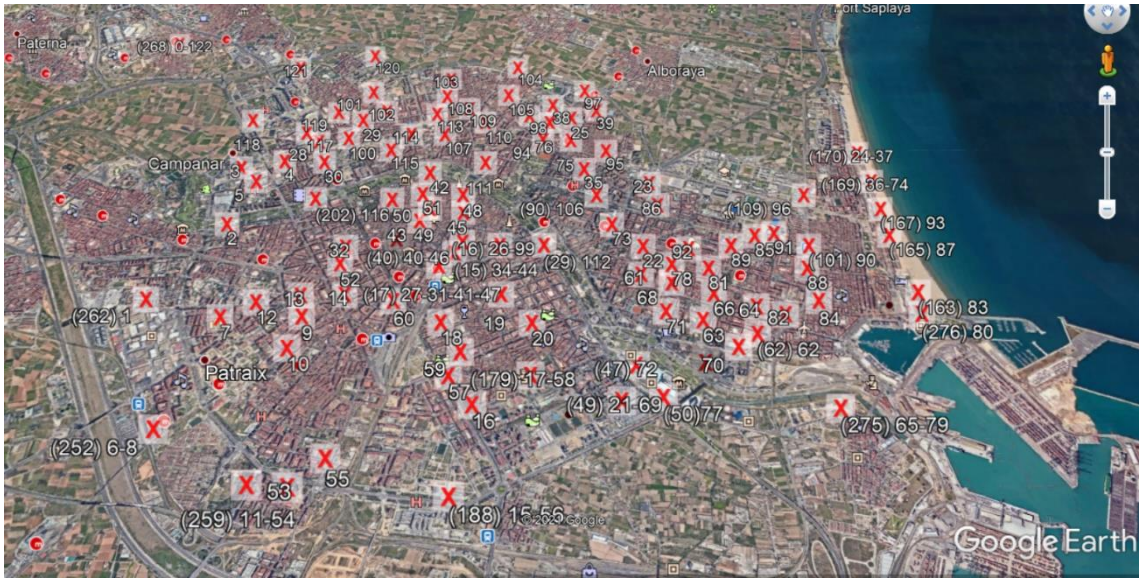


Figura 49, ruta solución 15/12/2014 (mínima distancia)

Figura 50, solución ruta 15/12/2014 (mínima distancia)

Figura 51, solución ruta 01/02/2015 (mínima distancia)

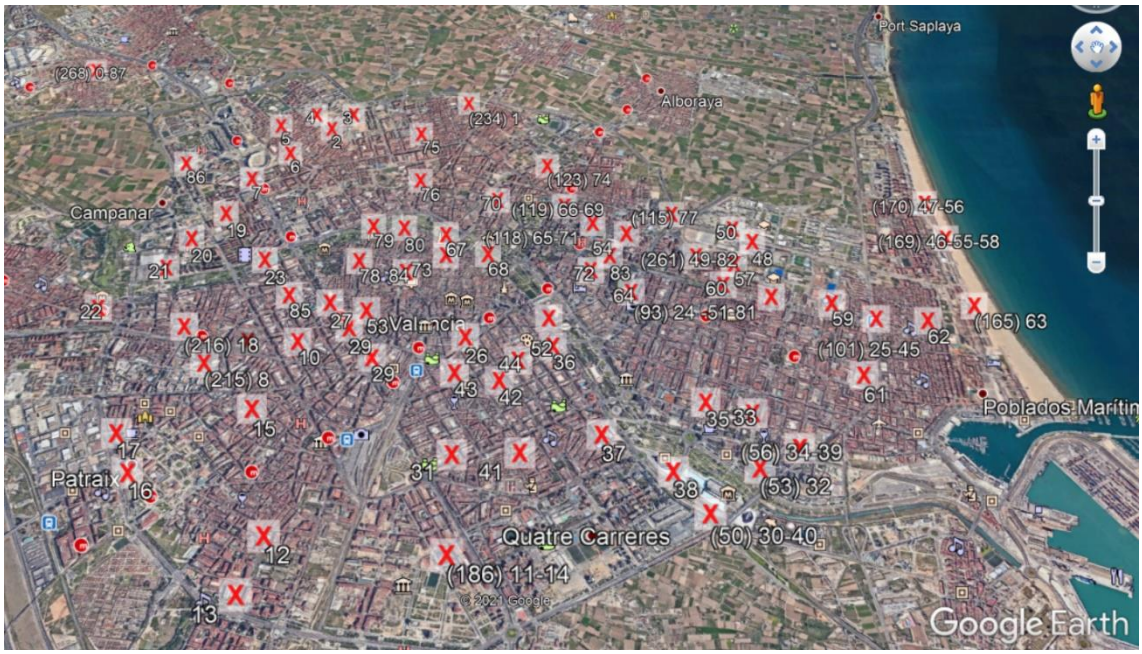


Figura 52, solución ruta 27/03/2015 (mínima distancia)

Figura 53, solución ruta 21/05/2015 (mínima distancia)

Figura 54, zoom sobre la ruta del 21/05/2015 (mínima distancia)

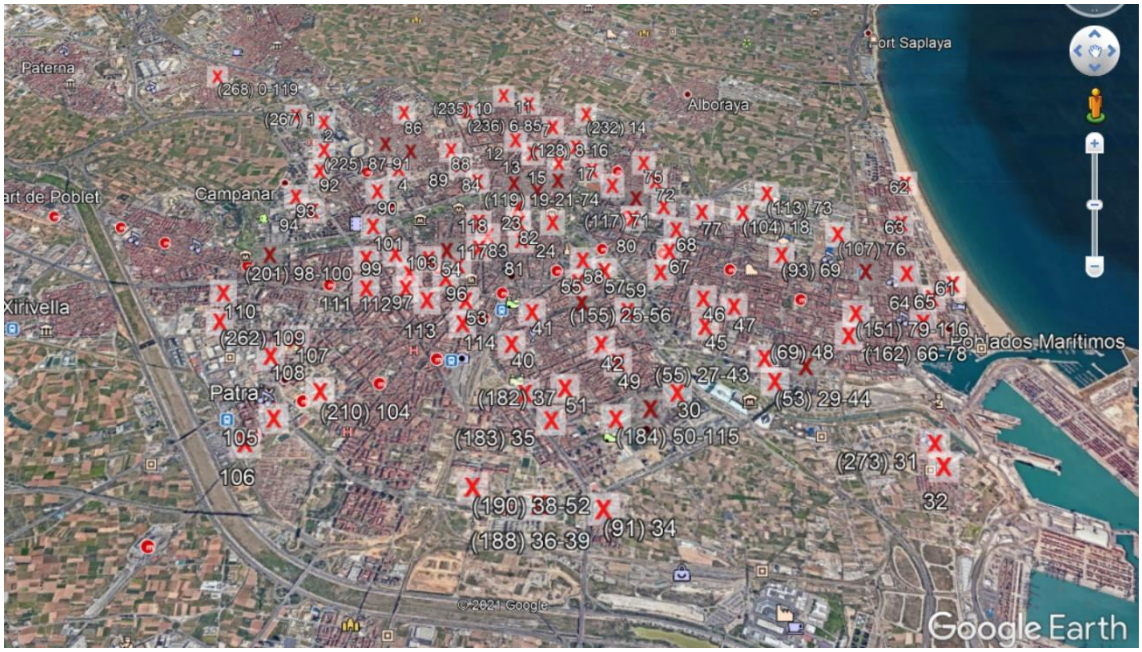


Figura 55, solución ruta 01/07/2015 (mínima distancia)

Figura 56, zoom sobre la solución ruta 01/07/2015 (mínima distancia)

12.3. Anexo 3: Rutas de las soluciones del método aleatorio.

Figura 57, ruta solución 27/09/2014 (aleatorio)

Figura 58, ruta solución 11/11/2014 (aleatorio)

Figura 59, zoom sobre la ruta solución 11/11/2014 (aleatorio)

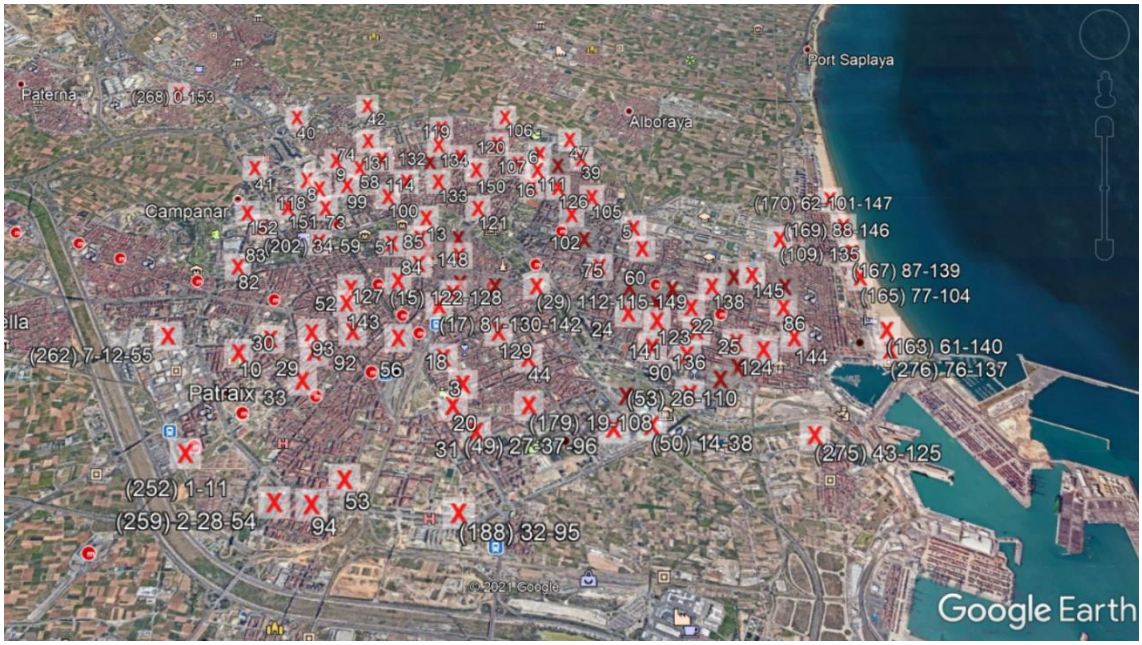


Figura 60, ruta solución 15/12/2014 (aleatorio)

Figura 61, zoom sobre ruta solución 15/12/2014

Figura 62, zoom sobre ruta solución 15/12/2014

Figura 63, ruta solución 01/02/2015 (aleatorio)

Figura 64, ruta solución 27/03/2015 (aleatorio)

Figura 65, zoom ruta solución 27/03/2015 (aleatorio)

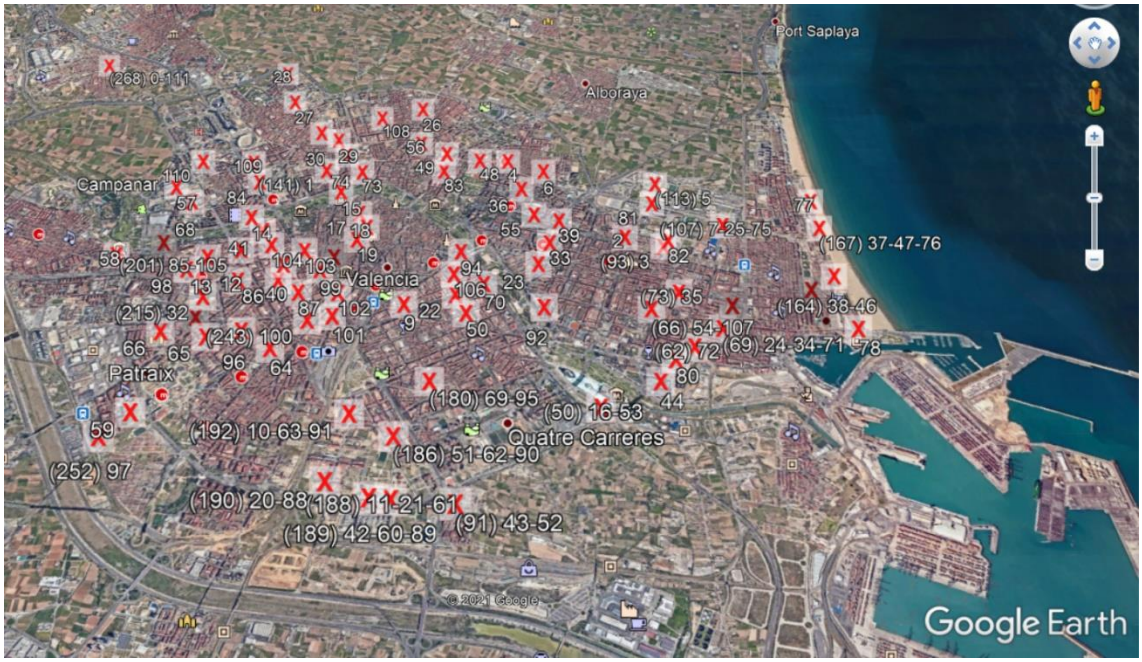


Figura 66, ruta solución 21/05/2015 (aleatorio)

Figura 67, zoom sobre ruta solución 21/05/2015

Figura 68, zoom sobre ruta solución 21/05/2015

Figura 69, ruta solución 01/07/2015 (aleatorio)

Figura 70, zoom sobre ruta solución 01/07/2015 (aleatorio)

Figura 71, zoom sobre ruta solución 01/07/2015 (aleatorio)

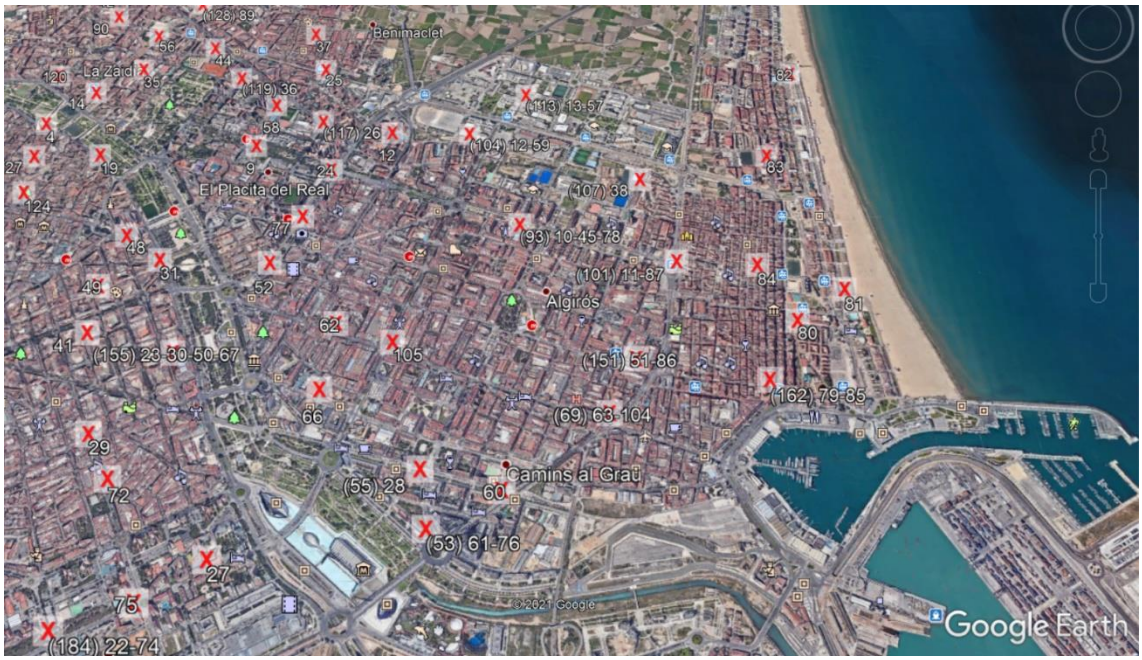


Figura 72, zoom sobre ruta solución 01/07/2015 (aleatorio)

12.4. Anexo 4: Código completo.

```
"""Simple Pickup Delivery Problem (PDP)."""

from ortools.constraint_solver import routing_enums_pb2
from ortools.constraint_solver import pywrapcp


def create_data_model():
    """Stores the data for the problem."""
    data = {}
    data['distance_matrix'] = [
    [0,15,15,8,8],
    [15,0,0,6,12],
    [15,0,0,6,12],
    [8,6,6,0,9],
    [8,12,12,9,0]
]
    data['pickups_deliveries'] = [

       [1,3],
       [2,4],
    ]
    data['num_vehicles'] = 1
    data['depot'] = 0
    data['vehicle_capacities'] = [10]
    data['flow'] = {
        0:0,
        1:2,
        2:1,
        3:-2,
        4:-1,

    }
    return data
```

Figura 73, primera parte del código.

```python
def print_solution(data, manager, routing, solution):
    """Prints solution on console."""
    print(f'Objective: {solution.ObjectiveValue()}')
    total_distance = 0
    for vehicle_id in range(data['num_vehicles']):
        index = routing.Start(vehicle_id)
        plan_output = 'Route for vehicle {}:\n'.format(vehicle_id)
        route_distance = 0
        while not routing.IsEnd(index):
            plan_output += ' {} -> '.format(manager.IndexToNode(index))
            previous_index = index
            index = solution.Value(routing.NextVar(index))
            route_distance += routing.GetArcCostForVehicle(
                previous_index, index, vehicle_id)
        plan_output += '{}\n'.format(manager.IndexToNode(index))
        plan_output += 'Distance of the route: {}m\n'.format(route_distance)
        print(plan_output)
        total_distance += route_distance
    print('Total Distance of all routes: {}m'.format(total_distance))


def main():
    """Entry point of the program."""
    # Instantiate the data problem.
    data = create_data_model()

    # Create the routing index manager.
    manager = pywrapcp.RoutingIndexManager(len(data['distance_matrix']),
                                           data['num_vehicles'], data['depot'])

    # Create Routing Model.
    routing = pywrapcp.RoutingModel(manager)
```

Figura 74, segunda parte del código.

```
# Define cost of each arc.
def distance_callback(from_index, to_index):
    """Returns the manhattan distance between the two nodes."""
    # Convert from routing variable Index to distance matrix NodeIndex.
    from_node = manager.IndexToNode(from_index)
    to_node = manager.IndexToNode(to_index)
    return data['distance_matrix'][from_node][to_node]



transit_callback_index = routing.RegisterTransitCallback(distance_callback)
routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index)

dimension_name = 'Weight'

def load_or_unload(from_index):
    from_node = manager.IndexToNode(from_index)
    origin = [x[0] for x in data["pickups_deliveries"]]
    destination = [x[1] for x in data["pickups_deliveries"]]
    if from_node in data["flow"]:
        return data["flow"][from_node]
    #if from_node in origin :
    #    return 1
    #if from_node in destination:
    #    return -1
    return 0
```

Figura 75, tercera parte del código.

```
    routing.AddDimensionWithVehicleCapacity(
        routing.RegisterUnaryTransitCallback(load_or_unload),
        0,
        data['vehicle_capacities'],
        True,
        'Capacity'
    )

    # Add Distance constraint.
    dimension_name = 'Distance'
    routing.AddDimension(
        transit_callback_index,
        0,  # no slack
        30000000,  # vehicle maximum travel distance
        True,  # start cumul to zero
        dimension_name)
    distance_dimension = routing.GetDimensionOrDie(dimension_name)
    distance_dimension.SetGlobalSpanCostCoefficient(100)

    # Define Transportation Requests.
    for request in data['pickups_deliveries']:
        pickup_index = manager.NodeToIndex(request[0])
        delivery_index = manager.NodeToIndex(request[1])
        routing.AddPickupAndDelivery(pickup_index, delivery_index)
        routing.solver().Add(
            routing.VehicleVar(pickup_index) == routing.VehicleVar(
                delivery_index))
        routing.solver().Add(
            distance_dimension.CumulVar(pickup_index) <=
            distance_dimension.CumulVar(delivery_index))
```

Figura 76, cuarta parte del código.

```
    # Setting first solution heuristic.
    search_parameters = pywrapcp.DefaultRoutingSearchParameters()
    search_parameters.first_solution_strategy = (
        routing_enums_pb2.FirstSolutionStrategy.PARALLEL_CHEAPEST_INSERTION)

    # Solve the problem.
    solution = routing.SolveWithParameters(search_parameters)

    # Print solution on console.
    if solution:
        print_solution(data, manager, routing, solution)


if __name__ == '__main__':
    main()
```

Figura 77, última parte del código