UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de un simulador de carga de vehículos eléctricos a nivel interurbano

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Llopis Herrero, Jaime

Tutor/a: Julian Inglada, Vicente Javier

Cotutor/a: Jordán Prunera, Jaume Magí

CURSO ACADÉMICO: 2022/2023

Resum

El cotxe elèctric està tenint una millor penetració en el parc automovilístic espanyol, degut a la concienciació sobre el canvi climàtic, la previsió de l'esgotament de combustibles fòssils i l'estalvi. A Espanya, hi existeixen molts punts de recàrrega públics pels vehicles, la majoria de baix voltatge, fet que obstaculitza la possibilitat de realitzar trajectes llargs entre ciutats amb vehicles elèctrics. Açò provoca als conductors de vehicles elèctrics el fenòmen anomenat *range anxiety* a l'hora de dur a terme un viatge interurbà. La millor xarxa de recàrrega de bateries d'alt voltatge d'automòbils existent a Espanya es la construïda per l'empresa Tesla. Una de les investigacions que s'estan realitzant en aquest aspecte als últims anys a l'entorn urbà pretén dissenyar una xarxa d'estacions de recàrrega òptima pels usuaris als municipis tenint en compte informaciò demogràfica vertadera de la poblaciò. Per mitjà d'un sistema intel·ligent, implementat mitjançant un algorisme genètic, els investigadors han trobat una manera de generar un mapa de punts de recàrrega que satisfaga les necessitats dels ciutadans en qualsevol entorn. El present Treball de Fi de Grau té el propòsit de dissenyar i desenvolupar un entorn de simulació per posar a prova les xarxes d'estacions de recàrrega generades per l'algorisme genètic a nivell interurbà, analitzant i interpretant de manera empírica la seua bondat. Adicionalment, es realitzaran experiments per comparar els resultats dels anteriors amb la xarxa d'estacions de recàrrega de Tesla i altres distribucions fictícies que es poden obtindre de forma aleatòria o matemàtica.

Paraules clau: vehicle elèctric, interurbà, estació de recàrrega, algorisme genètic, simulació multi-agent, SimFleet, Tesla

---

Resumen

El coche eléctrico está teniendo una cada vez mayor penetración en el parque automovilístico español debido a la concienciación sobre el cambio climático, la previsión de agotamiento de combustibles fósiles y el ahorro. En España existen varios puntos públicos de recarga de vehículos, la mayoría de bajo voltaje, hecho que obstaculiza el poder realizar trayectos largos entre ciudades con vehículos eléctricos. Esto provoca en los conductores de vehículos eléctricos el fenómeno llamado *range anxiety* a la hora de llevar a cabo un viaje interurbano. La mejor red de recarga de baterías de alto voltaje de automóviles existente en España es la construida por la empresa Tesla. Una de las investigaciones que se están realizando en este aspecto en los últimos años en el entorno urbano pretende diseñar una red de estaciones de recarga óptima para los usuarios en los municipios teniendo en cuenta información demográfica real de la población. Por medio de un sistema inteligente, implementado mediante un algoritmo genético, los investigadores han logrado generar un mapa de puntos de recarga que satisfaga las necesidades de los ciudadanos en cualquier entorno. El presente Trabajo de Fin de Grado tiene el propósito de diseñar y desarrollar un entorno de simulación para poner a prueba las redes de estaciones de recarga generadas por el algoritmo genético a nivel interurbano, analizando e interpretando de manera empírica su bondad. Adicionalmente, se realizarán experimentos para comparar los resultados de los anteriores con la red de estaciones de recarga de Tesla y con sendas distribuciones ficticias que se pueden obtener de manera aleatoria o matemática.

Palabras clave: vehículo eléctrico, interurbano, estación de recarga, algoritmo genético, simulación multi-agente, SimFleet, Tesla

---

Abstract

The electric car is continually increasing its market penetration in the Spanish vehicle pool due to climate change awareness, the expectation of fossil resources depletion in the near future and money saving. There are many public charging stations in Spain, mostly low-powered. This fact impedes people to make longer trips between cities with their electric cars. It also develops the phenomenon known as ”range anxiety” in users planning an interurban journey. The best existing electric supercharger network in Spain is the one built by Tesla. One of the developing investigations on this matter in the past few years pretends to design an optimal electric charging station network for citizens taking real demographic information into consideration. Using an intelligent system implemented via genetic algorithms, investigators have found a way to generate an electric charging station map that satisfies the needs of all citizens in any region. This Final Degree Project intends to design and develop a simulation environment to put the results of the genetic algorithm under proper testing conditions, performing an empiric analysis and interpretation. Additionally, more experiments will take place to compare these results with the ones regarding the Tesla charging station network and other fictitious station distributions, which can be mathematically or randomly obtained.

Key words: electric vehicle, interurban, charging station, genetic algorithm, multi-agent simulation, SimFleet, Tesla

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

Es por todos conocido que el medio ambiente de nuestro planeta corre peligro debido a la contaminación. Las emisiones de gases de efecto invernadero, los vertidos, la acumulación de residuos, el despilfarro de recursos básicos como el agua y muchas más acciones que, por desgracia, son cotidianas; tienen graves consecuencias tales como el calentamiento global, la polución del aire en ciudades, la extinción de muchas especies animales y vegetales etc. Son muchas las personas y las industrias que se implican en frenar el avance de este problema, aunque las iniciativas siguen siendo insuficientes para poder abordarlo de manera contundente.

Uno de los sectores que cobra especial importancia en esta cuestión es el sector privado de la automoción. Si nos centramos en España, el parque automovilístico está muy envejecido en comparación con la media europea (13,1 años de media frente a 10,1), con el mayor consumo de combustible y contaminación que ello conlleva [1]. La contaminación del sector del transporte se debe principalmente a las emisiones de gas de efecto invernadero que producen los vehículos y las plantas de extracción y refinería de petróleo que producen combustible. En España supone el sector que más emisiones genera, con un 27,7 % del total [2]. Tal es el nivel de polución generado en ciudades que, por ejemplo, en Madrid se ha tenido que regular la circulación de vehículos privados para tratar de reducir la concentración de sustancias tóxicas en el aire que se respira[1]. Esta medida solamente es un parche temporal para aliviar la calidad de vida de los madrileños, pero se requieren alternativas reales que solucionen este problema. Cabe mencionar también que el coche emplea un recurso fósil, la gasolina o el diésel, para poder moverse. Como todos los recursos fósiles, están destinados a agotarse. En consecuencia, los coches, tal como los conocemos hoy en día, pueden tener los días contados.

Hace ya varios años que irrumpió en España el vehículo eléctrico como alternativa a los equivalentes de combustión. Son muchas las ventajas de adquirir uno: la eliminación de toda emisión que genera el vehículo durante sus trayectos y del ruido que producen los motores de combustión; poder instalar un cargador en el lugar habitual de aparcamiento del coche, la posibilidad de que el vehículo recargue automáticamente sus baterías durante los trayectos (por ejemplo durante el frenado) y ahorrar en gasolina.

Su impacto en España siempre ha sido muy bajo, aunque aumenta considerablemente en los últimos años. La ANFAC[2] establece la penetración del coche eléctrico en 31.6 % en 2020, aumentando en 9 puntos desde el año anterior [1]. Este indicador refiere a la proporción de clientes que adquieren un producto en un mercado. En el caso de España, por tanto, a fecha del informe, casi uno de cada tres compradores de vehículos adquieren uno

[1] Noticia de Expansión, 28 de diciembre de 2016:
www.expansion.com/sociedad/2016/12/28/586397e3e2704e27028b4603.html

[2] Asociación Española de Fabricantes de Automóviles y Camiones

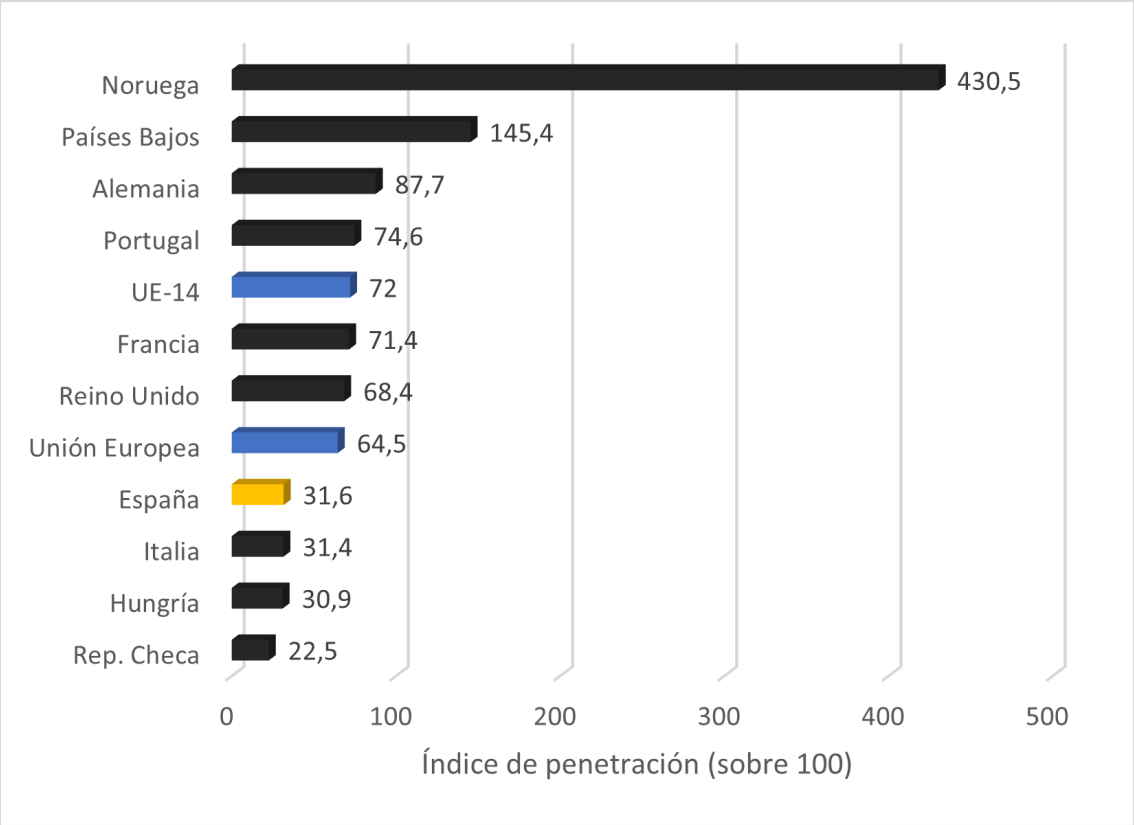


Figura 1.1: Índice de penetración del vehículo electrificado en Europa

Fuente: ANFAC [1], elaboración propia

eléctrico. Cabe mencionar que la penetración media en Europa asciende al doble de dicha cifra. A su vez, supone respectivamente, la mitad y una sexta parte de la penetración en los países líderes, Países Bajos y Noruega (ver figura 1.1). Se pronostica que la cuota continúe aumentando a lo largo de los siguientes años debido al abaratamiento de los costes de fabricación de baterías por debajo de lo previsto (actualmente un 15 % menor) y el bajo mantenimiento que requiere este tipo de vehículo [3]. En los últimos dos años, los precios de la gasolina y del diésel se encuentran pulverizando récords día tras día, lo que hace que muchas personas se planteen cambiar de coche para evitar esta situación tan insostenible.

Por desgracia, son también varias las desventajas. Principalmente: la poca autonomía que tienen en comparación con los coches tradicionales y la ausencia de una buena infraestructura de recarga de baterías a lo largo del país, tanto en número como en distribución. Quizá la red de estaciones de carga rápida más reconocida en España es la construida por la empresa americana Tesla, con más de 50 estaciones repartidas por toda la península[3] (ver figura 1.2).

Es cierto que estos coches suelen emplearse para trayectos urbanos, porque la batería tiene sobrada capacidad para hacer desplazar el coche por ciudad y es más sencillo repostar energía eléctrica en el punto de carga doméstico o en muchos otros emplazamientos como supermercados, zonas de aparcamiento, o a veces, en la misma calle. El conflicto aparece cuando se quiere realizar un viaje fuera de ciudad, porque dicho viaje puede quedar limitado a la distancia máxima que pueda recorrer el vehículo sin enchufarse a la red.

[3] www.tesla.com/es_ES/findus/list/superchargers/spain

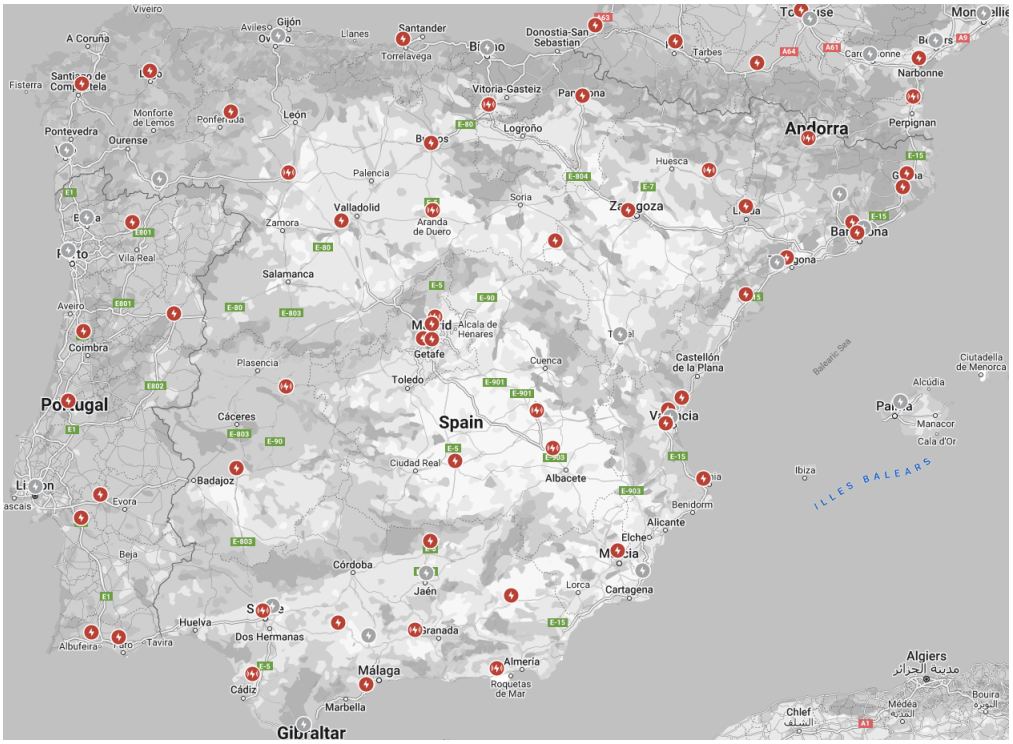


Figura 1.2: Red de supercargadores de Tesla en la Península Ibérica (color rojo)

Fuente: Tesla. Consultado el 29 de junio de 2022

Este hecho está relacionado con un concepto que en inglés se conoce como *range anxiety*. En pocas palabras, simboliza el miedo que padecen los conductores cuando tienen la incertidumbre de poder quedarse sin carga a mitad trayecto [4]. Se trata de una de las mayores preocupaciones de los propietarios de coches eléctricos, pues, como se ha mencionado anteriormente, actualmente en España no existe una red de estaciones de recarga en condiciones para satisfacer la demanda de estas personas.

En los últimos años se ha estado investigando acerca de un diseño único de dicha red, de manera que se pueda reducir al máximo la *range anxiety* de los conductores, minimizando al mismo tiempo la inversión necesaria para la construcción de las estaciones de carga pertinentes. Veremos más adelante una de las aproximaciones que se han realizado para poder colocar las estaciones de manera óptima según una serie de factores. Una vez obtenida una red que satisfaga las necesidades de conductores y del propietario de la red, ya sea el gobierno o una empresa inversora privada, antes de implementarla, surge la necesidad de realizar una prueba piloto para ponerla a prueba. Esta prueba tendría como requisito que se colocaran unas estaciones provisionales en las localizaciones calculadas, para que una carga experimental y representativa de vehículos eléctricos realizaran una serie de trayectos habituales a lo largo del territorio español. Instalar esas estaciones provisionales supondría un gran coste, y tanto en caso de éxito o fracaso, se tendría que desmantelar la red o construir las estaciones definitivas, con el desembolso adicional que supondría. Por esta razón, se baraja la alternativa de realizar una simulación virtual de una jornada ejemplar de trayectos interurbanos en España.

El presente proyecto propone realizar este experimento informático, colocando estaciones de recarga en el mapa español, haciendo circular vehículos que recarguen baterías según sus necesidades, y finalmente, recabando estadísticas acerca de todos los agentes involucrados en la simulación. Se pretende realizar una demostración empírica de la bondad de las distribuciones seleccionadas y generadas por los participantes del proyecto y

comparar los resultados con la red actual de estaciones de recarga de Tesla. Se trata, en definitiva, de una solución suficiente para comprobar si, hoy en día, podemos movernos libremente por España con un vehículo eléctrico, o cómo se puede mejorar con la información disponible.

1.1 Motivación

Este proyecto fue propuesto por los tutores del trabajo como continuación de la investigación realizada por Jordán et al [5, 6]. Dicho trabajo pretendía hallar una manera de optimizar la distribución de estaciones de recarga a lo largo de cierta zona geográfica. Inicialmente, su proyecto se realizó únicamente a nivel municipal, ampliándose posteriormente a un nivel nacional. Para completar la investigación, es necesario analizar si las disposiciones resultantes son útiles en la práctica, por tanto, se requiere de un software que sea capaz de simular una jornada cotidiana de viajes entre ciudades con vehículos eléctricos.

Además, como personas concienciadas con el medio ambiente, consideramos este trabajo como muy relevante para ayudar en la medida de nuestras posibilidades a afrontar el cambio climático. No es suficiente con las acciones que realizamos todos los días, como reciclar, ahorrar agua y plásticos, etc. Creemos que hay que implicarse, hacer algo más y aportar nuestro granito de arena. Pretendemos proporcionar nuevas ideas o herramientas para mejorar la red de estaciones de recarga en España y fomentar el uso de vehículos eléctricos.

1.2 Objetivos

El objetivo de este trabajo es disponer de un proceso de simulación de flotas de vehículos eléctricos en entornos interurbanos que permita llevar a cabo un estudio respecto a los mejores emplazamientos posibles de estaciones de recarga eléctrica a lo largo de una determinada región.

Dicho objetivo se subdivide en los siguientes subobjetivos:

1. Realizar un estudio de las alternativas existentes en cuanto a simuladores de flotas de vehículos y analizar cuál es la más adecuada para el desarrollo propuesto.

2. Implementar los cambios necesarios sobre el simulador para adaptarlo a las necesidades concretas de los experimentos.

3. Analizar la bondad de una serie de distribuciones de estaciones de recarga ficticias siguiendo un modelo matemático o aleatorio en función del número de estaciones colocadas y la autonomía de los vehículos.

4. Analizar la bondad de la distribución de estaciones de recarga óptima según el algoritmo genético desarrollado por Jordán et al [6], en función del número de estaciones colocadas y la autonomía de los vehículos.

5. Analizar la bondad de la red de estaciones de recarga de Tesla existente en España en función de la autonomía de los vehículos.

6. Realizar un análisis comparativo de los resultados de los experimentos, estableciendo qué distribuciones son más eficaces y eficientes.

7. Extraer puntos de mejora de la distribución Tesla a partir de las conclusiones de las distribuciones ficticias.

1.3 Estructura de la memoria

Una vez situados en el contexto del proyecto a desarrollar y para que sirva de orientación al lector, vamos a indicar brevemente qué orden se va a seguir en esta memoria y en qué consiste cada capítulo de la misma:

El capítulo que sigue a esta introducción (capítulo 2) consta de un análisis del estado de la cuestión. Está dividido en dos partes: primeramente se analizarán las tecnologías, trabajos e investigaciones existentes y relacionadas con el presente proyecto; y los distintos simuladores que se han barajado para llevarlo a cabo. Seguidamente, se expondrá cuál es el simulador escogido para llevar a cabo los experimentos, incluyendo la tecnología con la que se implementará y los motivos concretos que han decantado la decisión.

A continuación, en el capítulo 3, se analizará en profundidad las características específicas del simulador escogido, el problema a resolver y cómo se han diseñado las adaptaciones al programa empleado. Esto incluye cómo se plantea la configuración del programa para plasmar la situación de los vehículos y las estaciones de recarga con la mayor proximidad a la realidad posible. Además, se explicará el funcionamiento de la herramienta de generación de flotas y de estaciones para los archivos de configuración y las ampliaciones que se necesita realizar para poder utilizarla junto con el simulador.

Tras el diseño de la solución, en el capítulo 4, veremos cómo se han implementado los cambios en el comportamiento del simulador y qué implicaciones han tenido. Asimismo, se presentarán las modificaciones y adiciones llevadas a cabo en el paquete de generadores de ficheros de configuración. También veremos brevemente cómo se han generado los ficheros de configuración de los experimentos y qué pautas se han seguido para recrear los atributos de las estaciones de recarga y los transportes.

Siguiendo los objetivos planteados, en el capítulo 5 presentaremos las métricas obtenidas después de lanzar todas las simulaciones planificadas. Cada una de las más de cien simulaciones lanzadas proporcionará una entrada en el registro de resultados. Estos registros se agruparán por tipo de distribución, autonomía de los vehículos y número de estaciones colocadas. Individualmente, para cada distribución de estaciones, realizaremos en primer lugar un análisis puramente estadístico de los datos recuperados de los experimentos. En segundo lugar interpretaremos lo que suponen los resultados para la movilidad de los vehículos interurbanos eléctricos. A modo de cierre del capítulo, haremos una comparación de los resultados obtenidos en cada tipo de disposición de estaciones. Para la distribución de Tesla, se mencionarán aspectos a mejorar a la vista de los datos observados en las demás estrategias de emplazamiento de estaciones.

Finalmente, en el capítulo 6, veremos las conclusiones globales a las que se ha llegado después de analizar los resultados experimentales. Comentaremos la relación que ha tenido este trabajo con los estudios cursados por el autor. Se hará un breve comentario acerca de cuestiones que han quedado fuera del alcance del proyecto pero que podrán ser tratadas con posterioridad.

Al final del documento, en los anexos, se encuentra disponible un glosario (Apéndice A) de términos que puede ser consultado para una mejor comprensión de los conceptos que se tratan en este texto. Los conceptos desarrollados en el glosario serán marcados con un asterisco (*) al aparecer por primera vez en el documento. Por último, se adjunta una reflexión acerca de la relación del Trabajo de Fin de Grado con los Objetivos de Desarrollo Sostenible de la Agenda 2030.

CAPÍTULO 2
Estado del arte

Construir una red de estaciones de recarga de vehículos eléctricos a nivel nacional no es una tarea trivial. Sobre todo, lo más complicado es diseñar la red de tal manera que se pueda dar servicio a todos los conductores posibles, permitiendo encadenar paradas entre varias estaciones para hacer viajes más largos, pero con el menor número de estaciones construidas en aras de minimizar costes. En el mejor de los casos, se dispone de suficiente presupuesto para colocar estaciones cada pocos kilómetros, como las gasolineras en las carreteras españolas. Sin embargo, los recursos suelen estar limitados y las estaciones suelen ser muy caras de construir, mantener y alimentar con energía eléctrica. Fruto de este problema han surgido diversas investigaciones para averiguar qué opción es más favorable a la hora de ubicar estaciones en cualquier mapa. A continuación vamos a ver en qué punto se encuentran estos trabajos.

2.1 Trabajos previos

Uno de los estudios que se han tenido en cuenta para llevar a cabo este proyecto ha sido realizado por un grupo de investigación del *Institut Valencià d'Investigació en Intelligència Artificial* (VRAIN[1]) empleando inteligencia artificial. En una primera fase, Jordán et al deciden desarrollar un algoritmo genético* que, dado el mapa de una ciudad, calcule cómo debe distribuirse cierto número máximo de estaciones de manera óptima [6]. Las decisiones del algoritmo no debían basarse simplemente en el mapa de la ciudad o en minimizar la distancia entre estaciones, sino que debía recibir información demográfica y de tráfico real para aproximarse lo más posible a la realidad del municipio. Los factores que recibe el algoritmo genético como entrada son los siguientes:

1. La distribución de la población por barrios, es decir, el número de habitantes por cada zona del mapa.

2. Las calles más transitadas y con más tráfico.

3. Las zonas de la ciudad con más actividad social. Esto se mide contabilizando las publicaciones a las redes sociales Twitter e Instagram, agregando por barrio donde se ha enviado el tweet o post de Instagram.

4. Las ubicaciones candidatas a albergar una estación de recarga. Esto es importante para simplificar el funcionamiento del algoritmo, pues de no ser así habría infinitas opciones para distribuir las estaciones. Así, se reduce la cantidad de opciones, obteniendo una solución subóptima pero suficiente. Además, se garantiza que las

[1] por sus siglas en inglés, *Valencian Research Institute for Artificial Intelligence*

estaciones no se colocan en ubicaciones no preparadas para ello o en lugares poco accesibles, por ejemplo, en un túnel.

5. Se mide la popularidad de cada punto de interés, medida por el tiempo que pasan los individuos en las zonas cercanas al mismo, por ejemplo, un centro comercial, un pabellón de deportes etc. Estos datos se obtuvieron de motores de búsqueda como Google.

Una de las premisas del proyecto era obtener el algoritmo de tal manera que pudiera ser empleado para cualquier ciudad, permitiendo la introducción manual de los datos mencionados. Es sencillo obtener el mapa de la ciudad a estudiar y la información enumerada puede encontrarse en redes sociales y portales públicos de estadística y tráfico, el Instituto Nacional de Estadística etc. Por tanto, no llevaría mucho tiempo cambiar la ciudad bajo estudio con un algoritmo universal.

A la hora de ejecutar el programa desarrollado, dichos datos se procesan de la siguiente manera (ver figura 2.1): En la primera fase, el algoritmo selecciona los puntos de interés que va a tener en cuenta, que forman un subconjunto de los puntos que venían de entrada. En la segunda fase se establece el diagrama de Voronoi* de la ciudad, que la divide en zonas de influencia. En cada zona de influencia se encuentra un único punto de interés. Dicha zona representa los puntos del mapa cubiertos por dicho punto de interés, es decir, desde los que se puede llegar a él de manera más rápida. Seguidamente, en la tercera y cuarta fase, respectivamente, se recoge la información demográfica y, para cada zona de influencia de los puntos de interés, se agregan los datos y se establece su población, nivel de tráfico, actividad social y popularidad.

La quinta fase consiste en la introducción de todos los datos recabados en el algoritmo genético en sí, denominado agente optimizador de emplazamientos. El agente selecciona inicialmente una distribución aleatoria de puntos de interés para colocar las estaciones. Para evaluar la red de estaciones que se acaba de producir, el algoritmo cuenta con una función de utilidad que asigna un valor numérico a cada red de estaciones en función de lo bien que se ajuste a la situación demográfica de la ciudad. Cuanto mayor el resultado de la función, más beneficio proporciona a la población, es decir, las estaciones se encontrarán en lugares más habitados, transitados, populares y socialmente activos (en orden descendente de importancia). Se tiene en cuenta el coste total de la nueva infraestructura, que influye negativamente en el cálculo.

Finalmente, el algoritmo entra en la fase de reproducción, con sucesivos cruces y mutaciones. Cabe mencionar que se incluyó una modificación en el procedimiento de cruce, dado que se detectó que los puntos de interés geográficamente próximos comparten las mismas cualidades (tráfico, popularidad...). Por consiguiente, tiene sentido que dos cromosomas, al cruzarse, intercambien subconjuntos de puntos de interés que se encuentran geográficamente juntos en vez de coger estaciones alejadas, que no tienen por qué ser similares.

Los experimentos realizados para poner a prueba este algoritmo se hicieron sobre el mapa de la ciudad de València, obteniendo como solución una red óptima (maximizada la función de utilidad) de 42 estaciones de recarga distribuidas por el municipio y algunas de sus pedanías (ver figura 2.2).

Posteriormente se decidió expandir el mapa a nivel nacional [5], pues como se ha comentado con anterioridad, el algoritmo se desarrolló de manera que pudiera ser reutilizado para cualquier región. Evidentemente, el territorio español tiene muchas veces más lugares donde colocar unas hipotéticas estaciones de recarga que en una ciudad como València. En consecuencia, la carga computacional del algoritmo si considera todas las coordenadas del mapa puede ser enorme e impracticable. Se decidió, por tanto,

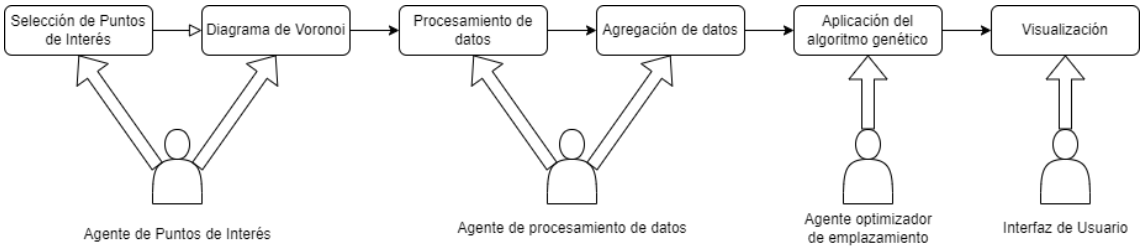


Figura 2.1: Diagrama de flujo de la ejecución del algoritmo genético

Fuente: [6]. Elaboración propia

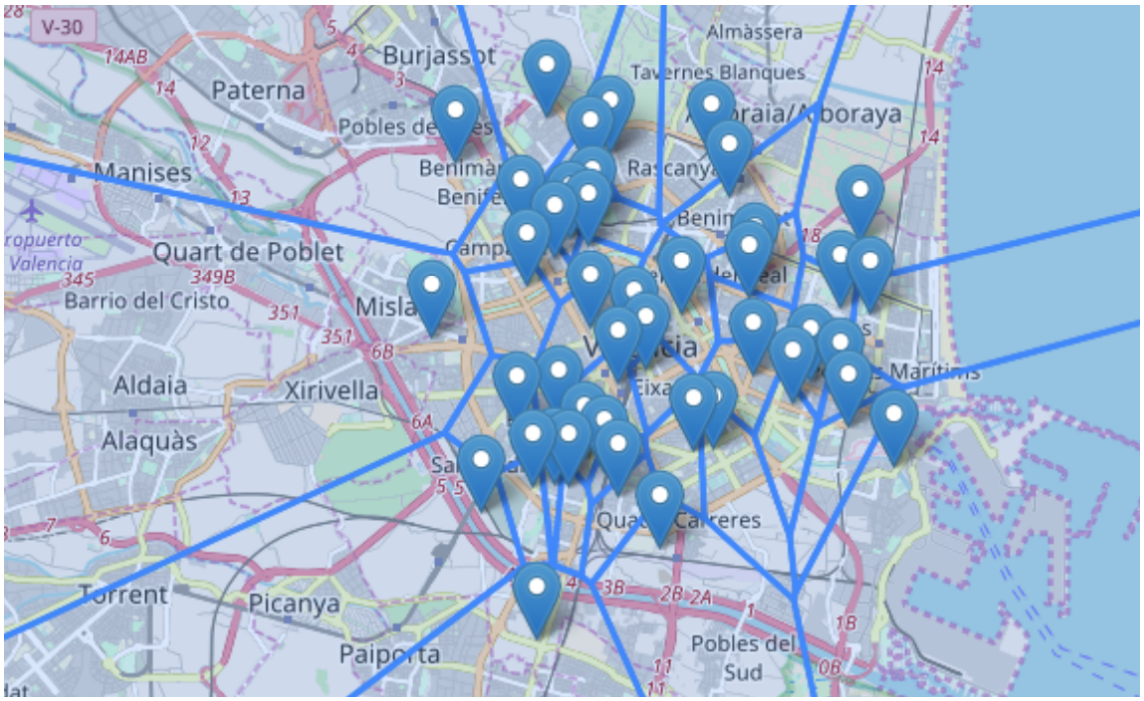


Figura 2.2: Solución óptima del algoritmo genético para el mapa de la ciudad de València

Fuente: [6]

Estaciones	Utilidad	Distancia máx. (km)	Territorio sin cubrir
50	0.00238	113.7	44.4 %
100	0.00460	101.1	20.8 %
150	0.00735	89.5	10.4 %
200	0.01495	82.9	5.1 %
250	0.01964	59.2	1.9 %
300	0.02986	60.3	1.2 %
400	0.04417	58.9	0.7 %
500	0.05219	43.8	0.3 %
750	0.10874	35.0	0.02 %
1000	0.14614	31.1	0.003 %

Tabla 2.1: Resultados de los experimentos previos a nivel interurbano

Fuente: [5], elaboración propia

simplificar el problema considerando exclusivamente la red de gasolineras existente en España como puntos de interés. Teniendo presente que todas ellas cuentan con una instalación de acceso a la red eléctrica, esta simplificación cobra mucho sentido, pues se puede aprovechar dicha conexión para los cargadores de coches. Cabe destacar que el voltaje disponible probablemente no sea suficiente para alimentar un cargador de coches y que por tanto habría que instalar transformadores para aumentarlo hasta un nivel que permita varios kilovatios de potencia.

Reducir el número de soluciones posibles hasta los conjuntos de gasolineras existentes en España provoca que la solución generada deje de ser una solución óptima, aunque se considera perfectamente válida por razones económicas y estructurales, como se ha mencionado en el párrafo anterior. La bondad de la disposición de estaciones fue medida esta vez por el porcentaje de territorio que tenía efectivamente acceso a una estación de recarga, la distancia máxima entre dos estaciones contiguas (a minimizar) y una función de utilidad muy similar a la empleada en el entorno de València. El resultado fue que con unas 250 estaciones se puede construir una buena infraestructura de recarga a nivel interurbano, dejando sin cobertura un 2 % de suelo español, aproximadamente. Es cierto que aumentando aún más el número de cargadores se sigue reduciendo la superficie sin accesibilidad a las estaciones, sin embargo, ya no disminuye proporcionalmente, por lo que resulta ineficiente en términos económicos. En otras palabras, colocando más estaciones, no se consigue aumentar de manera tan significativa el porcentaje de suelo desde el que se puede alcanzar una estación de recarga, por tanto, invertir en más estaciones no retorna un beneficio rentable. En todo caso, la distancia máxima entre dos estaciones ascendió en este caso a algo menos de 60km (ver resultados completos en la tabla 2.1).

2.2 Tecnologías y simuladores

Una vez terminada la fase de generación de las distribuciones ideales de puntos de recarga, se tiene que proceder a testearlas con pruebas representativas de la realidad. Como se ha mencionado al comienzo de este texto, no es factible construir una red provisional de estaciones y probarla con conductores reales, porque requeriría de mucho tiempo y dinero. Para realizar pruebas sin tener que movilizar ningún recurso, necesitamos un simulador que se adapte a las circunstancias de nuestro estudio o que pueda ser ampliado con facilidad para soportar los experimentos, de manera que las simulaciones sean lo más fieles a la realidad posible.

Según el diccionario de la Real Academia Española, un simulador se define como un «aparato que reproduce el comportamiento de un sistema en determinadas condiciones, aplicado generalmente para el entrenamiento de quienes deben manejar dicho sistema». En otras palabras, se trata de una herramienta que puede emplearse para modelar la realidad, aunque basta con plasmar aquellas partes de la realidad que nos interesan para experimentar. En nuestro caso, podemos centrarnos en el mapa español y su red de carreteras nacional, sin tener en cuenta el resto de países, la meteorología etc. La descripción hace alusión a la palabra "condiciones". Cada escenario de simulación se tiene que poder configurar de tal manera que las estaciones se ubiquen en función de una distribución concreta por todo el territorio y los vehículos puedan acceder a ellas cuando sea pertinente. Los vehículos también deben poder variar sus parámetros, como la autonomía máxima e inicial etc.

Siguiendo con la definición de la R.A.E., nuestro simulador no estará diseñado para entrenar a ninguna persona. A pesar de ello, en cierto modo sí que se va a utilizar de manera similar a un software de entrenamiento, puesto que se van a recabar múltiples datos relativos al transcurso de la simulación y evaluar los resultados de manera crítica, puntuando la bondad de cada una de las distribución de estaciones, como si se evaluara el examen de un aprendiz.

2.2.1. Simulación multiagente

La tecnología por excelencia para simular el comportamiento de un número masivo de entes es la simulación multiagente. Como plasma García-Valdecasas en [7], la simulación basada en agentes se define como «un método informático que permite construir modelos constituidos por agentes que interaccionan entre sí dentro de un entorno para llevar a cabo experimentos virtuales» [8]. En esta definición aparecen conceptos de especial relevancia, como "agente", "interacción" y "experimento virtual".

En el contexto de los simuladores multiagente, un agente propiamente dicho es una unidad de computación individual [8] que representa un actor principal de la simulación [7]. Es capaz de razonar y tomar sus propias decisiones de manera autónoma reaccionando a la información que percibe de su entorno, aunque siempre siguiendo unas determinadas reglas establecidas. Durante su vida colabora con otros agentes comunicando su estado o situación para lograr un objetivo individual o común, también fijado de antemano [9]. El hecho de que cada sujeto sea una unidad independiente y autónoma supone la principal ventaja de este tipo de simuladores: está muy enfocado hacia el diseño de los agentes que aparecen en el simulador como objetos individuales, normalmente homogéneos entre sí. Esto quiere decir que, aunque los agentes tengan una estructura interna compleja, estos sistemas permiten centrarse en el modelado de sus características y cómo deben comportarse a la hora de interactuar con el entorno y demás agentes. Si es pertinente durante la simulación, pueden introducirse tantas copias o instancias de dicho objeto como se desee. Resulta sencillo e intuitivo poder encapsular* sus características internas, dejando todo el control de la comunicación al agente durante la ejecución de las simulaciones.

A modo de definición formal, seguiremos la propuesta de Gilbert en [8]:

$$A \sim (S, R); S = \{S^1, S^2, ..., S^k\}; R : (S_t, I_t) \rightarrow S_{t+1}$$

donde A es el agente en forma de autómata, que posee un estado S y unas reglas de transición entre estados R. El conjunto de los k estados posibles es finito y está definido de antemano. En función de la situación del agente, es decir, del valor de sus atributos*, se encontrará en un estado u otro. Las reglas de transición R simbolizan los requisitos que

tienen que cumplir agente y entorno para pasar al siguiente estado. Toman como entrada el estado actual del agente S_t con sus atributos e información recibida de otros agentes I_t. Una vez disponible toda la información necesaria y satisfechas todas las condiciones, se aplica la regla de transición correspondiente, y el agente pasa a encontrarse en el estado S_{t+1}.

En relación a la "interacción" entre agentes, es completamente esencial que dispongan de información suficiente sobre las características de su entorno y la situación de otros agentes para que puedan tomar decisiones y transicionar entre estados [7]. Como se ha mencionado en la introducción de este apartado, los agentes colaboran entre sí como unidades de computación independientes, por lo que también se comportan de manera ajena a los demás agentes. Siendo esto así, los agentes no pueden detenerse a esperar mensajes de los demás, igual que en la comunicación humana, sino que deben transmitir y escuchar mensajes sobre la marcha. Es necesario disponer de una herramienta adicional de mensajería para manejar este proceso. La más ideal para esta situación podría ser un servidor XMPP (Extensible Messaging and Presence Protocol), que sirve como medio de envío y recepción de consultas y respuestas entre agentes.

XMPP es un protocolo estandarizado por el *Internet Engineering Task Force* (IETF) y de código abierto*, cuyas implementaciones son gratuitas y a su vez de código abierto, fácilmente accesibles desde internet. Los mensajes que se envían a través de esta plataforma están estructurados según el formato XML (*Extensible Markup Language*)*[10]. Las principales ventajas de este sistema es que permite la comunicación según el esquema *request-response*, por el que un agente envía información (*response*) solamente cuando se le ha solicitado (*request*); y la comunicación asíncrona, es decir, que el receptor no tiene que encontrarse constantemente a la escucha de mensajes nuevos. Como los servidores XMPP se encuentran descentralizados*, los mensajes se direccionan directamente entre agentes, sin pasar por un servidor central. Por tanto, cada usuario registrado en el servidor contiene un *buffer** a modo de bandeja de entrada y un identificador único (JID) para poder dirigir los mensajes de manera precisa y sencilla. Así se logra la mencionada asincronía, dado que si el receptor no está conectado en el momento del envío del mensaje, se almacena temporalmente en memoria. Similar a lo que ocurre con los correos electrónicos (servidores POP3, por ejemplo) y, en general, en todo proyecto de *Internet of Things** con una cantidad masiva de sensores [11]. Así, volviendo al contexto multiagente, cuando un agente se encuentra realizando una rutina y recibe una comunicación por XMPP, no tiene que interrumpirla de manera brusca para procesarla, puede contestar en un momento posterior. Desde el punto de vista del emisor del mensaje, normalmente este sí tiene que esperar la contestación, pero durante un tiempo máximo tras el que se da por ausente al destinatario y se lanza un error. Así, se evita que el primero se quede esperando eternamente y produzca un interbloqueo*. Como los mensajes siguen el formato estructurado XML, resulta muy sencillo extraer la información del mensaje para ser computada por un agente.

Por último, en la definición se hace mención al concepto de "experimento virtual". Durante un proyecto de desarrollo de simulaciones multiagente, a la hora de experimentar, en primer lugar hay que representar los agentes y definir cómo deben compartir información. En segundo lugar, a la hora de ejecutar la simulación, se debe configurar los parámetros de cada tipo de ellos. Por último, se recogen los resultados en forma de información medible. Las dos primeras fases permiten modelar el experimento que se quiere realizar como se desee, a ser posible de manera lo más realista posible. Concretamente, la segunda fase es la más flexible, pues se puede probar con distintas configuraciones para los objetos y lanzar simulaciones para cada de una de ellas, pudiendo comparar los resultados de manera muy sencilla. Por ejemplo, en [12] se ejecutan varios experimentos sobre el modelo de segregación étnica de Tomas Schelling, variando únicamente la "preferencia

de grupo". La repetición de estos experimentos es muy sencilla, pues el modelado de los individuos (primera fase) solo se tiene que realizar una única vez, mientras que el cambio de parámetros (preferencia de grupo) se puede hacer, de manera rápida y sencilla, tantas veces como se desee (segunda fase). La intención final es observar las diferencias en los resultados obtenidos (tercera fase), compararlos e interpretar la razón por la que difieren. Todo este proceso ocurre de manera virtual, no física. Es decir, en ningún momento es necesario movilizar ningún recurso físico. La simulación es, en cierto modo, "gratuita".

Para realizar estos experimentos basados en sistemas multiagente se necesita un software que permita ejecutar (y quizá visualizar) los experimentos. A continuación vamos a analizar los simuladores existentes que pueden servirnos para llevar a cabo nuestro proyecto.

2.2.2. Simuladores

Antes de analizar los programas en sí, debemos realizar una distinción entre los tipos de simuladores de tráfico que existen entre los que se han barajado para el desarrollo del proyecto. Los dos más relevantes son los simuladores basados en modelos microscópicos y macroscópicos. Se distinguen en el enfoque de los simuladores y las métricas que miden. Los simuladores basados en modelos microscópicos se concentran en lo que ocurre al vehículo en todo momento, su situación, velocidad, daños sufridos, el comportamiento del conductor etc. Normalmente, recoger mucha información de un número elevado de vehículos es computacionalmente muy costoso, por lo que los vehículos, que suelen ser reducidos en número, se mueven en un entorno pequeño, por ejemplo, una intersección, una glorieta, un paso a nivel ferroviario etc. En cambio, los simuladores basados en modelos macroscópicos recaban información general de tráfico y datos limitados de los coches, pudiéndose analizar el recorrido y otras métricas globales de los vehículos para un espacio de movimiento mucho más grande (una ciudad, provincia, un país entero...).

Además de los tipos de simuladores citados, existen los que se basan en modelos mesoscópicos, modelos que se encuentran en la frontera entre los microscópicos y los macroscópicos; y los que se basan en modelos submicroscópicos, en los que se tienen en cuenta hasta los más mínimos detalles de los agentes a simular. En estos casos, los vehículos o entes que forman parte de la simulación se dividen en sub-estructuras (motor, caja de cambios...) con las que se puede observar el comportamiento del agente después de sumar el comportamiento de cada sub-estructura. Es decir, la configuración del vehículo global depende de la configuración de cada parte individual. Se trata del modelo de simulador más exigente en términos de procesamiento y memoria, pues cada agente puede dividirse en múltiples partes y cada uno consume muchos recursos [13].

Eclipse SUMO

Un ejemplo de simulador basado en modelos microscópicos es Eclipse SUMO[2]. Las siglas significan *Simulation of Urban MObility*, y como consta en el nombre completo, se desarrolló en el notorio entorno de desarrollo Eclipse. En pocas palabras, Eclipse SUMO permite modelar un circuito de tráfico, real o ficticio, por el que se hace pasar una cantidad determinada de coches configurados de antemano. Los modelos son microscópicos porque, en general, los coches se comportan de manera aislada, se mueven según su propia ruta y están modelados detalladamente y de manera individual.

El comienzo de su creación data del año 2000, con la intención de proporcionar a la creciente comunidad de analistas de tráfico una herramienta para poder realizar sus in-

[2] www.eclipse.org/sumo/

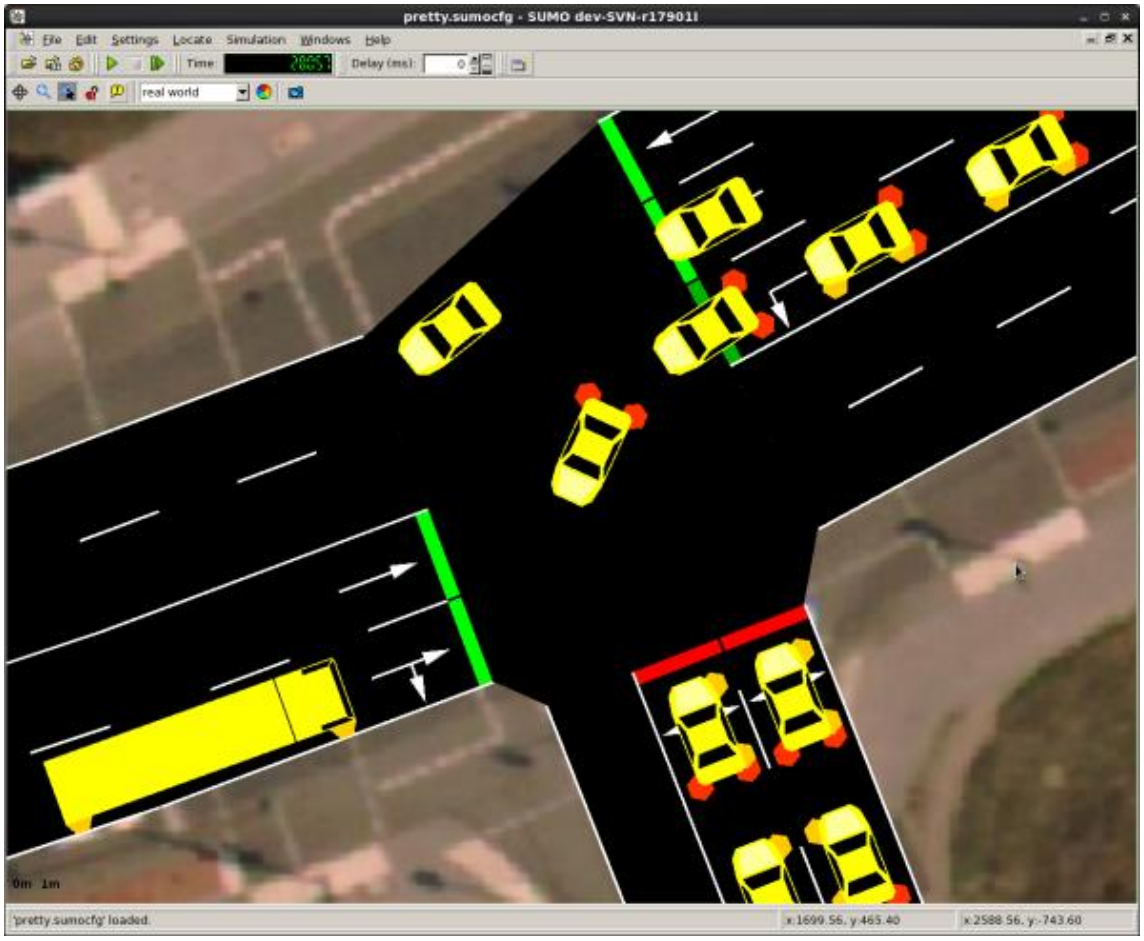


Figura 2.3: Captura de la interfaz gráfica de Eclipse SUMO.

Obtenido de www.openmobility.eclipse.org/technologies/eclipse-sumo

vestigaciones de manera más sencilla y estandarizada. Esto es, se pretendía construir un programa que permitiera realizar simulaciones sin tener que desarrollar código adicional para manejar el tráfico, el comportamiento de los vehículos y de las señales de tráfico etc.; y que además sirviera como herramienta estándar de comparación entre proyectos, al compartir todos ellos exactamente la misma infraestructura subyacente. Estamos hablando de un software de código abierto, cualidad que facilita su distribución al ser fácil de encontrar y descargar por todo el mundo. Además, esto también posibilita que cualquier programador desarrolle nuevas funciones y las haga, asimismo, accesibles en internet [14].

Las premisas de los fundadores en la creación de Eclipse SUMO respecto al producto final era que fuera portable, rápido y eficiente. La portabilidad implica que pueda ser ejecutado en distintos sistemas operativos sin necesidad de ajustes adicionales (por ejemplo, máquinas virtuales). En cuanto al intento de maximizar la eficiencia, ello conllevó en los primeros años a que Eclipse SUMO sacrificara su interfaz visual, teniendo que ser ejecutado desde consola. La interfaz, al comienzo, hubiera implicado una ralentización de las simulaciones que se quería evitar; aunque más adelante, con el producto más avanzado y optimizado de manera conveniente, se pudo acoplar. En la figura 2.3 podemos ver cómo se dispone la interfaz gráfica del simulador.

Adicionalmente, una característica de este programa que no poseen otros simuladores similares es que es completamente descentralizado. Significa que, en vez de ejecutarse desde un solo proceso en una única ventana, cada módulo software incluido en el paque-

te de Eclipse SUMO es independiente de los demás, y en función de las necesidades de cada simulación, se puede ejecutar o no. Esto posibilita que cada usuario pueda regular su simulación en términos de velocidad de ejecución y funciones empleadas; y que los desarrolladores puedan encapsular perfectamente el funcionamiento de cada módulo. Evidentemente, que el simulador esté tan desperdigado provoca que sea más difícil de entender y de utilizar. Se entiende que el simulador tiene una curva de aprendizaje muy extendida, pero puede resultar extremadamente beneficioso si se sabe manejar bien [14].

En su mayoría, las aplicaciones de este simulador tienen que ver de una manera u otra con la predicción de tráfico en un circuito cerrado. Es decir, se puede simular cómo se desarrollaría el tráfico si se varía el ciclo de un semáforo (tiempo que pasa entre dos luces rojas), se añadiera un nuevo carril en una calle, se sustituyera un cruce por una glorieta. Por ejemplo, el proyecto VITAL[3] del *Deutsches Zentrum für Luft- und Raumfahrt* (Centro aeroespacial alemán) investiga la mejora del tráfico mediante la implantación de algoritmos inteligentes que regulen el cambio de señal en los semáforos [15]. Los dos algoritmos estudiados son el *Traffic-Actuated Control of Traffic Lights* o "control de señales de tráfico activadas por tráfico" y otro que es denominado GLOSA[4]("recomendación de velocidad óptima bajo luz verde").

El primero de ellos calcula la congestión del tráfico en función del tiempo que tarda en recorrer cada vehículo los últimos metros anteriores a una señal, y ordena un cambio de color del semáforo según las necesidades observadas. En otras palabras, en caso de detectar mucho atasco, se mantiene el color del semáforo en verde, en caso contrario se cambia a rojo; pero siempre dentro de un umbral temporal determinado (mínimo y máximo) para evitar monopolizar el tráfico.

En el segundo, el semáforo se comunica directamente con los coches, comunicando el tiempo restante de "rojo". El vehículo calcula a su recepción la velocidad a la que se debe circular para evitar tener que detenerse completamente. La comunicación es bidireccional: los coches comunican su posición a los semáforos para que puedan cambiar de ciclo cuando sea más oportuno. Como, sobre todo en el caso de GLOSA, son muy contados los vehículos y señales de tráfico que disponen de esta tecnología de comunicación V2I (vehículo-infraestructura), en el estudio mencionado se realiza una simulación en Eclipse SUMO para estudiar la mejora teórica del tráfico bajo la implantación de esta tecnología de comunicación entre dichos agentes de tráfico. La conclusión del estudio es que las versiones simuladas de ambos algoritmos mejoran la fluidez del tráfico entre un 15 % y un 28 % [15] respecto al tráfico detectado en una intersección real.

En el caso de VITAL, el alcance espacial del proyecto no abarca más allá de una misma intersección y se requiere información concreta de la velocidad, posición y trayectoria de cada vehículo de manera individual. Todo hace indicar que un simulador basado en modelos microscópicos es el más indicado para someter el cruce de calles a todas las pruebas que hagan falta. Sin embargo, los autores indican en [15] que podría resultar altamente interesante ampliar el horizonte de estudio hacia un circuito de tráfico más grande, con varias intersecciones coordinadas entre sí.

Como se ha mencionado anteriormente, cualquier atributo que concierna a vehículos, peatones, la calzada etc. puede ser modelado. Eclipse SUMO no se limita exclusivamente al estudio de la regulación de tráfico mediante señales fijas o luminosas. Es más, los vehículos de cuatro ruedas no son los únicos que pueden representarse, sino también peatones, bicicletas etc. Sirviendo como ejemplo, en [16], Gutiérrez lleva a cabo un proyecto para poner a prueba estos agentes de tráfico alternativos al coche. Concretamente, analiza el impacto que tiene el comportamiento de los conductores en ciudad sobre la circulación

[3] *Vehicle-Actuated Intelligent Traffic Signal Control*
[4] *Green Light Optimized Speed Advisory*

de usuarios vulnerables, que son aquellos que carecen de escudo protector: peatones, ciclistas y motoristas. Para ello establece tres escenarios distintos sobre un mapa del barrio madrileño de Aluche, sobre el que hace circular coches y usuarios vulnerables al mismo tiempo, midiendo la distancia entre ellos y la suma de sus velocidades:

1. Comportamiento responsable de los vehículos: todos los vehículos respetan la velocidad máxima de 5 m/s.
2. Comportamiento agresivo de los vehículos: algunos vehículos pueden sobrepasar el límite máximo hasta los 20 m/s (coches) o 30 m/s (motocicletas).
3. Comportamiento moderado de los vehículos: algunos vehículos pueden sobrepasar el límite máximo hasta los 10 m/s (tanto para vehículos como para motocicletas).

Cabe destacar que las calles virtuales de Aluche han sido configuradas previamente para permitir adelantamientos, el cruce de peatones por pasos de cebra etc. La cantidad y las rutas de los vehículos (en este último caso, aleatorias), son generadas de antemano y fijadas de manera constante para todos los escenarios, de manera que las métricas a extraer de las simulaciones se puedan comparar de manera inmediata. Todo con el objetivo de que el modelo microscópico quede lo más ceñido posible a la realidad.

Las simulaciones de Gutiérrez arrojan un resultado que a priori parecía evidente, pero que queda demostrado de manera empírica: cuanto mayor es la velocidad de coches o motocicletas, mayor es el riesgo de que se produzca un accidente en el que se encuentre involucrado un usuario vulnerable [16]. El análisis de las tres situaciones demuestra que cuando la velocidad de los vehículos es alta, se producen más situaciones en las que la distancia entre ellos se reduce a niveles muy bajos, incurriendo en el riesgo de colisionar y provocar un accidente, que en el caso de los usuarios vulnerables puede suponer lesiones de gravedad o incluso la muerte. Por tanto, es conveniente que todos los conductores urbanos hagan un esfuerzo por mantener los límites de velocidad, en aras de proteger a las personas que circulan desprotegidas.

Nuevamente, el uso de Eclipse SUMO para realizar un estudio empírico sobre este asunto se justifica por las características del problema. Por mucho que el mapa donde se sitúan los experimentos sea mayor que en el caso de VITAL, resulta crucial poder modelar de manera realista y precisa los agentes de tráfico que transitan por él. Esta herramienta, como ya se ha comentado, al ser multimodal, dispone de las figuras que se requieren para representar vehículos de dos ruedas o personas de a pie. Estas figuras incluyen su comportamiento propio, es decir, diferencia el modo en que se conduce cada tipo de vehículos y las acciones que pueden realizar. Por ejemplo, para los vehículos de dos ruedas, adelantar una fila de coches cuando estos se encuentran estacionados ante un semáforo para colocarse en primera posición.

MATSim

MATSim[5], por sus siglas en inglés *Multi-Agent Transport Simulator*, es un simulador de código abierto. Como su nombre indica, sigue el esquema de la simulación multiagente. El inicio de su desarrollo data del año 2006 en la ETH Zúrich con la idea de crear un programa para simular condiciones de tráfico en determinadas regiones [17]. Al ser un simulador multiagente, permite la experimentación de múltiples agentes de tráfico a gran escala. Está programado en Java, aunque en sus inicios se implementó en el lenguaje C++.

MATSim forma parte del conjunto de microsimuladores, pues se centra en cada agente individual y sus métricas particulares. Es capaz de modelar una jornada de tráfico

[5] https://www.matsim.org/

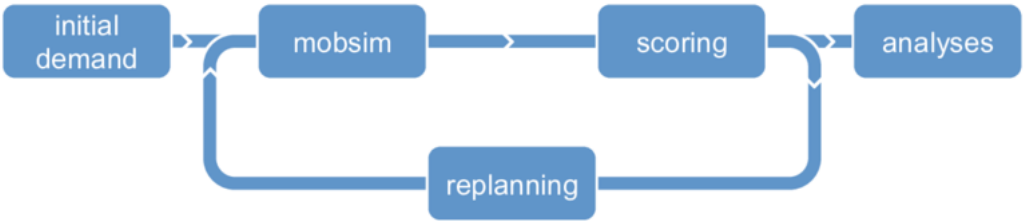


Figura 2.4: Ciclo iterativo de ejecución en una simulación de MATSim (*MATSim cycle*)

Fuente: [17]

rutinaria en una determinada zona, con agentes destinados al transporte de particulares, el transporte público, etc.

Los agentes de MATSim siguen un esquema iterativo para optimizar sus resultados durante una simulación. En la figura 2.4 podemos ver como la ejecución comienza con la inicialización de la demanda inicial de agentes. En cada iteración del ciclo, se realiza una ejecución del experimento. De este experimento se recogen los resultados, y dependiendo del número de repeticiones del ciclo que se haya configurado, se vuelve a iniciar el ciclo o no. Antes de repetir el experimento, los agentes planifican su comportamiento en función de la información que han recabado en el anterior intento. De esta manera, cada repetición de la simulación da resultados más óptimos. El nivel de optimalidad crece muy rápidamente en las primeras repeticiones [17].

Por ejemplo, en el caso del transporte público, un vehículo comenzaría recorriendo su ruta por cierto camino inicial. En función de cómo haya transcurrido el viaje durante una simulación (puede haber encontrado tráfico), el transporte puede decidir cambiar de ruta para acelerar su trayecto. Si se tratase de simulaciones de un servicio de mercancías, los camiones planifican de antemano el recorrido que van a seguir para repartir los bienes. En cada iteración del ciclo, repetirán la planificación corrigiendo la ruta para que esta sea lo más corta y directa posible, evitar zonas congestionadas con tráfico etc. Por tanto, esto podría ser útil a la hora de que unas hipotéticas flotas de vehículos eléctricos calculen la ruta a seguir para viajar a otra ciudad, optimizando el viaje para poder llegar a destino con el menor número de recargas o con el menor desvío de la ruta óptima posible.

En este aspecto, son varios los estudios en los que se ha empleado esta herramienta para probar situaciones de tráfico en las que se ven involucrados los vehículos eléctricos. Por ejemplo, en la ciudad de Zúrich (ciudad de origen de MATSim), se planteó cómo afectaría al tráfico de coches electrificados la futura llegada de estaciones de recarga de bajo voltaje. En las investigaciones llevadas a cabo por Waraich et al. en [18] se quería introducir puntos de recarga de coches eléctricos en una representación de Zúrich para descubrir si se generaban cuellos de botella en alguna de ellas y qué implicaciones tendría en el tráfico de la ciudad. Para poder realizar sus ensayos, escogieron MATSim y ampliaron uno de sus módulos para representar flotas de coches eléctricos. De hecho, este nuevo módulo fue incorporado al paquete del simulador para su uso por otros usuarios.

Y es que una de las características muy positivas de MATSim es, al igual que Eclipse SUMO, su programa es de código abierto y está dividido en varios módulos distintos. Esto facilita, como demostraron Waraich et al. [18], la edición del simulador para introducir nuevos agentes, cambiar su comportamiento, añadir módulos etc.; por parte de cualquier usuario del mundo.

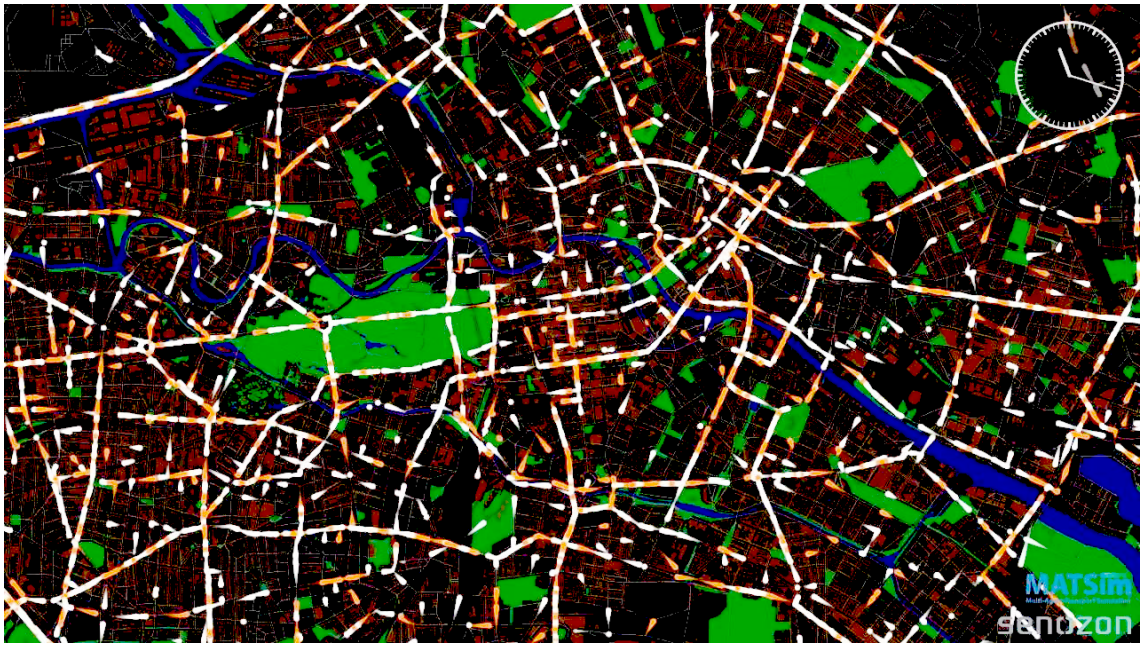


Figura 2.5: Captura de la interfaz de MATSim en la que se observa el flujo de tráfico durante un experimento sobre Zúrich

Fuente: *Institute for Transport Planning and Systems, ETH Zürich*

MATSim dispone de una interfaz gráfica muy visual e intuitiva para poder ver y analizar el flujo de tráfico. A modo de ejemplo, se incluye la figura 2.5 en la que se puede ver una captura del mapa de Zúrich durante una simulación de tráfico.

AnyLogic

La siguiente alternativa como herramienta de experimentación virtual es AnyLogic[6]. Es un programa de simulación que se emplea en una lista muy variada de sectores, desde el ferroviario hasta el minero y el de la distribución. Se diferencia de los otros simuladores en que, en este caso, es un software de pago propietario de la empresa The AnyLogicCompany. Como es un producto comercial ni es de código abierto ni es gratuito, por lo que hay que adquirir una licencia para poder utilizarlo. Evidentemente, tampoco se puede modificar. Lo que sí es un aspecto positivo es que proporciona varias tecnologías distintas para llevar a cabo los experimentos. Las tres tecnologías que incluye son la simulación multiagente, la dinámica de sistemas y sistemas de eventos discretos[7].

Para ilustrar el amplio abanico de opciones que ofrece el simulador, tenemos el caso que presentan Yang et al. en [19], en el que diseñan unos experimentos para simular la cola de los puntos de venta de abonos en el metro de Hangzhou. El propósito de esta investigación era recoger información acerca de la mejor planificación para la venta de abonos en una estación del metro en función de una serie de escenarios. Los autores hacen uso de la simulación basada en agentes para realizar sus experimentos. Otro caso de uso es el que mencionan Merkuryeva y Bolshakovs en [20], donde aplican un sistema de eventos discretos a la planificación temporal de trayectos de un medio de transporte.

Vemos pues que, por mucho que no se tenga acceso al código del simulador para realizar modificaciones, se dispone de una gran cantidad de configuraciones que cubren todo tipo de casuísticas. El asunto que atañe a este trabajo, la gestión de coches eléctricos en

[6] www.anylogic.com
[7] /www.anylogic.com/features

viajes interurbanos, no es una excepción. AnyLogic proporciona un módulo para realizar experimentos sobre problemas de gestión de flotas. En el ejemplo que se enlaza desde la página web de AnyLogic[8] se incluye un escenario de gestión del transporte marítimo de mercancías. El modelo se puede ejecutar de manera interactiva, observando cómo se desplazan los barcos de un puerto a otro y las estadísticas finales.

SimFleet

Por último tenemos SimFleet[9][21], un macrosimulador también basado en tecnología multiagente. Fue desarrollado por investigadores del VRAIN[10] y publicado por primera vez en 2017. El simulador también se encuentra disponible en internet, en plataformas de desarrollo colaborativo como GitHub. Por tanto, estamos frente a una herramienta de código abierto, habilitado para su descarga y manipulación por todo el mundo.

Según el VRAIN, la aparición de nuevos paradigmas en el sector del transporte motivaron la creación del simulador. A diferencia de otros similares como MATSim y Eclipse SUMO, el enfoque de este simulador debía estar puesto en la experimentación sobre estos nuevos métodos de transporte, tanto de bienes como de personas, pero a nivel global.

A modo de ejemplo, dos de estos paradigmas que justificaron su implementación son la "Cadena de Suministro Colaborativa" y el "*car-sharing*". El primero de ellos es una evolución de la cadena de suministro tradicional en la que los distribuidores de la cadena comparten información y se coordinan para realizar sus tareas de manera conjunta. Un repartidor que sigue este modelo no realiza su labor por cuenta de una sola empresa, sino que transporta envíos de varias de ellas al mismo tiempo. De esta manera, las empresas que trabajan conjuntamente comparten gastos y aumentan su eficiencia, reduciendo el nivel de inventario y aprovechando mejor la capacidad de carga de los camiones, que además recorren menos distancia [22]. En el segundo caso, más cotidiano que el anterior, dos o más particulares se ponen en contacto para realizar un viaje juntos, llenando coches que en otro caso circularían con pocos pasajeros y reduciendo el gasto en combustible y la contaminación. Es un modelo de transporte muy utilizado por la comunidad universitaria cuando los estudiantes viven fuera del alcance del transporte público del municipio donde se encuentra el centro docente, o por viajeros interurbanos en general [23]. Una de las aplicaciones más conocidas y extendidas para poner en contacto a los usuarios es la aplicación BlaBlaCar[11].

Campuzano et al. mencionan en su estudio sobre cadenas de suministro colaborativas que surge la necesidad de poner a prueba este modelo logístico y cuantificar el ahorro que se produce [22]. Este tipo de planteamientos, insistimos, son el origen de SimFleet. Por tanto, se construyó teniendo en mente desde el principio que su objetivo era la representación de agentes de transporte de personas o mercancías. En ese sentido, se decidió implementarlo como un simulador multiagente, que permite el modelado aislado de cada figura a participar en los experimentos. Cada figura se traduce, evidentemente en un agente, que debe tener una estrategia determinada. Esta estrategia puede ser relativamente simple, como la de un taxi que debe recoger a un cliente y llevarlo a donde haga falta; pero puede complicarse todo lo que se desee, como la del repartidor colaborativo que debe recoger paquetes de varias fuentes, de manera que todos tengan destinatarios que residan cerca, optimizando las labores de suministro, etc. También pueden combinarse ambos agentes en una misma simulación, basta con adjuntar ambas estrategias y crear

[8] www.cloud.anylogic.com/model/3a449f36-be23-4431-98d4-4d8f8f3b414c?mode=SETTINGS
[9] www.github.com/javipalanca/simfleet
[10] Instituto Valenciano de Investigación en Inteligencia Artificial, o en inglés, *Valencian Research Institute for Artificial Intelligence*
[11] www.blablacar.es

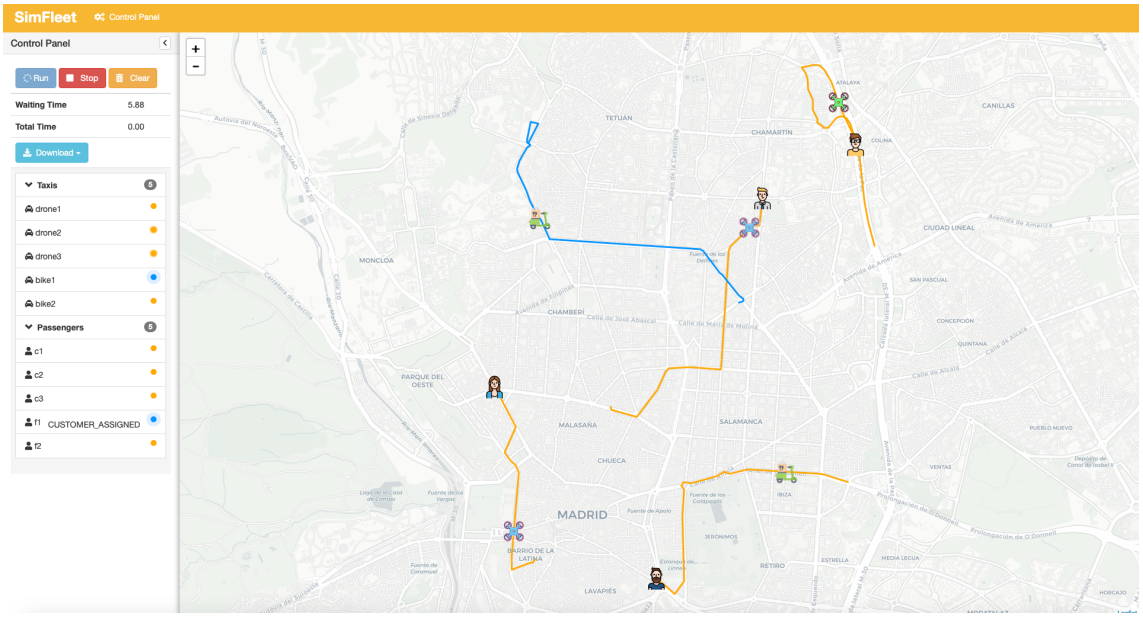


Figura 2.6: Captura de la interfaz gráfica de SimFleet durante la ejecución de un experimento

Fuente: documentación de SimFleet

dos flotas de vehículos distintas. En resumen, SimFleet, como simulador multiagente, es muy versátil a la hora de modelar la realidad de los actores del sector del transporte.

Uno de los aspectos cruciales que deben ser especificados para recrear las estrategias de los agentes es la comunicación que realizan con otros agentes del mismo o de distinto tipo. Tal como hemos visto en el capítulo relativo a la simulación multiagente, los agentes tienen que comunicarse para poder avanzar en sus labores. En los ejemplos que estamos analizando: un repartidor debe recibir aviso de dónde se encuentran los centros logísticos de donde extraer los productos o las materias primas, cuántos hay y a dónde están dirigidos. En el ejemplo del *car-sharing*, a los viajeros les tiene que constar quién va a realizar el trayecto que quieren hacer, de dónde parten (para ello existe BlaBlaCar), etc. En este marco, SimFleet facilita en gran medida la programación de las negociaciones de los agentes que intervienen en una simulación, abstrayendo todo lo relacionado con la comunicación de los agentes. Debajo de la capa de abstracción se encuentra un servidor XMPP, que hace llegar los mensajes enviados de un agente a otro de manera muy sencilla. Para mayor eficiencia, soporta la recepción asíncrona de los mensajes, de manera que los agentes no tienen que detenerse para recibirlos y procesarlos.

Un caso de uso concreto de este software de experimentos es el enunciado por Palanca et al. en [24]. En este caso de uso se presenta la versión predeterminada de SimFleet para simular flotas de taxis en una ciudad, explicando cómo funciona la negociación entre taxi y cliente para poner en marcha un servicio de recogida y de transporte hasta el destino del cliente.

El simulador se controla y ejecuta desde la línea de comandos. Sin embargo, el propio programa da soporte a una interfaz HTML para visualizar la ubicación de todos los agentes, la ruta que están siguiendo, el tiempo de ejecución, etc. En la figura 2.6 podemos ver una captura de ejemplo de la documentación de SimFleet[12], en la que se transportan paquetes con drones y bicis a particulares, en un buen ejemplo de coexistencia de dos tipos de agentes de transporte (flotas) distintos.

[12] www.simfleet.readthedocs.io/en/latest/usage.html

2.3 Propuesta

De entre todos los simuladores que se han barajado como alternativa, hemos escogido SimFleet como mejor opción. Uno de los factores que nos han hecho decantarnos por este paquete es que es el mismo programa con el que han investigado los tutores de este proyecto de fin de grado. El simulador fue creado por ellos junto con otros miembros del VRAIN. Es conveniente hacer uso de la herramienta que conocen bien para poder aprovechar al máximo todas sus virtudes.

Otro de los factores que apoyan la utilización de SimFleet es que, como hemos visto en el capítulo que precede, fue desarrollado con la intención de resolver el problema de la optimización de emplazamientos de estaciones a nivel urbano. También existía el propósito de hacerlo accesible de forma gratuita a todo el mundo para resolver problemas similares relacionados con la simulación de flotas de vehículos. Es un simulador muy sencillo de modificar y amoldar a nuestras necesidades individuales, simplemente reutilizando los agentes existentes pero modificando casi al completo algunas de sus estrategias.

El simulador fue diseñado en un principio para realizar experimentos sobre un entorno urbano. Sin embargo, cumple todas las características de escalabilidad para ser utilizado en entornos más grandes, como el interurbano. En nuestro caso, necesitamos un simulador capaz de simular sobre todo el mapa español. Al ser un simulador multiagente, en el que cada agente es una unidad separada de procesamiento, bastaría con cambiar el mapa para poder seguir con la investigación.

El argumento más convincente, no obstante, es el enfoque que tiene el simulador sobre los agentes y la información que se genera durante un experimento. Otros simuladores que hemos analizado recopilan información mucho más física y mecánica de los vehículos (microsimuladores), cuando nosotros solo necesitamos estadísticas mucho más generales. Al igual que con los agentes, este programa permite añadir más métricas que se quieran recabar del proceso de simulación, siempre que se implemente también el código correspondiente para calcularlas. Además, es muy fácil visualizar y manipular los datos para realizar cálculos posteriormente con la posibilidad de exportar los datos a formatos que se puedan persistir en memoria.

Por tanto, de entre todas las herramientas potenciales que se han considerado, esta es la ideal para poder llevar a buen puerto este trabajo de investigación. En el siguiente capítulo se detalla el funcionamiento interno del simulador, aquellos aspectos que deben cambiarse para adaptarlo a nuestros experimentos y las tecnologías que se van a emplear para complementar el estudio.

CAPÍTULO 3
Diseño de la solución

Como en todo proyecto en la disciplina del desarrollo de software, una vez completada la planificación del proyecto y seleccionado el simulador sobre la que trabajar, se avanza a la fase de diseño. En esta fase aún no se escribe ningún código, sino que se plantea cómo resolver el problema al que nos enfrentamos con la herramienta concreta de que se dispone. En nuestro caso, tenemos disponible un simulador, SimFleet, que posee una arquitectura y un funcionamiento concreto. La función de esta fase del proyecto es determinar cómo proceder de manera más adecuada para aprovechar sus características al máximo. En consecuencia, en esta sección vamos a realizar, en primer lugar, un estudio más en profundidad de SimFleet. Terminaremos explicando detalladamente qué necesita ser modificado para obtener el simulador que requerimos para los experimentos.

3.1 Arquitectura del sistema

Antes de continuar a la fase de diseño propiamente dicha, debemos establecer el punto de partida en el que nos encontramos. El simulador del que nos vamos a servir para experimentar con las posibles localizaciones de estaciones de recarga tiene unas características muy útiles por las que nos hemos decantado por él. En esta sección vamos a profundizar en el análisis de SimFleet a nivel de arquitectura.

El paquete SimFleet consiste, como sabemos, en una serie de ficheros de código escritos en el lenguaje de programación Python y ficheros de configuración en formato JSON. Los ficheros Python definen el comportamiento del simulador y de todos los agentes que viven durante sus ejecuciones. Los ficheros JSON determinan las características de los agentes, entre otras, el número de puntos de carga por cada estación, su potencia de carga, la velocidad máxima de los vehículos o su autonomía máxima.

3.1.1. La estrategia de los agentes

Primero, recordemos brevemente el concepto de agente dentro del contexto de la simulación multiagente. Un agente es una unidad de procesamiento independiente de una simulación multiagente. Se trata de un ente que actúa por sí mismo en función de su estado, del estado de los agentes de su entorno y de su comportamiento predeterminado (en forma de reglas). Por tanto, un agente debe tener claramente definidos cuáles son sus posibles estados, cómo y con qué agentes debe comunicarse para averiguar su estado y en base a qué reglas debe transicionar de un estado al siguiente. Lo más destacable de la simulación multiagente es que cada agente es una unidad que actúa por sí misma, tomando sus propias decisiones para avanzar en su propósito. Estos agentes pueden

ser cientos o miles dentro de una misma simulación, siguiendo una misma estrategia e interactuando entre ellos.

En SimFleet existen varios tipos distintos de agentes. Cada uno de ellos requiere un archivo de código en el que se especifique su estrategia. En este caso, los archivos están codificados en Python y tienen como nombre el agente al que definen (por ejemplo, simulator.py). En cada estrategia tendremos su comportamiento propio, cómo interacciona con los demás agentes y cuándo debe cambiar de estado. Los agentes más relevantes de SimFleet y su estrategia respectiva se enumeran a continuación.

SimulatorAgent

El núcleo de SimFleet reside en el agente simulador. Es el agente que se levanta en primer lugar, dando comienzo a la ejecución de un experimento. Su lanzamiento supone la inicialización de la infraestructura del simulador: directorio, servicio de comunicación XMPP, etc.; y de la instanciación de todos los demás agentes de manera ordenada, aplicando a cada uno los metadatos correspondientes según el fichero de configuración. Del mismo modo, también es el agente que finaliza la simulación, calculando y recopilando las estadísticas del experimento y deteniendo el proceso cuando se cumplen una serie de condiciones. Cabe destacar que no se puede variar la estrategia de de *SimulatorAgent*, solamente se pueden especificar unos pocos parámetros, como el tiempo máximo de ejecución, el puerto de escucha de la interfaz gráfica o el nivel de verbosidad de los *logs**.

Estamos hablando pues de un agente que sigue un orden de ejecución más o menos lineal, dando comienzo a la ejecución, supervisándola y deteniéndola en el momento oportuno. Detallemos cada fase de su ciclo de vida:

1. **Comienzo de la simulación**: al ejecutar el comando de SimFleet, se levanta una instancia de la clase *SimulatorAgent*. El agente, en primer lugar, extrae la configuración global de la simulación, como el tiempo máximo de ejecución y el puerto de escucha de la interfaz gráfica etc. Además, inicializa estructuras de datos que va a requerir más adelante, como por ejemplo los *dataframes** donde se almacenan los resultados del experimento (contienen tiempo de ejecución, estado final de cada transporte, tiempo medio de espera en las estaciones...). Por último antes de dar luz verde a la simulación, guarda dónde se encuentran las las estrategias de los demás agentes y despierta al agente directorio para poder registrar a los nuevos agentes.

   A continuación, por cada tipo de flota de transportes, clientes y estaciones, se genera una instancia de *FleetManagerAgent*. Este objeto contiene la estrategia de cada tipo de agente distinto y es el encargado de generar, a su vez, los agentes que le correspondan, y supervisarlos durante toda la ejecución. El manager creará e inscribirá los agentes de veinte en veinte, indicando a todos ellos la estrategia que deben seguir y los parámetros contenidos en el fichero de configuración que les conciernen. Una vez inicializados todos los agentes de todas las flotas se ejecutan todas las estrategias secuencialmente, se inicia el cronómetro y se da comienzo a la simulación.

2. **Transcurso de la simulación**: *SimulatorAgent*, durante la simulación, se dedica exclusivamente a evaluar si se cumplen las condiciones de finalización de la simulación. El comportamiento de los agentes de la simulación y su comunicación son responsabilidad de los *FleetManager* y del *DirectoryAgent* por medio servicio de mensajería XMPP, respectivamente. Por tanto, no lo tiene que administrar el agente de la simulación.

Existen dos condiciones de finalización. Por un lado, la ejecución puede detenerse por rebasar el tiempo máximo de simulación. Este límite puede especificarse en el fichero de configuración de cada simulación. Si no se ha especificado ninguna cota temporal, no habrá límite por este lado y el fin de la simulación dependerá exclusivamente de la segunda condición de finalización.

La segunda condición consiste en comprobar que todos los clientes han sido entregados en su destino. *SimulatorAgent* reconoce esta situación evaluando si, para cada uno de los *CustomerAgent*, se cumple que su estado es *CUSTOMER_IN_DEST* y que se encuentra en la ubicación final que se había determinado en el archivo de configuración.

Ambas condiciones se validan cada medio segundo. En caso de no haber configurado ningún umbral de tiempo máximo o no existir ningún cliente, la condición correspondiente no será tenida en cuenta. Se recomienda, no obstante, especificar un máximo temporal. Puede ocurrir, por descuido, que el simulador se ejecute de manera infinita si la estrategia no favorece que todos los agentes logren su meta. En cualquier caso, si alguno de los controles resulta positivo, se inicia la detención de la ejecución del experimento.

3. **Finalización de la simulación**: esta fase comienza exclusivamente en caso de cumplirse alguna de las condiciones de finalización. Lo primero de todo que realiza *SimulatorAgent* es detener el cronómetro nada más decretarse el fin del experimento y registrar la duración final. En ningún caso podrá ser mayor que la duración máxima.

   Seguidamente, el agente simulador detiene el directorio, evitando que otros agentes sigan realizando sus negociaciones. Después es el turno de la finalización de la ejecución de cada uno de todos los agentes restantes (primero las instancias de *FleetManager*, luego las de *TransportAgent*, etc.)

   El último cometido de *SimulatorAgent* es recopilar y visualizar las estadísticas recabadas en el proceso de simulación. Con dicho fin, tiene que reunir la configuración básica del experimento, que contiene el nombre de la simulación, su duración etc.; la información relativa a los managers de flotas, de los transportes, clientes y estaciones de recarga. Ello conlleva una serie de cálculos, pues además de los datos particulares de cada agente, se calculan una serie de agregaciones y de promedios que son muy representativos del experimento, como el tiempo medio de espera, la distancia media recorrida por los transportes, etc.

   Finalmente, antes de finalizar la ejecución y devolver el control al usuario, *SimulatorAgent* imprime todos estos resultados en el lugar que corresponde. SimFleet dispone de varias salidas donde depositar los datos. Dependiendo de la orden de ejecución en consola, utilizará una salida u otra.

   Por defecto, el simulador devuelve el *dataframe* que acaba de fabricar por la salida estándar, es decir, lo imprime en la misma consola desde la que se ha ejecutado la simulación. Esta es una buena opción para ver los resultados en el momento en que termina la simulación, pero los datos no se persisten, sino que desaparecen cuando se cierra la consola. Para resolver este problema, SimFleet soporta la exportación del *dataframe* a otros formatos, como JSON y Excel. En cada caso, la información que se mostraría en consola es volcada en un fichero del formato correspondiente.

   Adicionalmente, en caso de desear obtener una copia del diario de la ejecución, o *logs*, se puede derivar la salida de error a un fichero de texto. Para ello, se añade la directiva “2>” al comando de ejecución de SimFleet seguido del nombre del archivo donde escribir la información.

En resumen, *SimulatorAgent* es el agente nuclear de SimFleet. Su misión es iniciar la simulación, creando todos los agentes e iniciando sus estrategias respectivas; determinar el momento del fin de la ejecución y finalizarla, desmontando toda la infraestructura construida en la fase de arranque y calculando todas las estadísticas relativas a la simulación, depositando dicha información en el lugar pertinente para su posterior visualización.

DirectoryAgent

El agente directorio realiza todas las labores relacionadas con el envío y recepción de mensajes entre agentes protagonistas de las simulaciones: los transportes, los clientes y las estaciones. Es el único agente con acceso al servicio de mensajería asíncrono del simulador, que como sabemos, está implementado mediante el protocolo XMPP.

La estrategia o comportamiento se divide en dos partes cruciales para la comunicación inter-agentes:

- Por un lado, como todo servicio de mensajería, ya sea entre procesos de un programa, entre hilos de ejecución, entre agentes o entre seres humanos (por ejemplo, correo electrónico y postal), debe mantener un registro de todos los entes que existen en la población. Al igual que los servidores de correo electrónico disponen de una base de datos para saber a dónde dirigir un mensaje, nuestro servidor de mensajería necesita una estructura para saber a qué agente direccionar un mensaje, y cómo regresar la respuesta al remitente. Por este motivo, como hemos visto antes, este agente es el primero en inicializarse, quedando a la espera de peticiones de registro en el directorio. A la llegada de una petición, el agente anota en un diccionario* Python la correspondencia entre el identificador del emisor y su bandeja de entrada. Este proceso se repite tantas veces como agentes se creen y registren en el directorio. Posteriormente, cuando un agente quiera comunicarse con otro agente, el directorio sabrá por qué canal redirigir la conversación.

- Por otro lado, durante la duración de la ejecución de una simulación, el directorio es responsable de poner en contacto a los agentes que así lo soliciten. En SimFleet, la mayoría de mensajes se producen en el proceso de petición de recarga de un transporte a una estación. Para poder llevar a cabo una recarga, un vehículo tiene que solicitar permiso para poder acudir a una estación. Su petición será reenviada por el directorio, que es el único que sabe a qué agente dirigirse sabiendo su identificador. Si la estación está operativa y habilitada para proporcionar el servicio que requiere el transporte, esta aceptará la petición en forma de una respuesta afirmativa, que seguirá el mismo protocolo que a la ida. Lo mismo ocurre en caso de rechazarse la petición, salvo que el contenido del mensaje y la acción que acometerá el transporte a su recepción, serán distintos.

Resumiendo, *DirectoryAgent* es el intermediario del servicio de mensajería entre agentes del simulador. Sin un mapa con el que poder redirigir las conversaciones entre agentes por medio de su identificador, estos no podrían comunicarse. La comunicación entre agentes es una característica fundamental de la simulación multiagente, por lo que *DirectoryAgent* es fundamental en este aspecto.

TransportAgent

Llegamos al agente más representativo de SimFleet, el agente transporte. Como su nombre indica, modela la figura del medio de transporte. Puede tratarse de cualquier

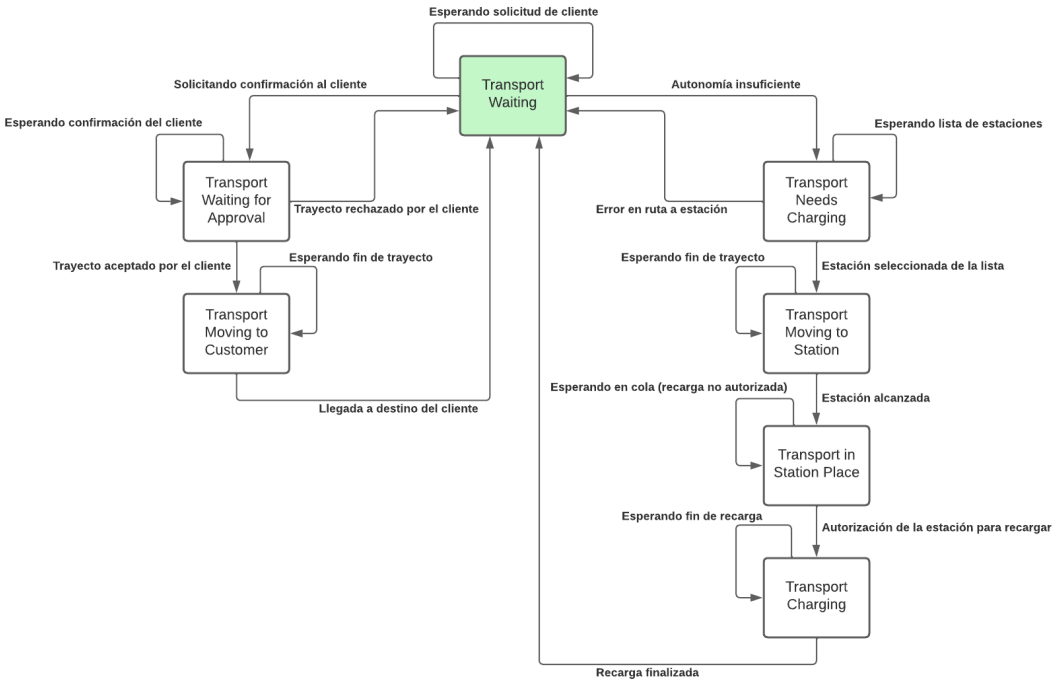


Figura 3.1: Máquina de estados de *TransportAgent* antes de las modificaciones

En verde, el estado inicial. Elaboración propia

tipo de transporte, desde el turismo, hasta el camión o el autobús. Todo depende de la estrategia que se le imponga. Originalmente, en SimFleet, *TransportAgent* sigue la estrategia del taxi, que consiste en esperar la llamada de un cliente, recogerlo, y depositarlo en el destino acordado. Veamos con más detalle el comportamiento de este agente.

TransportAgent, como todo agente principal de simulación multiagente, puede encontrarse en una serie de estados distintos, entre los que transiciona cuando se cumplen una serie de requisitos. En la figura 3.1 se encuentra el grafo correspondiente a la máquina de estados de este agente.

- ***TransportWaitingState***: este es el estado básico e inicial del transporte. Cada vez que el agente termina de realizar una actividad, como recargar o transportar a un cliente, vuelve a este estado. Al comenzar la ejecución del comportamiento en este estado, espera durante un minuto la llegada de un mensaje de un cliente que transportar. Si no recibe nada, vuelve a entrar en el mismo estado de espera, repitiendo la ejecución. En caso de recibir un mensaje, lee la ubicación del cliente y la de su destino y consulta la autonomía restante de las baterías para saber si puede realizar el servicio o si debe parar a recargarlas. Si no tiene suficiente energía, contesta al cliente con un mensaje de cancelación y entra en el estado *TransportNeedsChargingState*. En caso contrario, contesta al cliente ofreciéndose a realizar el trayecto y transiciona al estado *TransportWaitingForApprovalState*.

- ***TransportNeedsChargingState***: un transporte alcanza este estado cuando detecta de antemano que no puede desplazarse hasta la localización de su cliente o que la energía es insuficiente para llevarlo a su destino. El objetivo del vehículo en este estado es encontrar una estación donde poder recargar sus baterías. Para ello, si no dispone de la lista de estaciones disponibles, tiene que solicitarla al directorio. Esperará como mucho un minuto a que el directorio recopile las posibles estaciones.

En caso de recibir una contestación desfavorable o en su ausencia, reinicia el estado y, consiguientemente, todo el protocolo de solicitud de recarga. En caso contrario, el agente almacena la lista de estaciones para futuras ocasiones y da comienzo el algoritmo de selección de estaciones.

En esta versión por defecto de SimFleet, el transporte recorre todo el directorio de estaciones que posee calculando a qué distancia se encuentra de cada una de ellas. Se selecciona aquella que está más cerca de la ubicación actual del vehículo. Entonces, el agente comunica a la estación a través de la dirección obtenida por el directorio su intención de recargar en su emplazamiento. Una vez obtenido el permiso, solicita al servidor de rutas el camino más corto para llegar a ella. Si el servidor de rutas no está disponible, el transporte se pone en modo de espera en el estado inicial. Si no, se pasa al estado *TransportMovingToStationState* lanzando el protocolo de movimiento del agente.

Cabe destacar que en esta versión del simulador, no se comprueba el estado de carga de los vehículos antes de acudir a la estación, por lo que podría darse el caso de tener un vehículo circulando más distancia que lo físicamente permitido por la batería. Esto es así para evitar que un vehículo se quede en una situación de bloqueo infinitamente intentando ir a recargar sin suficiente potencia, dejando de lado sus funciones principales. Es una simplificación que en la práctica no supone demasiado cambio, pues en ciudad, que es donde se experimenta con esta versión, no hay demasiado kilometraje entre estaciones. En cualquier caso, se almacena la distancia recorrida en el cuentakilómetros del transporte para poder agregar la distancia total al finalizar el experimento.

- ***TransportMovingToStationState***: como se ha mencionado en el punto anterior, este estado representa el movimiento del transporte desde su localización actual hasta la estación. Salvo que el vehículo se encuentre ya dentro de la propia estación, tiene que esperar a que finalice el protocolo de movimiento. Para ello, resetea un atributo del agente y espera a que se dispare un evento. Cuando en el protocolo de movimiento se detecta la llegada del vehículo, se ejecutará un *callback** que dispare el evento, activando de nuevo la ejecución del estado actual. Antes de comenzar la recarga en uno de los puntos de la estación, tiene que solicitar permiso, entrando en el subsiguiente estado.

- ***TransportInStationState***: este estado consiste en la espera del transporte en la estación hasta que su agente autoriza la recarga de la batería. Si la estación impide la recarga es porque todos los terminales están ocupados y el vehículo debe hacer cola. Por tanto, vuelve a entrar en este mismo estado. En caso favorable, inicia el protocolo de recarga y entra en el estado *TransportChargingState*.

- ***TransportChargingState***: mientras el transporte se encuentra recargando, necesita averiguar por parte de la estación si ha finalizado el protocolo. El propio protocolo se encarga de aumentar la autonomía del vehículo al máximo cuando transcurra el tiempo de carga. El tiempo de carga depende de la carga que tengan las baterías al comienzo del repostaje y del voltaje del punto de recarga. Cada minuto, el agente envía un mensaje preguntando si ha terminado. Si no ha terminado, permanece en el mismo estado. En caso afirmativo, ejecuta la salida de la estación y regresa al estado inicial.

- ***TransportWaitingForApprovalState***: para poder entrar en este estado, partimos del estado de espera inicial. Decíamos con anterioridad que un transporte espera la llegada del mensaje de un cliente, y que tiene que solicitarle confirmación después de ratificar que tiene suficiente capacidad energética para poder realizar el viaje.

Mientras el cliente procesa esta solicitud, el transporte se encuentra en este estado. Al igual que en otros estados, el *timeout** asciende a un minuto, tras el cual, en caso de no encontrar respuesta, se reinicia el estado y se espera el contacto por parte de un nuevo cliente. Recibida la contestación del cliente, se chequea si hay suficiente autonomía para transportar al viajero. Dependiendo del resultado, se pasa al estado *TransportNeedsChargingState* avisando al cliente de que se cancela el servicio; o se avanza al estado *TransportMovingToCustomerState* iniciando el movimiento hacia el cliente. También puede cancelarse el servicio si el servidor de rutas no es capaz de proporcionar el camino a seguir hasta el cliente.

A diferencia del protocolo del viaje hasta una estación, en este caso (y en todos los demás trayectos) sí que se resta la autonomía del transporte en la cantidad de kilómetros que va a recorrer hasta su próximo destino. Así, la siguiente vez que se controle la autonomía, el valor será acorde a la distancia recorrida desde que se recargó por última vez.

- ***TransportMovingToCustomerState***: en último lugar tenemos el estado del desplazamiento hasta el lugar donde espera el cliente. Se sigue el mismo esquema que cuando se acude a una estación de recarga, la comprobación de que la tarea de movimiento asíncrona ha finalizado correctamente. Cuando se lanza el evento correspondiente a la llegada del transporte a la ubicación del cliente, se ejecuta el protocolo de transporte hasta el destino del cliente y se transiciona otra vez al estado de espera inicial.

En conclusión, el protagonista de los experimentos en SimFleet, *TransportAgent*, es un agente que espera activamente la llegada de una notificación de un cliente que desee realizar un trayecto mediante su medio. A su llegada, comprueba el estado de sus baterías para saber si puede realizar el servicio al completo o si tiene que recargarlas en primer lugar. En función del estado, recargará las baterías y esperará la llegada de un nuevo cliente, o se dirigirá a la ubicación del cliente para transportarlo. Hemos visto el alto nivel de comunicación con el cliente y de negociación con las estaciones para poder repostar, una característica muy importante de la simulación multiagente, y que sin el agente directorio, recordemos, no sería posible. En todo momento se tiene en cuenta la autonomía del vehículo, reduciéndose al realizar un desplazamiento y aumentándose correspondientemente en las estaciones.

StationAgent

A continuación, vamos a analizar la estrategia de las estaciones de recarga en SimFleet. Podría decirse que *StationAgent* es un agente pasivo, aunque sea otra figura protagonista de las simulaciones, pues sus acciones dependen de lo que hagan los demás agentes. Las estaciones intervienen en tres acciones distintas durante un experimento:

- En primer lugar tenemos la situación en la que un *TransportAgent* desea realizar una recarga en una estación concreta. Como hemos visto en el apartado correspondiente a dicho agente, este se dirige por mensajería a la estación. Las estaciones se encuentran en todo momento a la escucha, pendientes de la llegada de hipotéticas solicitudes de este tipo. La frecuencia con la que las estaciones consultan su bandeja de entrada es de cinco segundos. Aceptada o cancelada la propuesta de recarga, se envía la respuesta correspondiente de vuelta al agente remitente. La contestación favorable contiene una confirmación del identificador de la estación y de la ubicación de la estación en el mapa.

- En segundo lugar, cuando un vehículo llega a la estación, este tiene que pedir, del mismo modo, permiso para poder empezar a cargar baterías. Es responsabilidad de la estación llevar el conteo de terminales de recarga total (puede haber más de uno), desglosado en aquellos que están libres y los que están ocupados. Por tanto, cada vez que procesa la llegada de un nuevo vehículo debe consultar el número de terminales que están disponibles. En caso de estar todos los terminales de carga ocupados, se envía un mensaje rechazando la entrada del transporte en la estación. Como se ha detallado anteriormente, este volverá a intentar la entrada a la estación pasado un minuto. En el caso contrario (hay al menos un punto de carga disponible), se resta una unidad al contador de terminales libres y se comunica al vehículo que puede comenzar a recargar.
- Al terminar el protocolo de recarga, un *callback* lanza un evento, permitiendo que el transporte reanude su estrategia, abandonando la estación y continuando con sus tareas. Al liberar un terminal, la estación actualiza el número de estaciones libres, para que en la siguiente lectura de esta cifra se pueda dar acceso al siguiente vehículo en caso de haber cola.

En definitiva, el cometido del agente estación es comunicarse con los transportes que solicitan recargar baterías y autorizar su acceso a un terminal cuando exista alguno disponible. Del mismo modo, debe permitir la salida del transporte del recinto de la estación cuando la autonomía esté al máximo. En todo momento, el registro de estaciones libres tiene que estar actualizado para poder decidir cuándo permitir la entrada de vehículos, así que se aumenta/decrementa cada vez que se produce una entrada/salida de un transporte.

CustomerAgent

Como último agente protagonista de SimFleet tenemos al agente cliente. Representa a un individuo de a pie que se dirige a un lugar concreto, y que para realizar el trayecto solicita un servicio de transporte, como el taxi. Este modelo de agente puede utilizarse también para representar una mercancía que debe enviarse mediante un repartidor a su destinatario. Al igual que *TransportAgent*, puede encontrarse en una serie de estados entre los que puede transicionar.

- ***Customer Waiting***: Al igual que los transportes, los clientes disponen de un estado inicial de espera. Cuando un cliente está preparado para realizar el viaje, solicita al *FleetManager* correspondiente al tipo de transporte que quiere que le lleve a su destino (podría ser taxi, VTC, vehículo autónomo...). Al principio, el cliente no sabe cómo dirigirse al mánager de flotas, por lo que es posible que tenga que solicitar la lista de mánagers al agente directorio. Si en cinco minutos no recibe contestación o la petición es cancelada, vuelve a pedir la información de nuevo.

  Enviada la solicitud al mánager de flotas, este contestará con uno de los transportes que están libres y a la espera de realizar un servicio. Por defecto, la estrategia de los clientes es la de aceptar al primer transporte que se ofrezca a llevarles a su destino. Evidentemente, se puede sustituir esta estrategia por una en la que, por ejemplo, el cliente acepte solo transportes para los que tenga que esperar menos de cinco minutos, o que sean vehículos de alta gama, etc. En este caso, al aceptar un transporte, se le debe enviar un mensaje de vuelta para que evalúe si sus baterías están en condiciones su destino. Si no tiene suficiente autonomía, el agente contesta al cliente cancelando el viaje, por lo que vuelve a entrar en estado de espera y solicita al mánager de flotas un nuevo transporte al que dirigirse. Nótese que en este momento

el agente no tiene que volver a requerir la lista de mánagers al directorio, ya la tiene almacenada en su memoria. Si resulta que el transporte sí tiene la posibilidad de realizar el servicio, el cliente avanza al siguiente estado.

Puede ocurrir que llegue un mensaje de un transporte con retraso, es decir, cuando el cliente ya no está en estado de espera. Este es el único caso en el que el cliente va a rechazar a un transporte. Es un caso muy improbable, pero en el paradigma asíncrono del servicio de mensajería del simulador, se tiene que cubrir este caso, pues un transporte no contesta inmediatamente, sino cuando deja de estar ocupado realizando otras funciones.

- ***Customer Assigned***: representa el estado de un cliente que está esperando para ser recogido por un transporte, pero no tiene relevancia de cara a la lógica de la estrategia de ningún agente.

- ***Customer in Transport***: este estado sirve solamente para activar el cronómetro del viaje de un cliente cuando es recogido.

- ***Customer in Destination***: el transporte, cuando alcanza el destino del cliente, es el que comunica al agente cliente que el viaje ha terminado. Ello provoca que el cliente avance a este estado y pare el cronómetro del viaje. El tiempo de viaje es relevante de cara a las estadísticas individuales de cada cliente y se incluirá en la hoja de resultados de la simulación. Aquí acaba la función del agente, que permanece en este estado hasta que finaliza la ejecución del experimento.

Por tanto, el agente cliente es el modelo de la persona o el objeto que viaja en el transporte. En la estrategia predeterminada de SimFleet, su función es solicitar un vehículo para realizar el viaje a su destino deseado, seleccionando el primero que se muestre preparado a llevarlo a cabo. También es responsable de cronometrar el tiempo de viaje. Transiciona entre una serie de estados hasta acabar en el destino, momento en el que finaliza su aparición en la simulación.

FleetmanagerAgent

Por último, es necesario mencionar la existencia del agente que supervisa las negociaciones entre miembros de una flota de transportes y los clientes, el mánager de flotas. Se trata del primer tipo de agente que se instancia después de *SimulatorAgent*. Todos los transportes del mismo tipo tienen que estar registrados en el agente mánager correspondiente para poder redirigir las conversaciones con los clientes. El funcionamiento es similar al de *DirectoryAgent*.

En resumen, hemos visto los distintos agentes que concurren en una ejecución de SimFleet, el orden en que se ejecutan, cómo se comportan y cómo se realizan las comunicaciones entre ellos. Cabe destacar que la lista de agentes puede modificarse tanto como se desee, es decir, pueden crearse nuevos agentes para realizar funciones distintas a los existentes. Por ejemplo, podría crearse un agente autobús para instaurar un servicio público de transporte, con preferencia sobre el taxi.

3.1.2. Configuración de los agentes

Los principales agentes que acabamos de describir poseen una serie de atributos que se tienen que especificar al comienzo de una simulación. Se trata de todas aquellas variables dependientes que determinan el devenir de un experimento. Para proporcionar al simulador todos los datos que forman parte del *input* se hace uso de los ficheros de configuración. Estos ficheros de configuración no contienen más que un diccionario en formato JSON con los metadatos de un experimento. Vamos a detallar qué son estos datos que hay que proporcionar, entre otros:

- ***Fleets***: diccionario que contiene una entrada por mánager de flotas, que incluye el nombre y el tipo de la flota.
- ***Transports***: diccionario que contiene una entrada por transporte, que incluye la velocidad en kilómetros por hora, el identificador de su flota, el tipo de flota, la posición inicial, el nombre y las autonomías inicial y máxima.
- ***Customers***: diccionario que contiene una entrada por cliente, que incluye posición inicial, destino, nombre y tipo de flota a utilizar.
- ***Stations***: diccionario que contiene una entrada por estación de recarga, que incluye su nombre, la posición, la cantidad de terminales y su voltaje, expresado en kilómetros de autonomía que se recargan por segundo.
- ***Simulation Name***: nombre de la simulación, que aperecerá en la cabecera de la tabla de resultados del experimento.
- ***Max Time***: tiempo máximo de ejecución en segundos. El agente simulador comprueba que no se supere este límite, en cuyo caso finaliza la simulación aunque aún queden clientes por transportar.
- ***X_strategy***: ubicación en el código de la estrategia de cada agente (donde X representa transporte, cliente etc.).
- ***Route Host***: URL del servidor de rutas.

Cada vez que se lanza una simulación en SimFleet esta tiene que venir acompañada de un archivo de este tipo. La ventaja de este sistema de configuración es que permite almacenar distintas configuraciones para los experimentos, y así poder repetirlos las veces que haga falta. Estos experimentos pueden ponerse en cola, de manera que el simulador lea cada configuración de manera secuencial y ordenada.

3.1.3. Generación de agentes

Como acabamos de ver, los ficheros de configuración contienen mucha información acerca de los experimentos. Los primeros cuatro puntos son los más determinantes para la simulación, pues establecen la cantidad de agentes de cada tipo que existirán, sus posiciones iniciales y sus destinos. Redactar un archivo de configuración con una alta cantidad de entradas de agentes puede resultar muy tedioso. Además, en la mayoría de ocasiones, los agentes deben seguir un patrón a la hora de ubicarse en el entorno de un experimento. Es por ello que disponemos también de un proyecto de generadores de carga para SimFleet, desarrollados en distintos trabajos [25, 26]. Estamos hablando de unos programas que automatizan la creación de agentes y la escritura del archivo de configuración. El paquete contiene programas muy importantes para las investigaciones de los tutores del presente trabajo.

Uno de ellos es un script escrito en Python que genera ficheros de configuración para la versión predeterminada de SimFleet que acabamos de describir. Al ejecutar el programa tenemos que especificar qué distribución deben seguir las estaciones a colocar:

- La distribución *random* consiste en colocar las estaciones en ubicaciones arbitrarias a lo largo del mapa.
- En la *radial*, se ubican todas formando círculos concéntricos sobre el eje central del mapa.
- En la *uniform*, se divide el mapa especificado en cuadrículas del mismo tamaño, colocando una estación en su centro.

Es necesario disponer de una herramienta como el mencionado script para poder generar correctamente estas distribuciones. Se ejecuta con el siguiente comando desde la terminal del sistema:

```
stations_generator.py -d <distribution_type> -s <num_stations>
    -p <num_poles> -i <input_borders_file> -o <output_file>
    [-r (random)] [-c (num_circles)]
```

donde se tiene que especificar el tipo de distribución (*uniform*, *radial* o *random*), el número de estaciones a generar, el número de terminales de carga total que se repartirán entre las estaciones, la ruta a un fichero GeoJSON* donde se especifique los límites geográficos de la región donde emplazar las estaciones y el fichero de configuración donde rellenar el diccionario de estaciones. Opcionalmente, se puede solicitar que las estaciones no se ubiquen en el centro de su zona de influencia, sino en un punto aleatorio en su interior. En caso de generar estaciones para la distribución radial, se obliga al usuario a especificar el número de círculos concéntricos a dibujar en el mapa.

A parte del generador de estaciones, también se incluye un generador aleatorio de transportes. En este caso, el script realiza la misma función que el anterior generador: generar una cantidad específica de transportes con una ubicación inicial y final repartidas a lo largo de una región dada. Este programa solamente da soporte a la creación aleatoria de ubicaciones de comienzo y fin de trayecto y se ejecuta de la siguiente manera, también desde la línea de comandos:

```
random_load_generator.py -n <num_transports> -d <min_distance>
-m <GeoJSON with map>
[-c <simfleet config file>] [-s (save GeoJSON with routes)]
```

donde tenemos que introducir el número de transportes a generar, la distancia mínima entre las ubicaciones iniciales y finales y un archivo GeoJSON con las fronteras del mapa. Opcionalmente podemos proporcionar un fichero de configuración donde introducir el diccionario resultante y el nombre de un fichero donde generar un GeoJSON con las ubicaciones de partida y de llegada unidos por una línea.

Este generador, en breves palabras, escoge una ubicación inicial aleatoria, y a partir de ella, una segunda para la ubicación final. Estas dos ubicaciones serán las definitivas si ambas se encuentran dentro de los límites marcados por el mapa GeoJSON. Esto se repite tantas veces como vehículos se haya solicitado generar. Además, se le asigna a cada vehículo unos atributos (autonomía, velocidad) por defecto.

Por último, se nos proporciona en este paquete de generadores un código para convertir un fichero de salida del algoritmo genético desarrollado por Jordán et al. [6] en

un fichero de configuración SimFleet. Este código coge la colección de ubicaciones producida por el algoritmo y genera a partir de ellas el diccionario de estaciones. Al final obtenemos una lista de estaciones con las ubicaciones óptimas según el algoritmo.

3.2 Diseño detallado de las adaptaciones necesarias

Analizado en profundidad el punto de partida del simulador que hemos escogido, es el momento de decidir y diseñar las adaptaciones que queremos aplicarle. Recordemos que uno de los objetivos del trabajo es proporcionar el entorno necesario para poder analizar las posibles estrategias en el emplazamiento de estaciones de recarga. El fin de la obtención de este entorno de experimentación es averiguar la idoneidad de las distintas distribuciones que nos proporciona el simulador de carga, el algoritmo genético de ubicación óptima de estaciones desarrollado por los tutores del proyecto y la red de estaciones de Tesla en España. Es evidente que el modelo taxi-cliente actual no nos va a servir para cumplir estos objetivos, pero hay muchas características que podemos aprovechar de esta primera versión del simulador y otras que requieren una implementación más extensa. SimFleet está desarrollado de manera que facilita el moldeado de sus agentes. A continuación se detalla con más concreción los aspectos a cambiar.

3.2.1. Desaparición de *CustomerAgent*

El modelo predeterminado de SimFleet incluye la figura del cliente del taxi. En un entorno urbano, donde simular el transporte público del taxi, esto nos serviría. Para hacer que los transportes se desplacen por España, parando a recargar baterías en las estaciones disponibles, no es necesario incluir este agente en los experimentos. Podemos seguir dos aproximaciones para ocultar este agente:

1. Una opción sería mantener el agente del cliente, pero forzar a que, en los ficheros de configuración, los clientes partan exactamente de la misma ubicación inicial que los transportes. De esta manera, el proceso de recogida del cliente sería trivial e inmediato. La ventaja de esta alternativa es que no requiere ninguna codificación en el paquete del simulador, pues podemos escribir manualmente o través del generador los mismos lugares de salida para clientes y transportes. Aunque sí que requeriría una modificación pequeña, se podría modificar el comportamiento de los clientes para que lean la ubicación inicial de un vehículo al inicializarse y sobrescriban su propia ubicación inicial. En todo caso, la adaptación sería muy breve y evitaríamos tener que modificar el comportamiento de *TransportAgent* para que deje de negociar el viaje con clientes más alejados y tener que desplazarse primeramente hasta ellos.

   La desventaja de la alternativa anterior es que el cliente seguiría apareciendo en la simulación, por mucho que no tenga ningún rol en la misma. Esto implica que siga apareciendo en la hoja de resultados con estadísticas redundantes (el tiempo de viaje sería igual que el del transporte). Estas estadísticas deben ser calculadas en algún momento, para lo que los clientes necesitan recursos computacionales individual y globalmente. Esto resta recursos a los agentes verdaderamente importantes, los transportes y las estaciones, y además alarga el tiempo de ejecución, aunque en una medida muy poco perceptible. En resumen, el computador donde se simula quedaría ocupado innecesariamente procesando las acciones de los clientes. Por tanto, quedaría descartada esta aproximación al problema.

2. La alternativa escogida para esta adaptación del simulador es eliminar completamente la presencia del agente cliente. Ello no solo implica borrar el código relativo

a la estrategia y comportamiento de este agente, sino también todo el código relativo a la comunicación y negociación con el transporte. En caso de no eliminar estas partes del código, los vehículos quedarían eternamente esperando notificaciones de unos clientes que no existen, y las simulaciones no servirían para nada. A diferencia de la primera aproximación, esto requiere de más esfuerzo de codificación, pero a la larga tendremos un simulador y configuraciones más limpios y eficientes. Los recursos del ordenador se asignarán a los agentes relevantes y en los resultados no tendremos la parte correspondiente a los clientes. Los transportes no tendrán que negociar con ellos, por mucho que se encuentren en la misma ubicación y podrán iniciar el viaje al destino de inmediato.

Por tanto, para resolver esta modificación de SimFleet debemos eliminar la figura del agente cliente. Esto incluye borrar su código, y la parte correspondiente a la comunicación del transporte con ellos. También será necesario modificar el lector de los ficheros de configuración y el proceso de elaboración de los resultados.

3.2.2. Entorno interurbano

El simulador estaba inicialmente diseñado para simular el flujo de clientes y taxis en ámbitos municipales. Tenemos que ampliar el horizonte del simulador para que podamos experimentar sobre toda España y su red de carreteras interurbanas. Resolver este problema es sencillo, pues como hemos dicho, en temas de escalabilidad, el simulador está perfectamente preparado para soportar un entorno más allá del urbano. Lo único que vamos a necesitar es un nuevo mapa que introducir en el generador de estaciones, para que este pueda asignar debidamente las ubicaciones de estaciones y transportes dentro del país. El mapa está codificado en formato GeoJSON, que consiste en un polígono cuyos lados representan las fronteras de la región donde se ubican los agentes al principio y al final de una simulación.

En consecuencia, con la creación de un nuevo polígono, cuyos lados sean las fronteras de España (sin incluir Ceuta, Melilla, las Islas Baleares y las Islas Canarias) podremos indicar al generador de carga que puede utilizar todo el territorio español en la península para colocar agentes y sus respectivos destinos.

Nótese que el servidor de rutas que empleamos ya soporta la navegación por cualquier carretera del mundo, por lo que podemos seguir utilizando este servicio más allá de la ciudad de València.

3.2.3. Estrategia de *TransportAgent*: protocolo de trayecto y búsqueda de estaciones

El agente transporte es el que sufrirá más cambios para obtener el simulador que necesitamos para nuestros experimentos. Tal como hemos comentado con anterioridad, la figura del cliente va a desaparecer. En consecuencia, hay que adaptar el comportamiento de los vehículos para que, en vez de esperar la llamada de un cliente, se dirijan directamente al destino. Esto requerirá cambios en el estado de espera del transporte.

Otro de los aspectos de *TransportAgent* que requiere un remodelado es el algoritmo de búsqueda de estaciones. En la estrategia predeterminada del agente, a la hora de buscar una estación, se selecciona aquella que se encuentra más cerca del agente. En ciudad, esto es un protocolo sencillo, lógico y relativamente acorde con la realidad. Sin embargo, cuando queremos realizar un viaje fuera de un municipio, no podemos ir simplemente a la estación más cercana, porque esto puede suponer ir en la dirección contraria respecto

a nuestra meta. Hay que elaborar una estrategia que imite la realidad de los trayectos interurbanos, que consiste en escoger sobre la marcha una estación de carga o repostaje que no nos desvíe de nuestra ruta más rápida, a ser posible. Para ello, mantendremos la forma de calcular un viaje en el que cada transporte planifica antes de iniciar un desplazamiento su siguiente paso. Si no tiene suficiente autonomía para llegar a destino, deberá escoger como siguiente escala aquella estación que, estando dentro de su radio de alcance por autonomía, le acerque más a su destino final.

En caso de no poder alcanzar el destino, el simulador actual ignora la autonomía del vehículo para acercarse a la estación más cercana. Unido al nuevo protocolo de búsqueda de estaciones, debemos eliminar esta simplificación para poder acercarnos a la realidad del problema que estudia este trabajo: no siempre tendremos cobertura de estaciones suficiente para llegar a ciertas regiones de España. Por tanto, debemos añadir una comprobación de autonomía similar a la que se realiza antes de iniciar la recogida de un cliente. La diferencia es que, si el transporte no puede ir a su destino, pero tampoco a ninguna estación, debe abortar el proyecto y plasmarlo en las estadísticas de la simulación. Tenemos dos opciones para ilustrar esta situación:

La primera de ellas consistiría en adaptar el estado de espera del transporte para detectar este bloqueo del viaje y que el agente permanezca en este estado hasta que finalice la simulación. El motivo de plantearnos este acercamiento a la solución de este problema es que solamente habría que incluir esta casuística en la lógica de *TransportWaitingStatus*. Sin embargo, esta alternativa plantea problemas para diferenciar en la tabla de resultados si un vehículo ha quedado en este estado por haber completado su trayecto o por haberse quedado a mitad.

Por tanto, surge la idea de ampliar la máquina de estados del transporte diseñando un nuevo estado. Este estado debe representar la situación de un vehículo que no puede avanzar hacia el destino o hacia ninguna estación porque se quedaría sin energía de camino. Su denominación será *TransportDeadState*. Este estado será un estado final, es decir, un agente que entre en este estado permanecerá en él hasta el fin del experimento, no pudiendo transicionar a ningún otro bajo ningún concepto. La hoja de estadísticas del simulador muestra el estado de los vehículos al terminar una simulación, así que sería fácil visualizar, contabilizar y distinguir si un transporte ha podido completar su viaje. Además, necesitamos un nuevo estado que represente el último paso del viaje de un transporte, el camino hasta el destino final, ya que ya no dependemos del cliente. Su nombre será *TransportMovingToDestinationState*.

A modo de resumen, los cambios a efectuar sobre el agente transporte conciernen a la manera en la que se van a dirigir a su destino, que es directa, sin recoger ningún pasajero extra; y a la forma de seleccionar las estaciones, que es distinta a la que emplearíamos de estar en ciudad y debe adaptarse a la realidad de los viajes interurbanos. Cabe destacar que en este aspecto no es necesario realizar ninguna modificación de la estrategia de las estaciones, pues no tiene ninguna relación con los clientes y su comportamiento de cara al transporte debe ser el mismo.

En la figura 3.2 podemos visualizar el grafo de la máquina de estados diseñada para los experimentos.

Estrategia de *SimulatorAgent*: condiciones de finalización

Cuando hemos explicado la estrategia de *SimulatorAgent* hemos visto que una de sus responsabilidades fundamentales es la de supervisar la simulación y determinar la finalización de la misma. En la versión actual de SimFleet existen dos situaciones en que este agente detiene la simulación y desmonta su infraestructura, dando paso a la recopilación

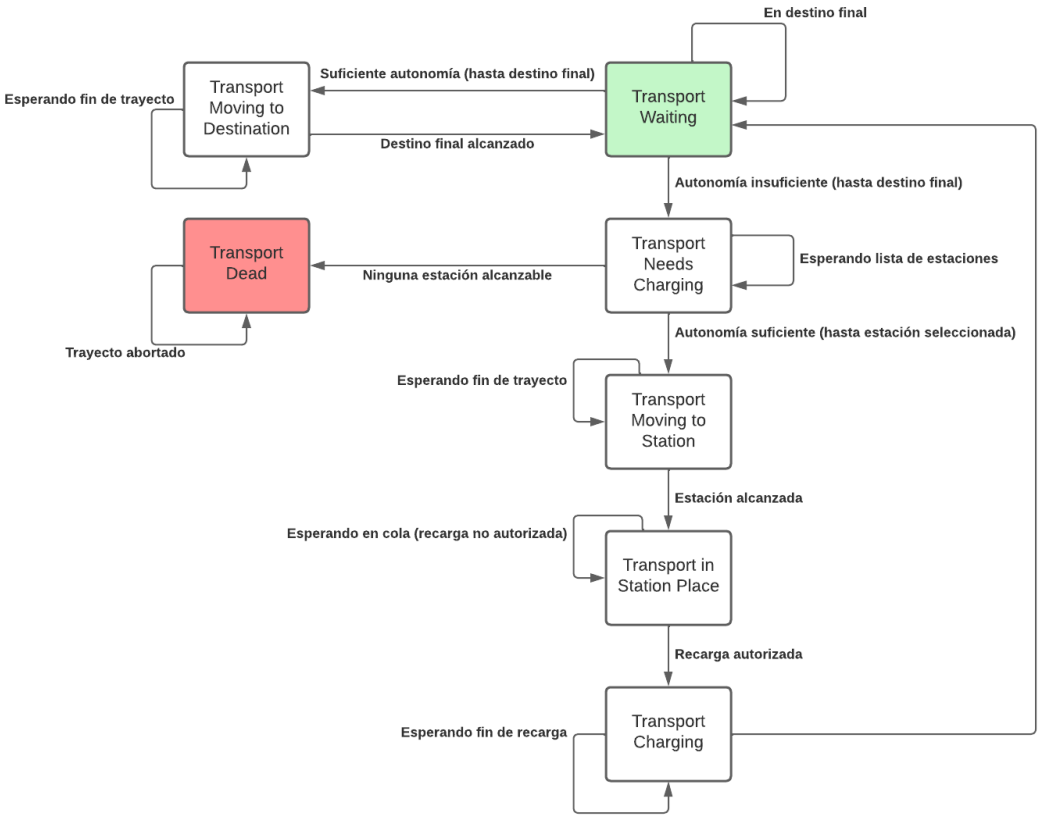


Figura 3.2: Máquina de estados de *TransportAgent* diseñada para la versión a desarrollar del simulador

En verde, el estado inicial. En rojo, el estado final. Elaboración propia

e impresión de estadísticas. Una de ellas es que se sobrepase el tiempo máximo de simulación. Esta condición de finalización es muy importante para evitar que, en caso de error o interbloqueo de los agentes, una simulación no acabe nunca. Por tanto, vamos a mantenerla tal como está, aunque aumentaremos el umbral máximo para dar suficiente tiempo a los agentes a realizar trayectos mucho más largos y recargar baterías con mayor capacidad.

La otra condición es que todos los clientes hayan llegado a su destino mediante el transporte en taxi. Ya hemos comentado que el cliente va a desaparecer, por tanto, no serviría de nada mantener esta comprobación. Necesitamos modificar la condición para que siga cumpliendo la misma función que antes: determinar cuándo todos los agentes han terminado de realizar su cometido.

Para ello nos tenemos que preguntar cuándo se cumple esto en nuestros experimentos. Queremos obtener estadísticas relativas a los viajes completados de los transportes. Por tanto, pensando en los transportes individualmente, ellos completan su función cuando

1. han llegado a su destino ó
2. han fracasado en el intento, es decir, han abortado el viaje por falta de autonomía para dar el siguiente paso.

Por tanto, la nueva condición de finalización debe consistir en evaluar para todo transporte si se cumplen ambos aspectos. Al mismo tiempo se debe controlar el cronómetro de la ejecución por si se sobrepasa el límite.

3.2.4. Generadores de carga

Los generadores de carga deben adaptarse de manera acorde con los cambios del simulador y el *input* que deben recibir a partir de la instauración de la nueva versión. Con la eliminación de *CustomerAgent*, los nuevos ficheros de configuración deben de omitir el registro de los clientes. Esto plantea un problema, pues son los clientes los que contienen el destino al que se dirigen los transportes. Bastará con migrar ese atributo al diccionario de los transportes, para que ellos mismos contengan la información completa relativa al origen y destino de sus viajes. Consecuentemente, en SimFleet se tendrá que adaptar la lectura de los atributos de los transportes para que sean ellos quienes almacenen esas ubicaciones.

Además, una vez tengamos actualizadas todas las estrategias, hay que indicar al generador que las incluya en los nuevos ficheros de configuración.

Respecto a la generación de la configuración de los experimentos sobre la distribución Tesla, disponemos de una tabla Excel con los nombres, coordenadas y voltajes de las estaciones existentes en España. Necesitamos diseñar un programa que transforme este fichero y convierta los datos contenidos en él en un diccionario de estaciones, para poder insertarlo en el fichero de configuración pertinente.

3.3 Tecnología utilizada

La implementación de los cambios a efectuar en el simulador se realizará, por temas de compatibilidad, manteniendo el mismo lenguaje en el que están codificados todos los programas, tanto los del simulador como los scripts de los generadores. El entorno

de desarrollo más intuitivo y conocido por el autor es PyCharm[1], la herramienta de la empresa JetBrains para desarrolar en Python. Concretamente, emplearemos la versión gratuita, pues no es necesario emplear las herramientas avanzadas.

En cuanto al control de versiones[2], haremos uso de Git, un sistema muy utilizado y sencillo de manejar, que además se integra dentro de PyCharm, lo que facilita enormemente su utilización.

[1] https://www.jetbrains.com/pycharm/
[2] ver glosario

CAPÍTULO 4
Desarrollo de la solución propuesta

Prosiguiendo con las fases de un proyecto informático, el siguiente paso es la implementación de todas las adaptaciones previamente diseñadas. Hay varios aspectos que cambiar en SimFleet antes de poder realizar los experimentos. Entre otras cosas, debemos eliminar un agente y reformar el comportamiento de otro, modificar los scripts de generación de carga etc.

Vamos a dividir este capítulo en tres secciones. En la primera de ellas veremos cómo se han aplicado los cambios relativos al simulador en sí. En la segunda explicaremos la implementación de los cambios diseñados en el paquete de generadores de carga. Por último, relataremos cómo se han empleado los generadores de carga modificados para crear las configuraciones de todos los experimentos y cómo se han ajustado los parámetros de los vehículos para ajustarse a la realidad.

4.1 Adaptación del simulador

El primer paso a la hora de realizar cambios sobre el simulador ha sido copiar la estrategia de los transportes en un nuevo fichero y sincronizarlo con Git. Así es sencillo llevar un histórico de los cambios realizados sobre el código. Cada vez que se ha introducido un paquete de medidas, se han confirmado y sincronizado con el repositorio, posibilitando la revisión de los cambios a posteriori y permitiendo deshacerlos de golpe si hay cualquier error.

En primer lugar se ha llevado a cabo la eliminación de toda la lógica perteneciente al agente cliente. Evidentemente, el código que refiere al comportamiento propio de los clientes en *SimFleet* ha sido borrado completamente. Su desaparición del simulador desencadena la eliminación de todos los protocolos de interacción con otros agentes. El único agente que se comunica directamente con *CustomerAgent* es el transporte. Recordemos que el transporte empieza una carrera cuando recibe una solicitud de un cliente a través del mánager de flotas, que actúa como mero intermediario. Por tanto, el agente transporte es el siguiente en ser modificado.

Antes de cambiar el comportamiento del agente en sí, como hemos anunciado en el capítulo dedicado al diseño de las modificaciones, vamos a añadir los atributos, eventos y estados nuevos que necesitamos. En primer lugar, añadimos sendas variables nuevas. Necesitamos dos nuevos atributos de clase para *TransportAgent*, de tipo booleano*, que indiquen si el vehículo ha completado su ejecución. Uno de ellos recibirá el nombre de *final_destination_reached* e indicará si el transporte ha llegado correctamente a destino. El otro, *is_dead*, si el vehículo se ha detenido durante el trayecto por haber quedado bloqueado. Sincronizaremos el valor de la primera variable con un nuevo evento (*trans-*

port_in_final_destination_event), que se lanzará a la finalización del comportamiento de movimiento del transporte, avisando al mismo de que puede volver al estado de espera.

El segundo se evaluará a *True* cuando el agente entre en el nuevo estado *TransportDeadState*. Este estado es final, el agente no regresará a otro estado una vez entre en este. Lo único que se hará en este estado es activar el atributo mencionado, permaneciendo en el mismo estado. Esta variable se leerá a la hora de calcular las estadísticas del proceso, obteniendo la proporción de trayectos abortados por los transportes.

Asimismo, la ubicación final del trayecto a realizar ya no vendrá dada por un cliente. Será el propio vehículo el que, desde una variable propia obtenga el destino al introducir un archivo de configuración para calcular su ruta.

Aprovechamos en este punto también para crear el estado para el último trayecto de los transportes: *TransportMovingToDestinationState*. Lo hemos considerado necesario para distinguirlo del estado en el que se viaja a una estación de recarga. En este estado, no habrá ninguna lógica relativa a los clientes, y se lanzará el evento *transport_in_final_destination_event* justo antes de regresar al estado de espera.

Por tanto, en el estado de espera, en vez de aguardar la llegada de una petición, pasaremos directamente a observar el valor de los dos nuevos atributos. En caso de haberse determinado la llegada del cliente a su destino, se permanece en el mismo estado. Si se ha decretado el bloqueo del vehículo previo levantamiento del evento, el agente tendrá que avanzar al estado de "trayecto abortado", y cesará inmediatamente en su intento de alcanzar la meta. Si no se comprueba ninguna de las dos situaciones anteriores, o bien el transporte se encuentra en la casilla de salida o acaba de terminar de recargar baterías. En cualquier caso, tendrá la carga al máximo y tendrá que averiguar a dónde dirigirse. En primer lugar, se solicita al servidor de rutas el camino hasta la ubicación final configurada en el agente. El agente debe comparar la distancia restante con la que puede recorrer sin tener que recargar. Si la autonomía es mayor o igual, el siguiente estado del transporte será el nuevo *TransportMovingToDestinationState*. Si la autonomía no es suficiente, el agente pasará a *TransportNeedsChargingState*.

En este estado encontramos otro gran escollo de la fase de implementación, el algoritmo de búsqueda de la mejor ubicación donde recargar baterías. El algoritmo se ha rehecho prácticamente al completo, pues recordemos que no nos sirve la estrategia anterior porque ya no nos encontramos en un entorno urbano. Se mantiene la búsqueda de estaciones por medio del directorio, pero no la forma de elegir a cuál de todas ellas acudir. Se adjunta el extracto de la lógica del estado que contiene el nuevo protocolo de selección:

```
1    reachable_stations = []
2    for key in self.agent.stations.keys():
3        dic = self.agent.stations.get(key)
4        if self.has_enough_autonomy(self.get("current_pos"),
5                                    dic['position']):
6            reachable_stations.append((dic['jid'], dic['position']))
7    if len(reachable_stations) == 0:
8        self.set_next_state(TRANSPORT_DEAD)
9        return
10   next_station = min(reachable_stations,
11                      key=lambda x:
12                      distance_in_meters(x[1], self.agent.final_dest))
13   if self.get("current_pos") == next_station[1]:
14       self.set_next_state(TRANSPORT_DEAD)
```

```
15          return
16      station_id = next_station[0]
17      self.agent.current_station_dest = (
18          station_id,
19          self.agent.stations[station_id]["position"]
20      )
```

En la primera linea inicializamos una lista donde almacenaremos las estaciones de recarga. A continuación, y hasta la línea 6, podemos ver cómo el agente lee las estaciones que ha obtenido previamente del directorio. La información de cada estación tiene el formato de un diccionario, con su posición, número de terminales etc. De él leemos su ubicación y calculamos la distancia por carretera respecto de la propia ubicación del transporte. Solamente añadiremos la estación a la lista definitiva si la autonomía es suficiente para alcanzarla.

La siguiente cuestión es comprobar que alguna de las estaciones esté efectivamente accesible. Esto se realiza en las líneas 7 a 9, controlando que *reachable_stations* no esté vacía. Ello simbolizaría que ninguna de las estaciones en toda España queda a mano del agente y que este tiene que abortar el viaje, transicionando al estado correspondiente.

Una vez ratificado que tenemos la posibilidad de recargar, es el momento de escoger una de las estaciones obtenidas. Mediante la función lambda* de las líneas 10 a 12, recorremos las estaciones al alcance del agente, quedándonos con aquella que minimiza la distancia en línea recta con el destino final del agente. Nótese que cada estación es una tupla de dos valores: el identificador de la estación y sus coordenadas en el mapa.

Antes de finalizar el algoritmo se tiene que examinar una última condición (líneas 13 a 15). Puede darse la situación en la que un transporte haya seleccionado como siguiente estación aquella en la que ya se encuentra. En la práctica, esto puede producirse en numerosas ocasiones, sobre todo cuando hay pocas estaciones en el mapa. Si un transporte se encuentra en una zona donde no hay ninguna estación a su alrededor (en el radio de su autonomía máxima), sin esta comprobación detectará que la estación que más le aproxima a su destino es la misma en la que ya se encuentra. Este hecho provoca que los vehículos que se encuentren en esta tesitura recarguen sus baterías eternamente en el mismo sitio, aun teniendo las baterías a plena capacidad. Para evitar un bucle infinito, se deriva al transporte al estado de finalización temprana del viaje, *TransportDeadState*, pues nunca llegará a su destino. Si, efectivamente, se trata de una estación distinta, está garantizado que estará más cerca de la ubicación final. Se establece, en consecuencia, la estación como destino del siguiente "salto".

Además, con este chequeo previo cumpliríamos con otra de las premisas que se habían establecido en la fase de diseño. Antes, los taxis ignoraban la distancia hasta su siguiente estación por simplicidad. El camino hasta una estación en entornos interurbanos puede ser mucho más grande que en ciudad, por lo que no podemos considerar válida esta simplificación. Con el algoritmo que hemos elaborado, es imposible que se seleccione una estación a la que el transporte no pueda llegar por medios propios porque son descartadas en las primeras lineas.

Con esto se completa la implementación de la nueva versión de *TransportAgent*. A continuación, se procede a detallar todas las adaptaciones sobre *SimulatorAgent*. En el capítulo anterior hemos explicado que consisten en programar correspondientemente las condiciones bajo las que SimFleet debe dar por terminada una ejecución. En este aspecto, el simulador comprueba de manera periódica (cada medio segundo) una condición booleana para detectar que se cumplen todos los requisitos para dar el experimento por

finalizado. Podemos ver a continuación cuál es el resultado del desarrollo de la nueva función:

```
1    def is_simulation_finished(self):
2        if not self.config.max_time:
3            return self.all_transports_in_destination_or_dead()
4        return self.time_is_out()
5                or self.all_transports_in_destination_or_dead()
```

En este extracto de código, en primer lugar, se controla la existencia del límite temporal máximo en el fichero de configuración. Recordemos que este parámetro es opcional, pero muy recomendable. Si no se ha especificado ningún umbral de tiempo, el resultado depende de si todos los vehículos han terminado su viaje:

```
1    def all_transports_in_destination_or_dead(self):
2        if len(self.transport_agents) > 0:
3            return all([(transport.is_in_final_destination()
4                         or transport.is_dead)
5                         for transport in self.transport_agents.values()])
6        else:
7            return False
```

Para constatar este hecho, el programa necesita, primeramente, evaluar si existen transportes en la simulación. Si por cualquier razón no hubiera agentes de este tipo, se devuelve *False* a la comprobación global. Por tanto, la simulación se tendrá que detener manualmente, pues siempre se permitirá que continúe la ejecución. Lo normal es que haya al menos un transporte. Para todos los que se hayan configurado, se comprueba que el vehículo o bien se encuentra esperando en la misma ubicación que su destino, o que ha tenido que abortar el viaje y la variable correspondiente (la que hemos añadido al agente) evalúa a *True*. La sentencia *all* deja de recorrer la lista de agentes tan pronto como encuentra el primero que no cumple con alguna de las dos casuísticas.

Volviendo a la comprobación global, queda por analizar el código que revisa el cronómetro de la simulación. Esta parte es una sencilla comparación entre el tiempo de ejecución y el máximo. Como las sentencias booleanas están cortocircuitadas* en Python, siempre se comprueba en primer lugar la condición temporal, que es poco costosa comparada con la evaluación del estado individual de cada uno de los transportes. Por tanto, en el momento en el que se sobrepasa el límite temporal, se devuelve *True*, dando paso al desmantelamiento de los agentes y, por ende, de la simulación.

4.2 Adaptación de los generadores de carga

En cuanto al paquete de generadores de archivos de configuración de SimFleet y sus agentes, también hay varios programas que debemos adaptar a las necesidades propias de nuestro proyecto. En primer lugar, al igual que en el simulador, debemos de evitar crear agentes de tipo cliente al crear configuraciones. Para ello, hay que encontrar y eliminar aquella parte que, de manera iterativa, creaba clientes en diccionarios individuales y los añadía a una lista. Con la eliminación total de los *CustomerAgent* estaríamos eliminando un parámetro crucial para nuestros experimentos, que es el destino final. La solución adoptada ha sido migrar este atributo a los *TransportAgent*, de manera que dispongan desde el comienzo de su ciclo de vida el lugar desde el que parten y a dónde

tienen que dirigirse. Por tanto, se modifica la creación de transportes, para que además de escoger una localización inicial aleatoria, también calculen una localización de destino aleatoria. Antes, esta ubicación se asignaba al cliente. Por coherencia, al remover la lógica relativa a los clientes, ya hemos tenido en cuenta que los transportes deben leer esta información de su propio atributo de clase en el *TransportWaitingState*.

Las estrategias de todos los agentes se encontraban incrustados directamente en el generador, pues son las estrategias por defecto. Para que el simulador utilice la versión nueva y adaptada de cada agente, se ha especificada en cada caso la nueva ubicación en el código de las estrategias correspondientes.

Para poder realizar los experimentos relacionados con la red de estaciones propietaria de Tesla, hemos extraído de su sitio web[1] y depositado en una hoja Excel toda la información necesaria. Para poder emplear dichos datos, se ha programado un generador adicional que se encargue de traducir el fichero Excel con todos los datos de las estaciones en un diccionario JSON equivalente. La librería Python utilizada en este generador es pandas*, cuya función es precisamente dotar de una herramienta para la lectura y escritura de ficheros Excel. Leyendo el archivo línea por línea desde el generador, se va rellenando el diccionario de estaciones con su nombre, coordenadas en el mapa y potencia. Para los experimentos sobre esta distribución, ya que disponemos de información fácilmente accesible y exacta, se han configurado los *StationAgents* con el número de terminales y el voltaje que tienen en la realidad.

Por último, todos los generadores de estaciones obligan a la concreción de un mapa para poder saber dentro de qué limites puede ubicar los agentes. El mapa de València actual no nos sirve, así que se ha exportado un nuevo polígono para toda la Península Ibérica salvo Portugal y Gibraltar. Esta vez, el instrumento de creación ha sido la página interactiva `www.GeoJSON.io`, donde se puede dibujar la silueta de España y convertir los puntos fronterizos en un polígono en formato GeoJSON. Todas las configuraciones de agentes se generan a partir de este mapa.

4.3 Configuración de los experimentos

En este momento ya tenemos todas las herramientas programadas para nuestros fines. Falta por determinar los metadatos de cada agente, es decir, sus características propias además de la ubicación, que viene calculada por los generadores. Procedemos a detallar qué valores se han asignado a los atributos configurables de cada agente:

- ***FleetManagerAgent***
  - **Nombre y tipo de flota**: "IUecar"[2].
- ***SimulatorAgent***
  - **Tiempo máximo de ejecución**: 9.000 segundos, que equivale a dos horas y media. Suficiente para que los transportes lleguen a tiempo a destino con cierto margen.
- ***StationAgent***
  - **Número de terminales**: solamente uno, excepto en la distribución Tesla, en la que el número de terminales viene determinado por la información obtenida en su sitio web.

[1] `https://www.tesla.com/es_ES/findus/list/superchargers/spain`
[2] Coche eléctrico interurbano, por sus siglas en inglés

- **Potencia de carga**: 3 kilómetros de autonomía por segundo. Se ha considerado utilizar como plantilla las estaciones de Tesla de menor voltaje, de 150 kilovatios hora. Según la empresa, sus denominados "supercargadores" pueden llegar a cargar una capacidad de 275 kilómetros de autonomía en un cuarto de hora[3]. Por regla de tres, eso significa que la autonomía aumenta en 300 metros cada segundo. Multiplicado por el factor de aceleración de la simulación, obtenemos una potencia de 3 kilómetros por segundo.

  Por simplificación, se ignora el hecho de que las baterías no se cargan a la misma velocidad durante todo el proceso. Es conocido que, a partir del 80 %, la curva de recarga decrementa su pendiente, tardando más en finalizar que si fuera lineal. Esta simplificación solo afectará al tiempo de viaje de los transportes, en una magnitud poco significativa, desde luego irrelevante para esta investigación.

- ***TransportAgent***

  - **Velocidad**: 1.000 kilómetros por hora. De esta manera, y estimando una velocidad media de 100 kilómetros por hora en carretera, aceleramos el transcurso de la simulación por el factor 10 respecto a la realidad. Los trayectos de los transportes fueron generados con mínimo seiscientos kilómetros de distancia en línea recta. Por tanto, un desplazamiento sin paradas intermedias llevaría menos de una hora. Dependiendo del número de recargas de batería, esta duración se verá incrementada en mayor o menos magnitud.

    En consecuencia, todos los parámetros temporales se han multiplicado por el mismo factor.
  - **Autonomía y autonomía máxima**: coinciden, pues suponemos que los vehículos tienen un punto de recarga en su lugar de origen (casa, garaje, etc.). Su valor varía según el experimento.
  - **Tipo de flota**: "IUecar".

Para todo experimento se han configurado estos atributos. Entre experimentos, se mantiene constante el número de vehículos (y sus trayectos) y la potencia de las estaciones. Sin embargo, varía la autonomía de los transportes, el número de estaciones y la distribución de las mismas. En relación a la autonomía máxima de los transportes, se ha decidido generar experimentos para cuatro capacidades distintas: 50, 100, 200 y 400 kilómetros de autonomía.

El equivalente al Real Automóvil Club de España (RACE) en Noruega, el *Norges Automobil-Forbund*[4] puso a prueba la capacidad de cuarenta modelos de coches eléctricos distintos para averiguar su autonomía efectiva en la práctica[5]. La media de la autonomía que mostraron asciende a 473 kilómetros. Nos hemos basado en este valor para estimar los valores para la capacidad autónoma de nuestros coches interurbanos. Como veremos en el capítulo 5, experimentar con mayores autonomías no es relevante, pues los resultados ya son muy buenos para la máxima capacidad que hemos considerado. Además, la intención es obligar a los transportes a realizar al menos una parada, por lo que no pretendemos aproximarnos demasiado a los 600 kilómetros de distancia mínimos para cada trayecto. En la figura 4.1 se muestra el mapa elaborado con las ubicaciones de origen y destino de los 500 vehículos. Es difícil distinguir los trayectos que realizan los transportes por la cantidad de elementos impresionados sobre el mapa, pero se pretende ilustrar la aleatoriedad de los viajes y el hecho de que se cubre todo el suelo español.

[3] https://www.tesla.com/es_es/supercharger
[4] NAF, Asociación Noruega del Automóvil en español
[5] https://nye.naf.no/elbil/bruke-elbil/test-rekkevidde-vinter-2022#rekkeviddetest

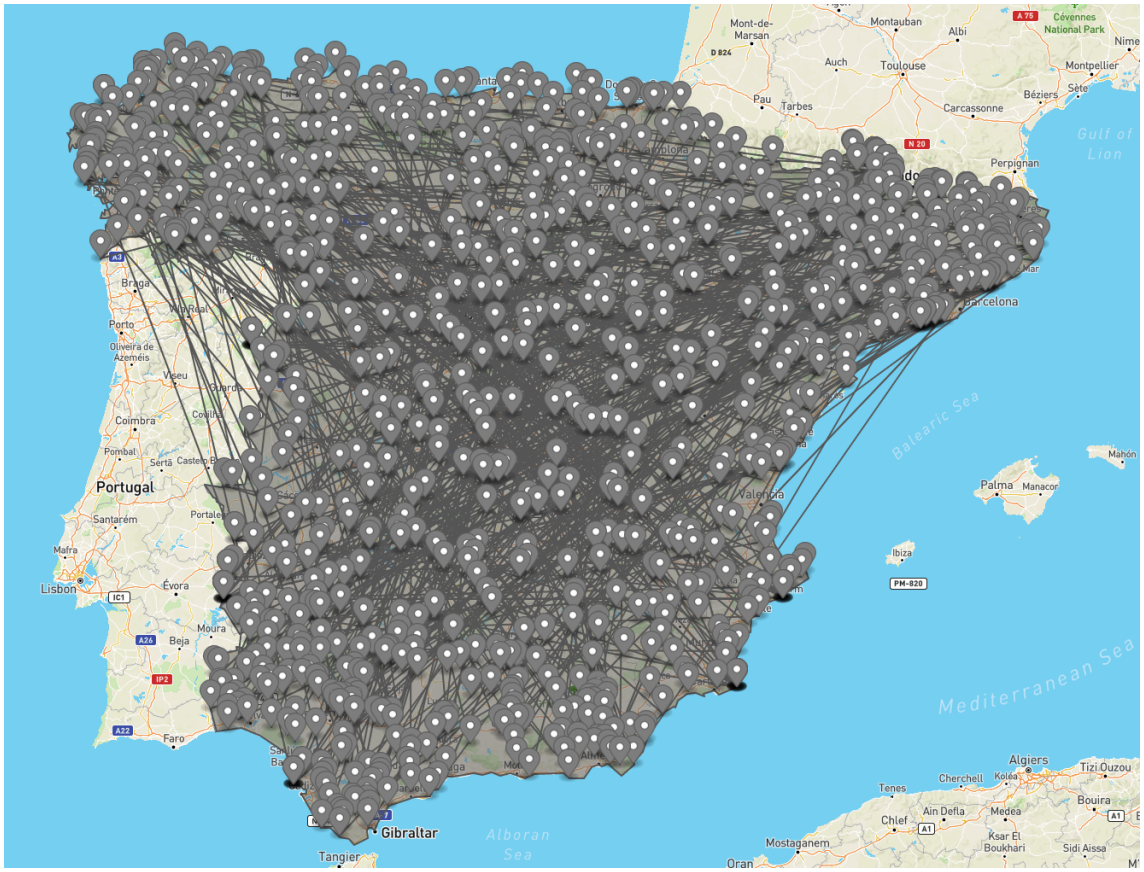


Figura 4.1: Trayectos aleatorios configurados en el simulador (500 vehículos)

En cuanto a la cantidad de estaciones, se ha determinado una horquilla de entre 50 y 1.000 estaciones (o puntos de recarga, según el caso) a emplazar. Los saltos en este número serán de 50, 100, 250, 500, 750 y 1.000 estaciones. Del mismo modo que con la autonomía de los vehículos, veremos que los experimentos se saturan de cara a las estadísticas finales con más de 1.000 estaciones. Además, en la práctica, la instalación de más de 1.000 puntos de recarga es irreal y poco eficiente en términos económicos.

Con todo ello, realizaremos experimentos para cinco distribuciones, con seis cantidades de estaciones y cuatro gamas de vehículos distintas. Cada combinación es importante, por lo que, el producto cartesiano* de todas las opciones genera hasta 120 configuraciones distintas a introducir en SimFleet.

CAPÍTULO 5
Resultados experimentales

Una vez tenemos todas las adaptaciones necesarias implementadas en SimFleet y todos los ficheros de configuración generados, es el momento de arrancar las simulaciones. Al finalizar las mismas, SimFleet retorna unos determinados ficheros de resultados. Estos ficheros se dividen en dos tipos. Por un lado recibimos dos ficheros de texto: el de la salida estándar del simulador (con la tabla de resultados predeterminada) y un fichero de texto de salida de error. Por el otro lado, y de manera opcional, se genera un fichero en formato JSON, que no contiene más que la tabla de la salida estándar expresada en una notación que facilita enormemente el procesamiento de datos automatizado en Python u otros lenguajes. Del mismo modo, ayuda a exportar los datos en tablas Excel. En nuestro caso, como no hace falta realizar una gran cantidad de cálculos o manipulaciones, sino tablas y gráficas para visualizar los datos, se ha optado por la exportación a Excel.

A continuación vamos a analizar los resultados obtenidos en las simulaciones. Recordemos que se han realizado experimentos sobre cada tipo de distribución por separado, variando tanto la autonomía máxima de los transportes como la cantidad de estaciones sobre el mapa. En consecuencia, se va a realizar un análisis detallado de los resultados para cada distribución de manera independiente. Se indicará en cada caso el efecto que suponen los cambios de las variables independientes: la autonomía y el número de estaciones. Por su parte, las variables que se van a emplear para valorar la bondad de cada distribución son la eficacia y la eficiencia.

La eficacia se interpretará mediante la proporción de vehículos que no pueden completar el trayecto planeado, pues es una manera de expresar si la distribución cumple su objetivo principal: que los transportes puedan realizar trayectos interurbanos en sí. Se calcula dividiendo el número de transportes que se han declarado "bloqueados" y han abortado el viaje entre quinientos, el número de transportes total.

En cuanto a la eficiencia, la discerniremos observando la desviación total media que tienen que realizar los conductores para desplazarse a cada estación en la que se detienen de camino al destino. De esta manera, sabremos si los mencionados trayectos requieren un mayor esfuerzo por parte de los conductores o si se pueden realizar sin problema. Calculamos la desviación media total de la siguiente manera:

$$\frac{\sum_{500-1}^{i=0}(d_i - D_i)}{\sum_{500-1}^{i=0} 1} \; \forall i \mid S_i \neq \text{"}TRANSPORT\_DEAD\text{"}$$

Donde i representa el identificador de cada uno de los 500 transportes, d representa la distancia recorrida por el vehículo i, D la distancia recorrida por el vehículo i con autonomía infinita (sin realizar recargas) y S el estado en que se encuentra el agente i (ver máquina de estados). En pocas palabras, para realizar el cálculo, solo se toman las distan-

		Eficiencia	
		Baja	**Alta**
Eficacia	**Baja**	En general, los vehículos se quedan sin energía en algún punto del camino. Los que logran llegar a destino, en total, tienen que desviarse muchos kilómetros para encontrar las estaciones que les acercan al destino.	En general, los vehículos se quedan sin energía en algún punto del camino. Los que logran llegar a destino, en total, se desvían pocos kilómetros para encontrar las estaciones que les acercan al destino.
	Alta	En general, los vehículos consiguen completar su trayecto, pero, en total, tienen que desviarse muchos kilómetros para encontrar las estaciones que les acercan a su destino.	En general, los vehículos consiguen completar su trayecto y, en total, se desvían pocos kilómetros para encontrar las estaciones que les acercan a su destino.

Tabla 5.1: Guía de ejemplo para el análisis de resultados experimentales

cias recorridas por vehículos que al finalizar la simulación no se encuentran declarados como "bloqueados" (pues esos vehículos contienen distancias parciales hasta el momento en el que abortaron su trayecto). Del resto, se calcula el promedio de la diferencia entre la distancia recorrida por los vehículos y la distancia óptima.

Para facilitar la comprensión de las incógnitas que vamos a interpretar se puede consultar la guía de ejemplo en la tabla 5.1.

5.1 Distribución aleatoria

Comencemos por la distribución aleatoria de estaciones de recarga. Recordemos que en este caso, como el nombre indica, la ubicación de cada estación ha sido generada de manera arbitraria, sin seguir ningún patrón. Cabe indicar nuevamente que todo experimento con mismo número de estaciones en total tiene las mismas estaciones en las mismas localizaciones. Esto es, se varía el nivel de autonomía de los vehículos, pero se mantiene constante el número de estaciones y su localización. A modo de ejemplo, vemos cómo se han colocado las estaciones en los experimentos con 50, 250, 500 y 1000 estaciones en la figura 5.1.

La información obtenida en el simulador y que se va a comentar seguidamente se encuentra en la tabla 5.2.

5.1.1. Eficacia de la distribución

En primer lugar, analicemos el porcentaje de vehículos que no consiguen alcanzar su destino en algún momento del trayecto, para cada escalón de autonomía y número de estaciones colocadas (ver figura 5.2).

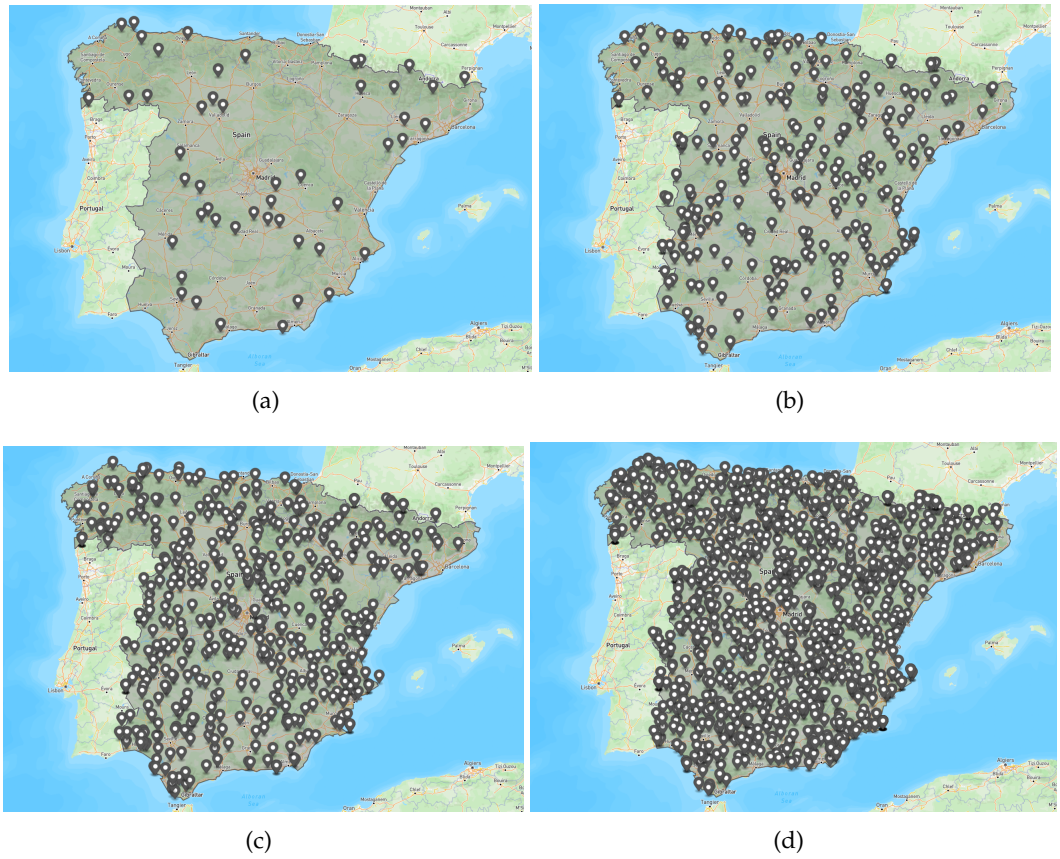
Apreciamos a primera vista el alto porcentaje de trayectos abortados para el nivel más bajo de autonomía de nuestras simulaciones. Para coches que no alcanzan más de 50 kilómetros sin recargar, haría falta una cantidad superior a las 250 estaciones para permitir algún trayecto entre ciudades. Se podría reducir el porcentaje de viajes interrumpidos de un 77.8 % a un 17.8 % aumentando el número de estaciones desde 500 a 1000 estaciones.

Si duplicamos la autonomía de los coches hasta los 100 kilómetros, observamos que sería conveniente colocar un número mayor de 50 estaciones para posibilitar el movimiento de los coches, pues con esa cantidad solo dos de ellos han alcanzado su meta (el 0.4 %). A partir de las 250 estaciones se obtiene un porcentaje de viajes cancelados muy aceptable, con un 5 %. Con 750 estaciones, todos los trayectos quedan cubiertos. Observa-

Distribución aleatoria		Trayectos abortados (en %)	Desviación media (en km)
50km	50s	100	-
	100s	100	-
	250s	100	-
	500s	77.8	416.09
	750s	42.6	380.09
	1000s	17.8	335.20
100km	50s	99.6	353.64
	100s	77	381.59
	250s	5	249.58
	500s	1.8	198.31
	750s	0	182.23
	1000s	2.2	172.74
200km	50s	52	245.65
	100s	0	166.28
	250s	0	124.90
	500s	0	108.60
	750s	0	99.78
	1000s	0	98.51
400km	50s	0	92.90
	100s	0	62.70
	250s	0	55.59
	500s	0	51.45
	750s	0	49.18
	1000s	0	46.79

Tabla 5.2: Tabla de resultados, distribución aleatoria

Eje horizontal: autonomía de los transportes (km) y número de estaciones (s)



(a) (b)

(c) (d)

Figura 5.1: Mapas de estaciones en la distribución aleatoria

(a) 50 estaciones; (b) 250 estaciones; (c) 500 estaciones; (d) 1000 estaciones

mos que añadiendo estaciones hasta alcanzar las 1000 vuelven a aparecer coches que se declaran "bloqueados". Esto puede deberse a la naturaleza del método de obtención de las estaciones. Como no se sigue ningún criterio matemático, un aumento de estaciones no tiene por qué implicar una mejora de la eficacia.

Volviendo a doblar la autonomía (200 kilómetros), tenemos que basta con colocar 100 estaciones de recarga para facilitar la llegada del 100 % de coches a su destino, pues con 50 estaciones la mitad (52 %) no podría alcanzarlo.

Finalmente vemos que, con los vehículos de mayor gama que hemos considerado en este trabajo, cuya autonomía asciende hasta los 400 kilómetros, es suficiente con el mínimo de estaciones para habilitar todos los viajes de la muestra.

5.1.2. Eficiencia de la distribución

En segundo lugar, fijémonos en la distancia media adicional que tienen que recorrer estos vehículos por parar a recargar baterías en las estaciones de recarga (figura 5.3).

Como se ha explicado con anterioridad, para los vehículos con las baterías más modestas, no tenemos datos de desviaciones en las simulaciones con un número menor de 500 estaciones. A partir de esta cantidad, ya podemos apreciar que, de media, hay que recorrer unos 416 kilómetros de más para realizar el trayecto correspondiente. Podemos reducirlo no de manera muy significativa hasta los 335 kilómetros si colocamos el máximo de estaciones en el mapa. Recordemos que nuestra muestra de coches hace trayectos que, como mínimo, abarcan una distancia de 600 kilómetros en línea recta, lo que, evi-

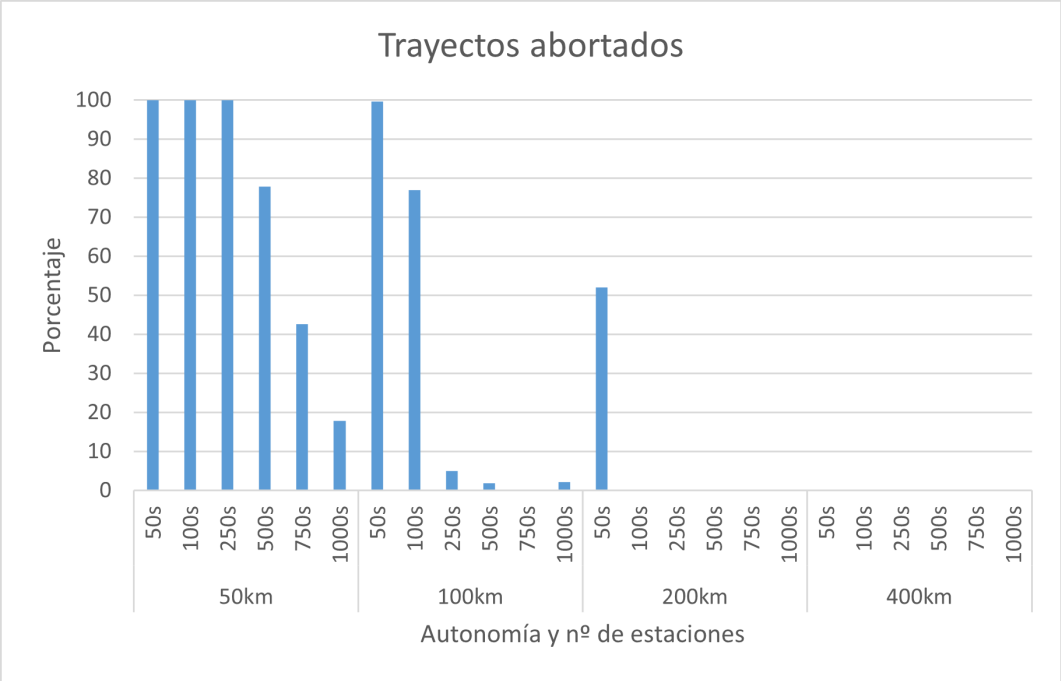


Figura 5.2: Porcentaje de transportes que abortan su trayecto, distribución aleatoria

dentemente, en carretera, acaba resultando en trayectos más prolongados. Eso significa realizar más de 12 paradas para recargar las baterías. Supone un rodeo por cada parada para salir de la autovía, internarse en una carretera secundaria, repostar y regresar a la ruta. Según los números, hay que añadir 50 kilómetros al contador por cada parada. En consecuencia, esto implica que, para este tipo de vehículo eléctrico, un viaje por España solamente tiene como horizonte la siguiente estación en la que hay que parar, pues solamente disponen de esos 50 kilómetros de margen.

Si nos fijamos en el siguiente escalón de autonomías, vemos que con 50 estaciones aleatorias, la desviación media por vehículo se ha reducido respecto a la simulación correspondiente al anterior escalón. Esto evidencia el efecto que tiene un aumento de la autonomía de los transportes: cuanto mayor es, más pueden avanzar en su trayecto óptimo antes de tener que buscar una estación donde repostar. Es decir, aumenta el margen u horizonte para encontrar una estación que se encuentre de camino al destino, reduciendo el número de paradas y de kilómetros adicionales. Como iremos viendo durante este capítulo, este fenómeno se manifestará en todas las distribuciones, cada vez que aumente la autonomía de los coches.

Concretamente, para 100 kilómetros de autonomía, tenemos que con 50 estaciones de carga repartidas aleatoriamente nos tendremos que desviar unos 354 kilómetros de media. Un valor similar al experimento que precede en este análisis. La curva desciende ahora de manera bastante pronunciada, reduciendo el desvío medio hasta los 198 kilómetros si colocamos 500 estaciones de recarga en la red española. Duplicando el número de estaciones, un hipotético viaje sólo vería aumentado el kilometraje en 173 kilómetros respecto al viaje óptimo.

Continuando con las simulaciones de vehículos de autonomía igual a 200 kilómetros, comenzamos con un desvío medio de 246 kilómetros si solamente ubicamos 50 estaciones de carga. Aumentando esta cifra hasta las 250 estaciones, el desvío medio se divide rápida y progresivamente por la mitad, hasta los 125 kilómetros, un valor bastante aceptable. Aún podríamos reducirlo más aumentando el número de estaciones, pero como mucho lograremos alcanzar los 99 kilómetros de desvío medio.

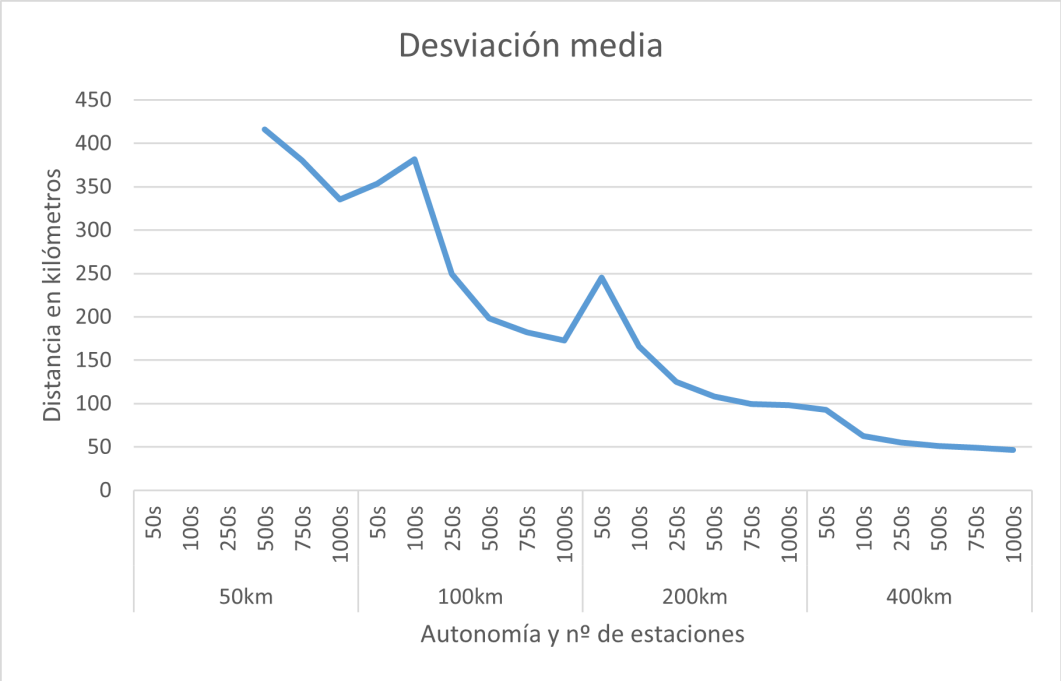


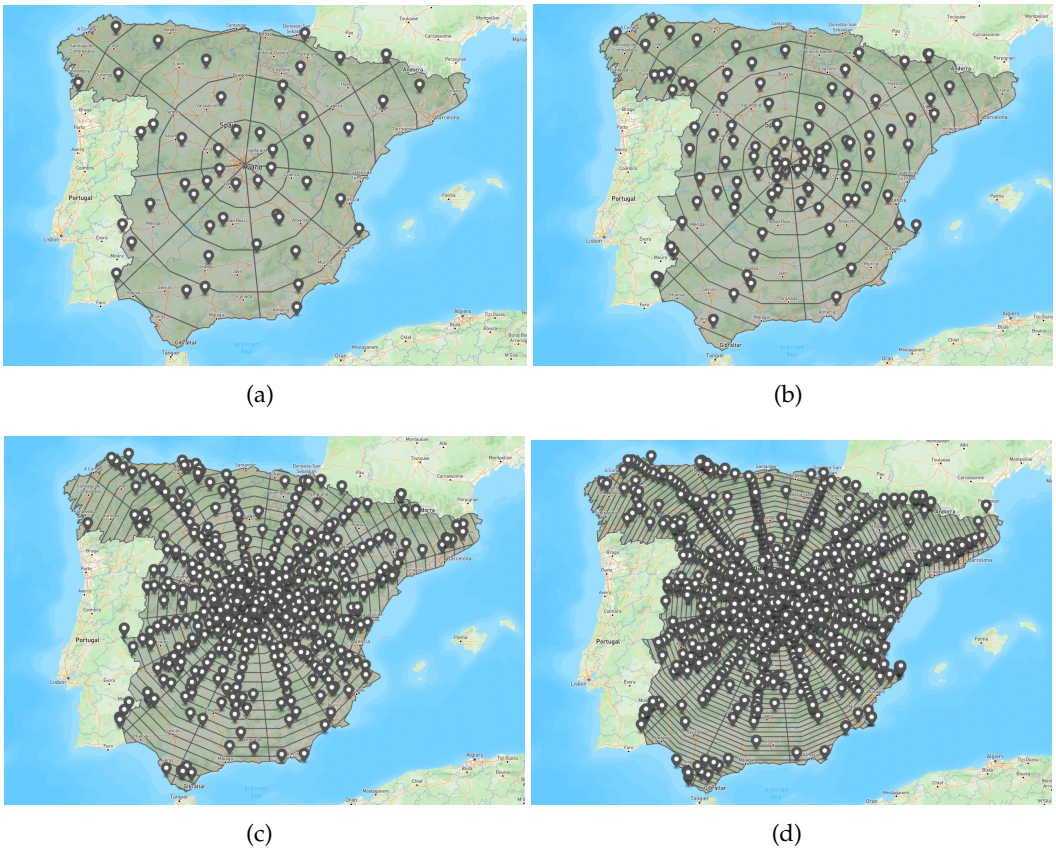
Figura 5.3: Desviación media respecto a la ruta óptima, distribución aleatoria

Por último, en relación con los vehículos de autonomía igual a 400 kilómetros y para el experimento realizado sobre el mapa con 50 estaciones aleatorias, el resultado es ligeramente inferior al del experimento anterior: 93 kilómetros. El valor mínimo de toda la distribución lo obtenemos con el máximo de estaciones desplegadas, con menos de 47 kilómetros de desvío medio total. Sin embargo, también podemos considerar el resultado con 100 estaciones como un resultado suficiente, pues solamente con 50 estaciones más que en el primer caso, el desvío medio alcanza algo más de 62 kilómetros.

5.1.3. Conclusiones

En resumen, la distribución aleatoria de estaciones de recarga no es una distribución del todo descartable. En cuanto a su eficacia, los vehículos con autonomías más bajas tendrían muchos problemas para moverse entre ciudades de España. Solo algunos trayectos concretos serían posibles. A partir de los 200 kilómetros de autonomía y con más de 100 estaciones, alcanzaríamos la accesibilidad total en el mapa español. En cuanto a la eficiencia, con autonomías menores que 200 kilómetros obtenemos unas distancias adicionales que los viajeros no podrían permitirse, pues para un viaje de al menos 600 kilómetros podrían llegar a tardar varias horas adicionales hasta alcanzar su destino. Los transportes de autonomía máxima sufren un desvío medio total por debajo de los 100 kilómetros, que ya es mucho más tolerable e incluso normal.

La conclusión que se extrae es que no es estrictamente necesario realizar un estudio exhaustivo de la ubicación idónea donde colocar las estaciones, pues colocándolas en cualquier punto del mapa obtenemos un resultado aceptable para las autonomías más altas. Sin embargo, estaríamos condicionando la posibilidad de realizar viajes interurbanos a tener un coche eléctrico más caro, con una batería más potente. La realidad es que, actualmente, los coches eléctricos tienen un precio muy elevado, y no todo el mundo puede permitirse una inversión tan grande. Estaríamos dejando a muchas personas sin poder moverse por el territorio español, una necesidad relativamente básica. Por tanto,



(a) (b)

(c) (d)

Figura 5.4: Mapas de estaciones en la distribución radial

(a) 50 estaciones; (b) 250 estaciones; (c) 500 estaciones; (d) 1000 estaciones

si queremos dar accesibilidad total a todo el parque automovilístico eléctrico de España, debemos explorar otras alternativas.

5.2 Distribución radial

Seguidamente vamos a realizar el análisis de los resultados correspondientes a las simulaciones en las que las estaciones de recarga de baterías han sido colocadas de manera radial (la tabla 5.3 reúne dichos resultados). A modo de recuerdo, esta distribución consiste en dividir el territorio sobre el que se experimenta en varios anillos, divididos a su vez en sectores, y colocar una estación en el interior de dicho sector. De esta manera, obtenemos un mapa en el que las estaciones forman círculos concéntricos (ver figura 5.4).

5.2.1. Eficacia de la distribución

Nuevamente, para extraer conclusiones acerca de la eficacia de esta distribución, vamos a observar detenidamente los porcentajes de trayectos que no se pueden llevar a cabo por falta de energía (ver figura 5.5).

A primera vista, observamos que el resultado de las simulaciones con vehículos de mínima autonomía es muy desfavorable, pues ni con el máximo de estaciones podríamos facilitar la llegada de la mitad de los trayectos configurados. En todo caso, con menos de 100 estaciones es imposible que algún coche pueda moverse entre ciudades. A partir de

Distribución radial		Trayectos abortados (en %)	Desviación media (en km)
50km	50s	100	-
	100s	100	-
	250s	96.8	333.44
	500s	80	300.28
	750s	66.8	315.44
	1000s	64.8	306.29
100km	50s	95.4	343.82
	100s	40.8	296.20
	250s	19.4	214.19
	500s	12.6	206.45
	750s	2.6	200.37
	1000s	4.6	192.47
200km	50s	0.4	203.52
	100s	1	141.64
	250s	0	115.67
	500s	0	112.40
	750s	0	106.12
	1000s	0	106.34
400km	50s	0	87.31
	100s	0	59.25
	250s	0	55.51
	500s	0	52.31
	750s	0	50.89
	1000s	0	51.57

Tabla 5.3: Tabla de resultados, distribución radial

Eje horizontal: autonomía de los transportes (km) y número de estaciones (s)

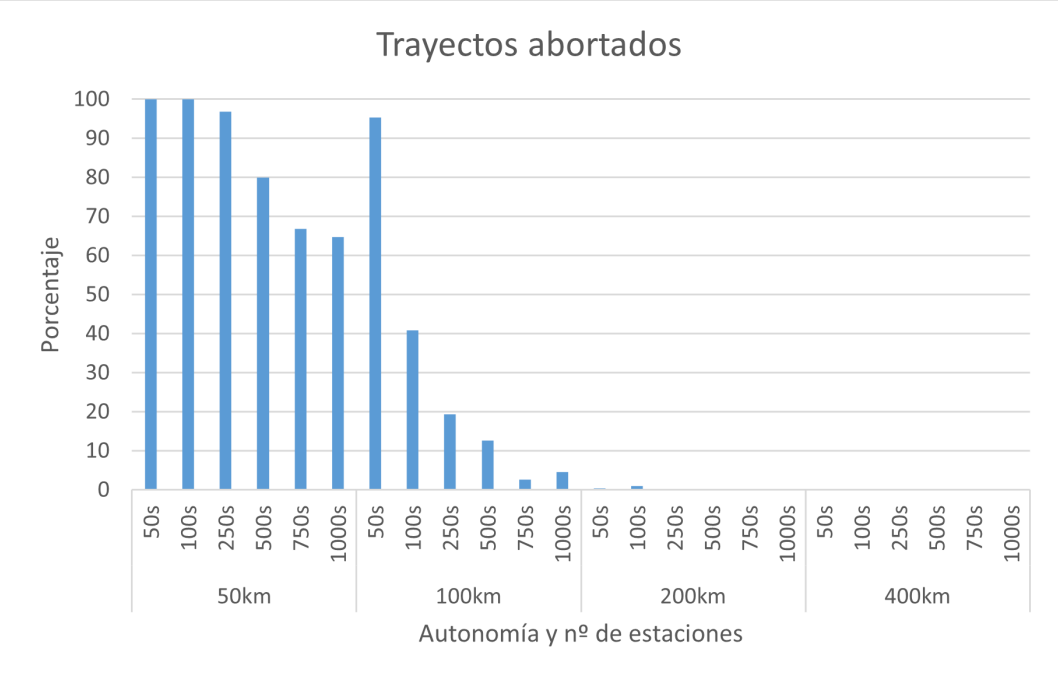


Figura 5.5: Porcentaje de transportes que abortan su trayecto, distribución radial

las 750 estaciones, la movilidad de los coches no cambia de manera significativa, situando el porcentaje de trayectos abortados aproximadamente en un 65 %.

Si avanzamos al siguiente peldaño en los niveles de autonomía, seguimos teniendo una cifra de viajes interrumpidos muy cercana al 100 %, lo que demuestra que la autonomía no está influyendo sensiblemente en la eficacia de la distribución. Es decir, ni con el doble de autonomía pueden desplazarse los coches con tan pocas estaciones. Sí es cierto que el porcentaje disminuye muy radicalmente aumentando el número de estaciones: hasta las 250 estaciones, el número de trayectos abortados se decrementa hasta el 19.4 %, que es un valor aceptable. El mejor valor se alcanza configurando 750 estaciones, con solo un 2.6 % de trayectos fallidos.

Continuando con las simulaciones con vehículos de 200 kilómetros de autonomía o más, prácticamente no hay ninguno de ellos que no logre llegar a destino (menos de un 1 %). Por tanto, los coches de más alta gama pueden recorrer España con esta distribución sin mayor problema.

5.2.2. Eficiencia de la distribución

Observemos pues lo que concierne a la eficiencia de la distribución, que se cuantifica con la distancia media que tienen que añadir los viajeros a su trayecto por tener que salir de la carretera para encontrar una estación donde recargar. Podemos visualizar la evolución de esta medida en la figura 5.6.

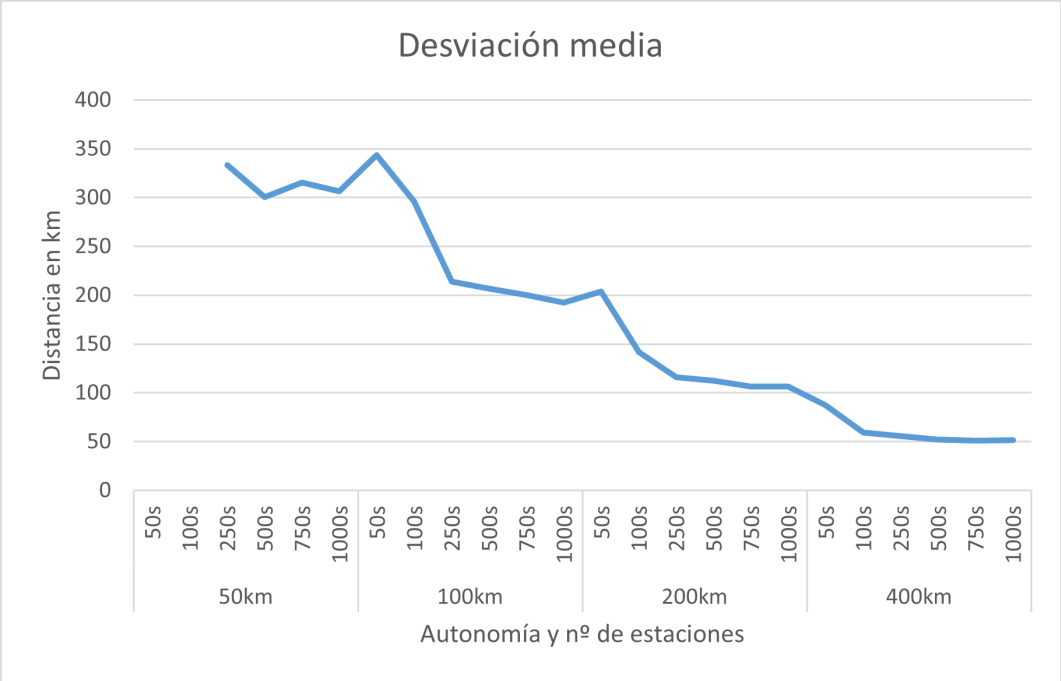


Figura 5.6: Desviación media respecto a la ruta óptima, distribución radial

Comenzando con las simulaciones relativas a los vehículos con la autonomía mas baja, debemos obviar en este punto los valores de los experimentos con 50 y 100 estaciones, pues como hemos visto en el apartado anterior, ningún trayecto ha sido completado. Esta es la única distribución que proporciona un dato válido con 250 estaciones, aunque solo se proporciona por parte del 3.2 % de los transportes. En general, obtenemos desviaciones que oscilan entre los 300 y los 333 kilómetros.

Para el siguiente escalón de autonomía, con 50 estaciones sumamos 344 kilómetros de media a nuestro viaje. La desviación desciende rápidamente hasta los 214 kilómetros

si añadimos 200 estaciones más. Con más estaciones no tenemos los mismos rendimientos marginales, es decir, la desviación no mejora de manera tan proporcional. Solamente conseguiremos reducirla hasta los 192 kilómetros.

El primer experimento de los vehículos de 200 kilómetros de autonomía arroja un resultado similar a los que comentábamos en el párrafo anterior. Como viene ocurriendo con la mayoría de saltos de autonomía, la cifra del desvío medio se ve decrementada muy rápidamente con los experimentos inmediatamente posteriores. En este caso, el dato más equilibrado se encuentra nuevamente ubicando 250 estaciones, dejando el desvío medio total en 116 kilómetros. Y esta cifra no cambia significativamente añadiendo más estaciones.

En último lugar tenemos las simulaciones con los coches de mayor autonomía. Observamos a primera vista que la desviación media no va a disminuir tanto como con en experimentos anteriores. Más en detalle, con 50 estaciones añadiríamos 87 kilómetros a nuestro trayecto. Duplicando las estaciones, la desviación desciende hasta los 59 kilómetros. A partir de este punto, la cifra continúa descendiendo de manera asintótica hasta los 51 kilómetros, por lo que no resulta muy rentable considerar colocar más de 100 estaciones.

5.2.3. Conclusiones

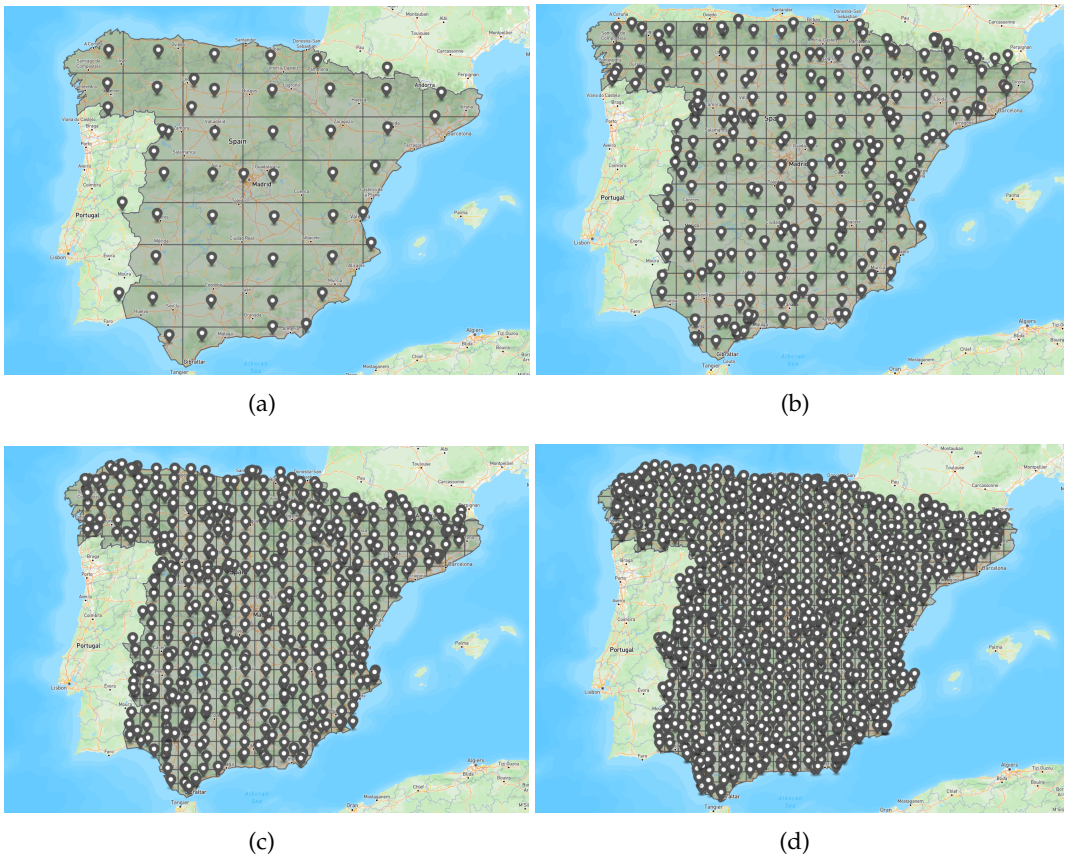
En resumen, hemos visto que la distribución radial de estaciones de recarga da cobertura de movimiento total a vehículos de alta capacidad autónoma. Los coches con menos de 200 kilómetros de autonomía sufrirían mucho para alcanzar sus destinos. Una minoría es capaz de hacerlo, pero a costa de aumentar muy considerablemente la distancia a recorrer.

En la otra cara de la moneda podemos destacar que, vistos los resultados de las simulaciones con coches de autonomía alta, la desviación media del camino óptimo se reduce muy rápidamente cuando hay pocas estaciones. Es decir, con pocas estaciones conseguimos un rendimiento proporcional mayor que si existen muchas. Por tanto, estamos ante una distribución más barata que las demás, dado que no hay que invertir una cantidad excesiva de recursos para alcanzar una calidad satisfactoria para los viajeros.

Sin embargo, cabe mencionar que, como se ha apuntado al principio de las conclusiones, y al igual que en la distribución aleatoria, esto solo sirve si contamos con un parque automovilístico eléctrico moderno y con altas prestaciones. En caso de decantarnos por una distribución de estas características, estaríamos dejando atrás a muchos conductores que no disponen de un vehículo de tan elevada gama ni recursos para hacerse con uno. Continuaremos explorando otras alternativas para ofrecer accesibilidad total a la mayor cantidad de viajeros posible.

5.3 Distribución uniforme

A continuación procederemos con el análisis de los resultados generados por las simulaciones realizadas sobre una distribución uniforme de estaciones de recarga (los resultados completos se pueden consultar en la tabla 5.4). En pocas palabras, esta distribución consiste en convertir el mapa español en una rejilla con tantas celdas como estaciones se deseen colocar. Evidentemente, se coloca una estación en cada celda. Así, obtenemos un mapa en el que todas las estaciones están repartidas de manera ordenada, a una distancia similar de sus adyacentes. Sirva la figura 5.7 como ejemplo ilustrativo.



(a) (b)

(c) (d)

Figura 5.7: Mapas de estaciones en la distribución uniforme

(a) 50 estaciones; (b) 250 estaciones; (c) 500 estaciones; (d) 1000 estaciones

Distribución uniforme		Trayectos abortados (en %)	Desviación media (en km)
50km	50s	100	-
	100s	100	-
	250s	100	-
	500s	14.8	404.39
	750s	6.2	324.34
	1000s	3	307.87
100km	50s	100	-
	100s	68.2	362.22
	250s	0	217.06
	500s	0	188.80
	750s	0	181.82
	1000s	0	172.98
200km	50s	0	167.12
	100s	0	141.95
	250s	0	118.62
	500s	0	104.60
	750s	0	99.32
	1000s	0	99.89
400km	50s	0	70.66
	100s	0	58.34
	250s	0	57.87
	500s	0	52.87
	750s	0	48.87
	1000s	0	46.92

Tabla 5.4: Tabla de resultados, distribución uniforme

Eje horizontal: autonomía de los transportes (km) y número de estaciones (s)

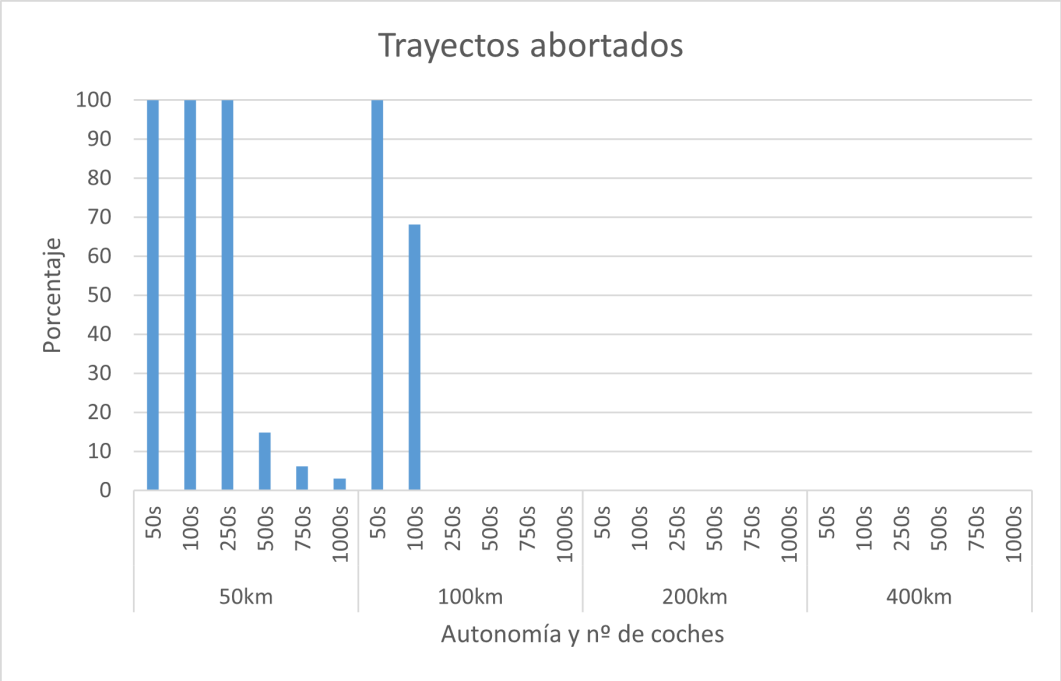


Figura 5.8: Porcentaje de transportes que abortan su trayecto, distribución uniforme

5.3.1. Eficacia de la distribución

Una vez más, extraigamos la información acerca de la eficacia de la distribución. En otras palabras, de la proporción de los trayectos que se pueden realizar de manera efectiva. Los datos se pueden visualizar en la figura 5.8

Centrándonos primeramente en los vehículos de baterías más reservadas, vemos que en las primeras simulaciones no se ha podido culminar ni un solo trayecto interurbano por España. En cambio, a partir del experimento con 500 estaciones, las simulaciones arrojan un nivel de eficacia muy alto para el poco kilometraje que tienen los coches que circulan, rondando un porcentaje de trayectos abortados del 15 %. Con 250 estaciones más, la eficacia aumenta, decrementando la cifra hasta el 6.2 %, o incluso al 3 % si ubicamos todas las estaciones posibles.

En cuanto al segundo escalón de autonomías, volvemos a ver que con el mínimo de estaciones todos los coches se quedan sin poder llegar a su destino. Con 100 estaciones, la eficacia mejora, pero sigue imposibilitando el viaje a la gran mayoría de viajeros de la muestra (concretamente, al 68.2 %). Sin embargo, con 250 estaciones o más apreciamos que el 100 % de los vehículos han podido desplazarse libremente hasta su destino final. Esto es un dato extremadamente bueno, pues con una cuarta parte de las estaciones que hemos presupuestado para este trabajo, todos los vehículos de al menos 100 kilómetros de autonomía pueden circular de manera eficaz por suelo español.

En relación con los vehículos con mayores autonomías que los anteriores, ninguno de ellos se ve obligado a abortar su viaje por falta de energía. Todos y cada uno de ellos son capaces de arribar a su destino planificado. En consecuencia, esta es la primera distribución que otorga accesibilidad total a esta gama de coches eléctricos con solo 50 estaciones localizadas en el mapa. Veremos a continuación con qué grado de dificultad circulan los mencionados vehículos.

5.3.2. Eficiencia de la distribución

En este subapartado vamos a analizar la distancia añadida al trayecto respecto a un viaje sin paradas como medida de eficiencia. Los datos que se van a tener en cuenta han sido dispuestos de forma gráfica en la figura 5.9.

Figura 5.9: Desviación media respecto a la ruta óptima, distribución uniforme

Empecemos indicando los resultados de las simulaciones con vehículos de mínima autonomía. Como hemos visto en el análisis de la eficacia de esta distribución, hasta la simulación con 500 vehículos no obtenemos un resultado válido, pues todos los vehículos se contabilizaron como "bloqueados". El primer dato significativo, el correspondiente a dicha simulación, arroja una desviación media total de 404 kilómetros. Teniendo en cuenta que los trayectos configurados tienen una longitud mínima (en línea recta) de 600 kilómetros, esto implicaría, en el peor de los casos, un aumento del 67,33 % de la distancia recorrida. Lograríamos reducir este kilometraje extra poniendo más estaciones, pero no hasta menos de 307 kilómetros (con 1000 estaciones).

De nuevo, para aquellas simulaciones realizadas con vehículos con una autonomía de 100 kilómetros, recogemos un valor no válido para la primera de ellas. Hemos enunciado con anterioridad que, para estos vehículos, con 50 estaciones es imposible circular entre ciudades en España. También habíamos visto que con 100 estaciones había una muestra bastante limitada de vehículos con viajes concluyentes. A la vista de los resultados, dichos vehículos tuvieron que desviarse 362 kilómetros de media para poder recargar baterías de camino al destino. Si sumamos estaciones hasta alcanzar las 250 estaciones, la desviación media cae rápidamente hasta los 217 kilómetros; y con 500 hasta los 189 kilómetros. Se podría discutir con cuál de estos valores se obtiene mayor rendimiento, pues, desde luego, añadiendo más estaciones no se aprecian diferencias notables en el resultados. Ubicando las 1000 estaciones en el lugar que les corresponde por la distribución uniforme, la desviación media se coloca en los 173 kilómetros.

Seguidamente, para los vehículos de 200 kilómetros de autonomía sí que poseemos un dato válido en la primera simulación, pues el efecto de aumentar la autonomía ya permite a todos los vehículos su libre circulación por suelo español. La curva comienza

en este caso con un valor de 167 kilómetros, que es ligeramente inferior al que acabamos de analizar en el párrafo inmediatamente anterior. La cifra desciende prácticamente de manera continua hasta los 105 kilómetros con 500 estaciones colocadas en su ubicación pertinente. A partir de este punto y hasta que se colocan las 1000 estaciones, la desviación varía de manera casi imperceptible hasta los 100 kilómetros.

Por último, en relación con los vehículos de baterías más potentes, ya con 50 estaciones obtenemos un desvío medio a de 70 kilómetros. Tener el doble de autonomía supone una mejora muy notable en esta ocasión, en comparación con todos los valores analizados en el párrafo previo. Destacamos en esta sección de la gráfica el resultado de la simulación con 100 estaciones, con 58 kilómetros de desvío; y el correspondiente a las 500 estaciones, con 53 kilómetros. Finalmente, la desviación media desciende ligeramente por debajo de los 50 kilómetros cuando colocamos 750 estaciones o más.

5.3.3. Conclusiones

En resumen, hemos visto que esta distribución es distinta a las demás en cuanto a su eficacia y eficiencia. Lo más llamativo es que funciona muy bien con los vehículos de mayor autonomía, pues absolutamente todos los vehículos de 200 kilómetros de autonomía podrían quedar habilitados para completar su trayecto y con las desviaciones más bajas hasta ahora. Los vehículos con autonomías más limitadas requieren de al menos 500 estaciones para poder comenzar a realizar desplazamientos entre ciudades.

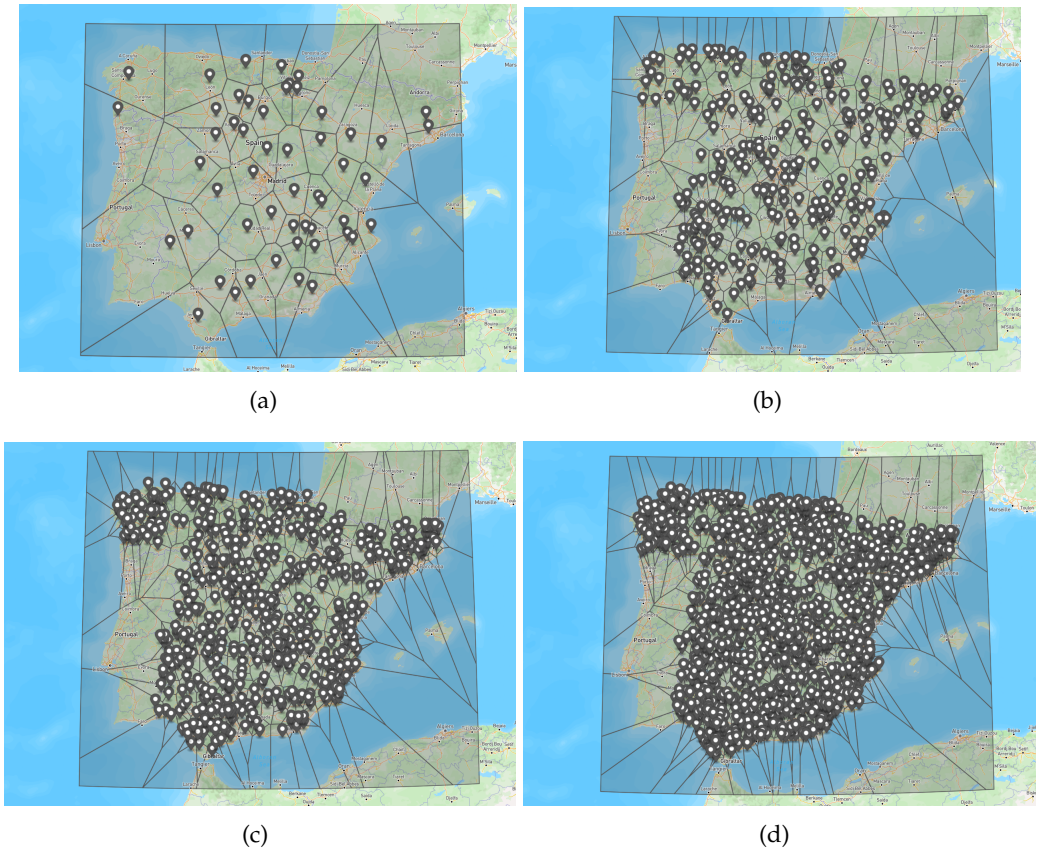
No obstante, con 500 estaciones obtenemos un buen rendimiento en varios aspectos: en relación a la eficacia, con tantas estaciones tenemos el primer experimento para el que los vehículos de 50 kilómetros de autonomía comienzan a completar sus trayectos, de los cuales solo un 18 % se quedan sin finalizar. Es un porcentaje bastante aceptable para vehículos que no están diseñados para realizar viajes interurbanos y que probablemente no se adquieran con esa intención. Todos los demás vehículos, sean de la gama que sean, se pueden mover de manera completamente libre por España. En relación a la eficiencia, en general, las curvas de los experimentos con las distintas autonomías ven ralentizado su avance a partir de las 500 estaciones. Estaríamos hablando de aquellas combinaciones de estaciones que generan un mayor rendimiento, por lo que es una distribución más cara que las demás si se quiere alcanzar el rendimiento óptimo.

Otro aspecto muy importante es el económico, pues debemos tener en cuenta que hay que equilibrar los gastos que vamos a acometer y el rendimiento efectivo que nos va a aportar la distribución por la que apostemos. Colocar las 500 estaciones puede resultar en un desembolso bastante exigente, pero desde mi punto de vista obtenemos un beneficio para los conductores muy elevado. Por tanto, dicha inversión estaría justificada.

A modo de conclusión final, nos encontramos ante una distribución muy beneficiosa pero cara. Con un poco de esfuerzo económico podríamos acercarnos a la tan ansiada accesibilidad universal, con desviaciones bajas. Salvo unos pocos vehículos de 50 kilómetros de autonomía, todos serían capaces de finalizar sus trayectos.

5.4 Distribución basada en el algoritmo genético

Avanzando en nuestro análisis, es el turno de la distribución basada en el algoritmo genético de generación de estaciones en sus ubicaciones óptimas, que obtiene sus conclusiones a partir de información demográfica real de la sociedad española (para más detalles, ver capítulo 2). En la figura 5.10 podemos ver cómo se ha plasmado el "output" del algoritmo en SimFleet y en la tabla 5.5 los resultados experimentales.



(a) (b)

(c) (d)

Figura 5.10: Mapas de estaciones en la distribución basada en el A.G.

(a) 50 estaciones; (b) 250 estaciones; (c) 500 estaciones; (d) 1000 estaciones

Distribución basada en el A.G.		Trayectos abortados (en %)	Desviación media (en km)
50km	50s	100	-
	100s	100	-
	250s	100	-
	500s	80.6	221.90
	750s	33.2	219.27
	1000s	19	201.01
100km	50s	99.8	265.46
	100s	93.6	214.89
	250s	1.2	190.03
	500s	0	140.71
	750s	3.4	120.43
	1000s	1.4	116.19
200km	50s	12.4	137.57
	100s	0	105.93
	250s	0	94.71
	500s	0	79.27
	750s	0	71.76
	1000s	0	69.07
400km	50s	0	60.63
	100s	0	51.14
	250s	0	44.80
	500s	0	42.42
	750s	0	37.93
	1000s	0	38.16

Tabla 5.5: Tabla de resultados, distribución basada en el A.G.

Eje horizontal: autonomía de los transportes (km) y número de estaciones (s)

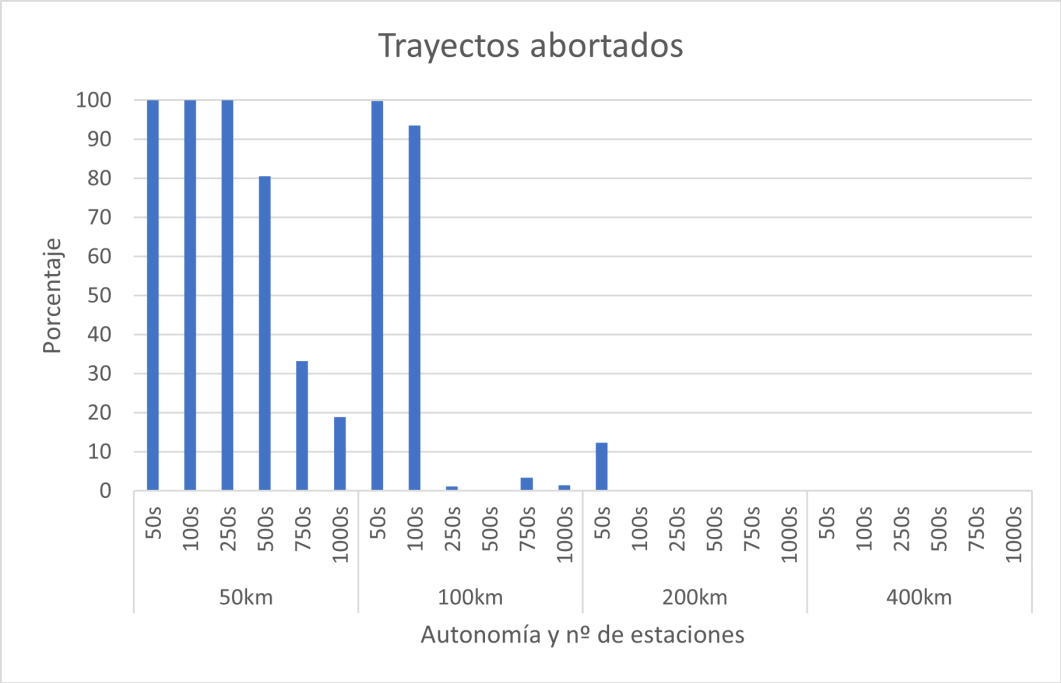


Figura 5.11: Porcentaje de transportes que abortan su trayecto, distribución basada en el A.G.

5.4.1. Eficacia de la distribución

Para empezar, comencemos examinando lo relativo a la eficacia de la distribución. La figura 5.11 contiene todos los datos relevantes.

Salta inmediatamente a la vista el alto nivel de trayectos abortados en el primer cuarto de la gráfica. Con 250 estaciones o menos es imposible realizar cualquier trayecto de la muestra. Con dicha cantidad, únicamente el 19 % logra alcanzar su meta. Si añadimos más estaciones, la eficacia de la distribución aumenta, dejando aproximadamente una quinta parte de los vehículos sin poder desplazarse.

Con vehículos de 100 kilómetros de autonomía, las dos primeras simulaciones arrojan un resultado muy desfavorable, con un 100 % y un 94 % de viajes cancelados respectivamente. Esta vez alcanzamos un nivel de eficacia aceptable a partir de las 250 estaciones. Ahí el nivel de coches inhabilitados desciende en picado hasta el 1 %. Nótese que, después del experimento con 500 estaciones, vuelve a crecer (de manera poco significativa) el porcentaje de trayectos abortados. Esto se explica por la saturación de la red, es decir, con más de 500 estaciones, como mucho, se puede mantener el mismo nivel de eficacia. A priori debería de haberse mantenido la eficacia al máximo, pero teniendo en cuenta que el criterio de ubicación de las estaciones (el resultado del algoritmo genético) no es aditivo, esto puede ocurrir. Es decir, las estaciones no siguen un modelo matemático tan simple como en la distribución homogénea o radial, ni se van "sumando" a lo que ya existe. Por tanto, un aumento de estaciones puede cambiar completamente la ubicación de las previamente existentes.

A partir de este punto, exceptuando un experimento, todos los vehículos logran hallar el camino a su destino. La única excepción ocurre en la simulación correspondiente a la distribución con 750 estaciones y 50 kilómetros de autonomía, donde aún hay un 12 % de transportes que cancelan su viaje.

5.4.2. Eficiencia de la distribución

En esta sección valoraremos la eficiencia de este modelo de posicionamiento de estaciones a partir de la distancia adicional que recorren los viajeros por tener que acercarse a una estación de recarga. Tomaremos para ello los datos que se hallan en la figura 5.12.

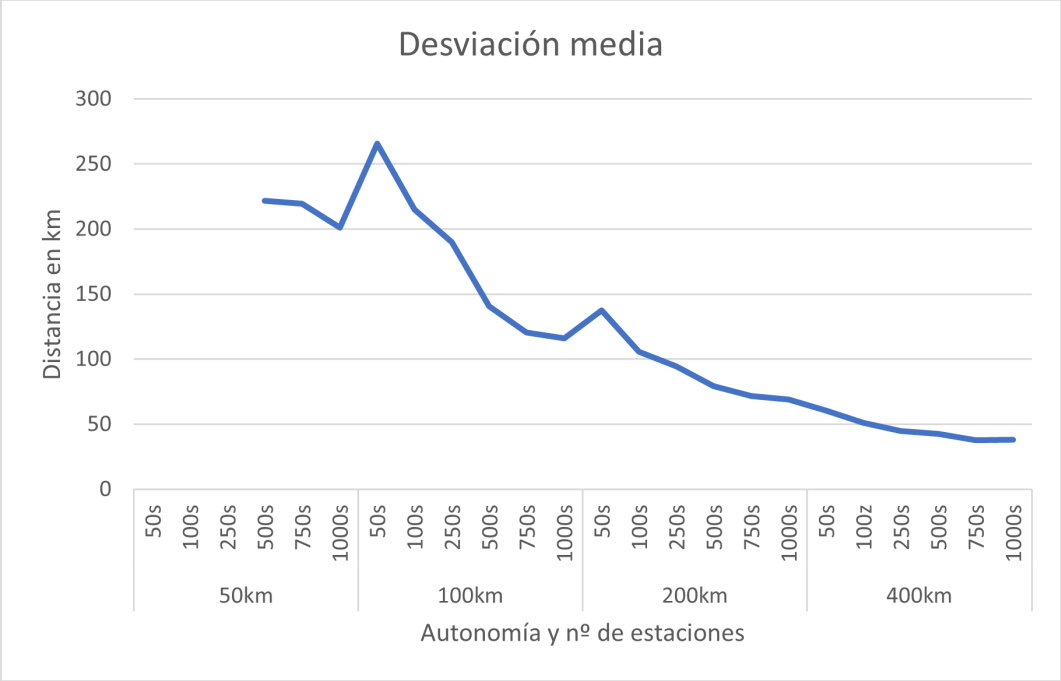


Figura 5.12: Desviación media respecto a la ruta óptima, distribución basada en el A.G.

Empezaremos por los vehículos con menor capacidad de carga. Como hemos visto en el apartado anterior, los primeros tres experimentos no arrojan ninguna desviación, pues ningún vehículo ha llegado a destino. Los restantes, que son los que tienen más estaciones en el mapa, tienen todos un resultado que supera los 200 kilómetros. Colocar más estaciones no ha tenido prácticamente ningún efecto sobre la eficacia.

Siguiendo con los transportes con 100 kilómetros de autonomía, el primer resultado puede resultar engañoso, pues solo hay un único coche que nos proporciona su desviación, que asciende a 265 kilómetros. La curva desciende de manera lineal si continuamos añadiendo estaciones hasta las 750, donde la desviación media ya solo es de 120 kilómetros. Esto es menos de la mitad del valor con el que comenzamos la serie, una mejora más que notable. Respecto al último experimento, vemos que es posible reducir la cifra hasta los 116 kilómetros.

En relación con los dos tipos de vehículos con mejores baterías, ambos guardan mucha similitud en cuanto a la progresión de sus resultados en función del número de estaciones. Los vehículos de 200 kilómetros de autonomía se desvían 138 kilómetros de su ruta ideal si solo existen 50 estaciones a lo largo de la red de carreteras españolas. Conforme incrementamos la cantidad de estaciones hasta 750, la desviación media se va decrementando hasta los 72 kilómetros. Por otra parte, los coches con 400 kilómetros de autonomía tienen que desplazarse 61 kilómetros con 50 estaciones. Si fueran 750 estaciones, este desplazamiento sería algo menor de 40 kilómetros. Por tanto, en ambos casos hemos alcanzado el rendimiento óptimo con 750 estaciones. En el último valor de la serie apreciamos un empeoramiento de la eficiencia muy ligero. Nuevamente, podría deberse a los motivos expuestos en la sección anterior acerca de la saturación de la red de estaciones.

Distribución Tesla	Trayectos abortados (en %)	Desviación media (en km)
50km	100	-
100km	100	-
200km	2.2	97.18841534
400km	0	37.5655462

Tabla 5.6: Tabla de resultados, distribución TESLA

Eje horizontal: autonomía de los transportes (km) y número de estaciones (s)

5.4.3. Conclusiones

Dado el análisis realizado, podemos extraer la conclusión de que la distribución basada en el Algoritmo Genético requiere mucha inversión para ser efectiva. En general, comparado con los demás valores de la serie, con pocas estaciones obtenemos una eficacia y eficiencia bastante limitada, aunque depende, evidentemente, de la autonomía que tengan los vehículos. Los experimentos que han demostrado ser mejores son aquellos que contienen 750 estaciones en el mapa, pues presentan las desviaciones medias más bajas y niveles muy bajos de vehículos en situación de "bloqueo". Si bien es cierto que tantas estaciones suponen un alto desembolso de recursos, puede valer la pena.

En cualquier caso, las curvas de eficiencia de esta distribución son mucho más lineales que en otras distribuciones. En otras palabras, se tarda más en alcanzar el punto de inflexión, por lo que podemos regular y equilibrar la inversión en estaciones y el nivel de eficacia/eficiencia al que queremos aspirar. En este sentido, este método para ubicar estaciones es mucho más flexible que otros.

Como aspecto negativo a destacar, el impacto a nivel de eficacia en los vehículos de menor autonomía es bajo. Son bastantes los que tienen que abortar el trayecto en algún punto, y los restantes tienen que aumentar la distancia del mismo en más de 200 kilómetros. Cabe destacar que, en ocasiones, el algoritmo genético prioriza la colocación de más terminales en una misma estación para compensar la alta de demanda de recarga en su zona. Esta es la razón por la que el número de estaciones puede diferir en muy pocas unidades en los experimentos, provocando que empeoren métricas como el porcentaje de viajes cancelados. Si incluyéramos el tiempo de espera en los puntos de recarga, veríamos que, en comparación con las demás distribuciones, donde solo existe un terminal por estación de recarga; es notablemente menor en la distribución basada en el algoritmo genético.

5.5 Distribución Tesla

La última distribución de estaciones de recarga que queda por analizar es la distribución existente de estaciones Tesla en España. Para poder realizar estos experimentos, recordemos, hemos extraído de la página web de Tesla las ubicaciones españolas donde se encuentran las estaciones de recarga propias de Tesla[1]. En el momento de configuración de los experimentos, existían 42 puntos de recarga en España (ver figura 5.13). Como el número de estaciones es constante para esta disposición, solamente podemos realizar el análisis de la movilidad en función de la autonomía de los vehículos. Los datos se encuentran en la tabla 5.6.

[1] www.tesla.com/es_es/findus/list/superchargers/Spain

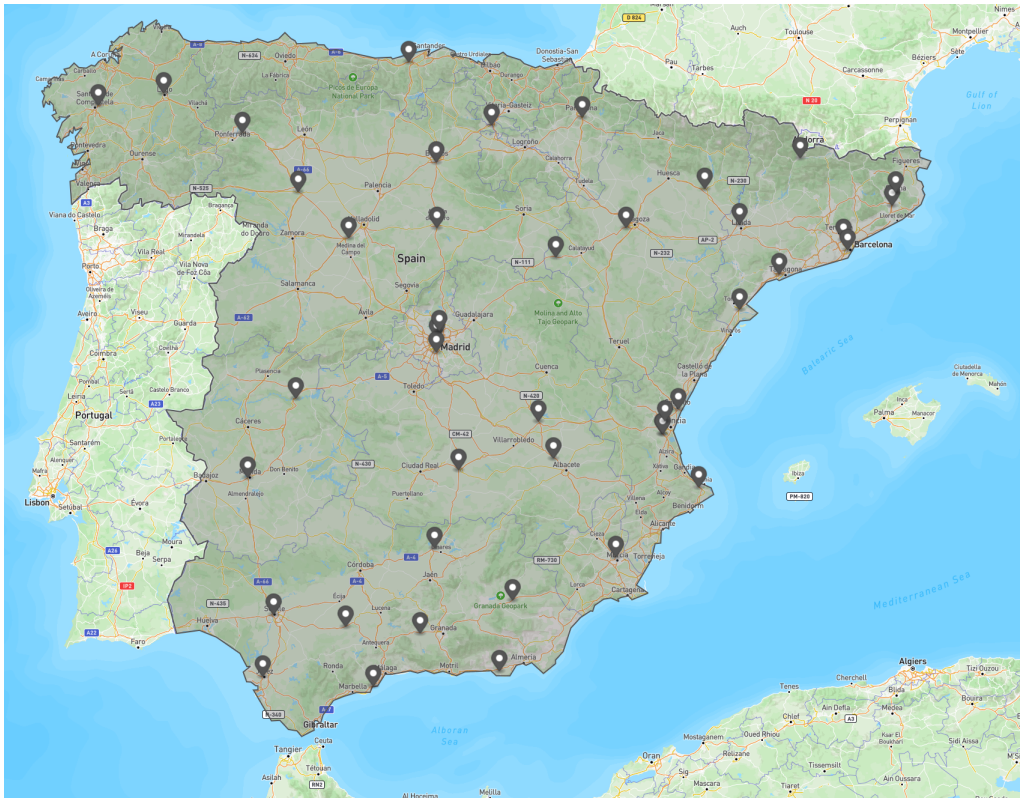


Figura 5.13: Mapa de estaciones en la distribución Tesla

5.5.1. Eficacia de la distribución

Como venimos realizando, comenzaremos analizando el grado de eficacia de la distribución. A primera vista podemos concluir que no es posible utilizar la red de estaciones Tesla para realizar desplazamientos prolongados entre municipios si no disponemos de un coche con una autonomía mayor de 100 kilómetros. Los dos primeros experimentos, cuyos coches tienen una autonomía más limitada, arrojan un resultado de 100 % de coches que tienen que abortar su viaje. El tercer experimento, relacionado con los transportes con 200 kilómetros de autonomía, sí que da cobertura a prácticamente todo el territorio español. Solamente el 2.2 % de ellos no pudieron alcanzar su destino, lo que equivale a 11 transportes de 500. Por último, con las baterías más potentes, los viajeros ya sí que pueden moverse de manera totalmente libre a donde deseen.

5.5.2. Eficiencia de la distribución

En cuanto a la desviación media total, que es la medida de la eficiencia de las distribuciones, únicamente podemos poner el foco en las simulaciones correspondientes a los vehículos de más alta gama. Esto es debido a que, como se ha enunciado en el párrafo inmediatamente anterior, no podemos tomar las desviaciones como válidas para el cálculo de la media porque ningún transporte ha culminado su trayecto. Llama la atención el gran resultado de las dos simulaciones restantes (ver figura 5.15), con desviaciones medias menores que 100 kilómetros en los dos casos. Los coches con autonomía de 200 kilómetros tienen que añadir 97 kilómetros a su viaje para poder hacer uso de las estaciones de Tesla. Mientras tanto, los que tienen la autonomía más alta solamente suman menos de la mitad, unos 38 kilómetros.

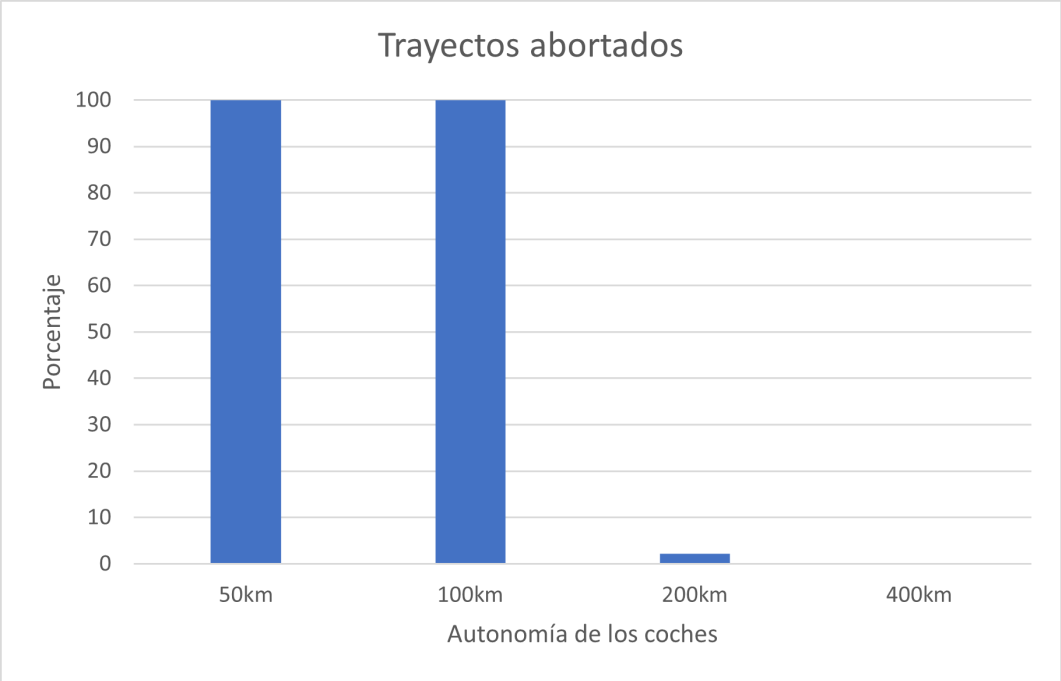


Figura 5.14: Porcentaje de transportes que abortan su trayecto, distribución Tesla

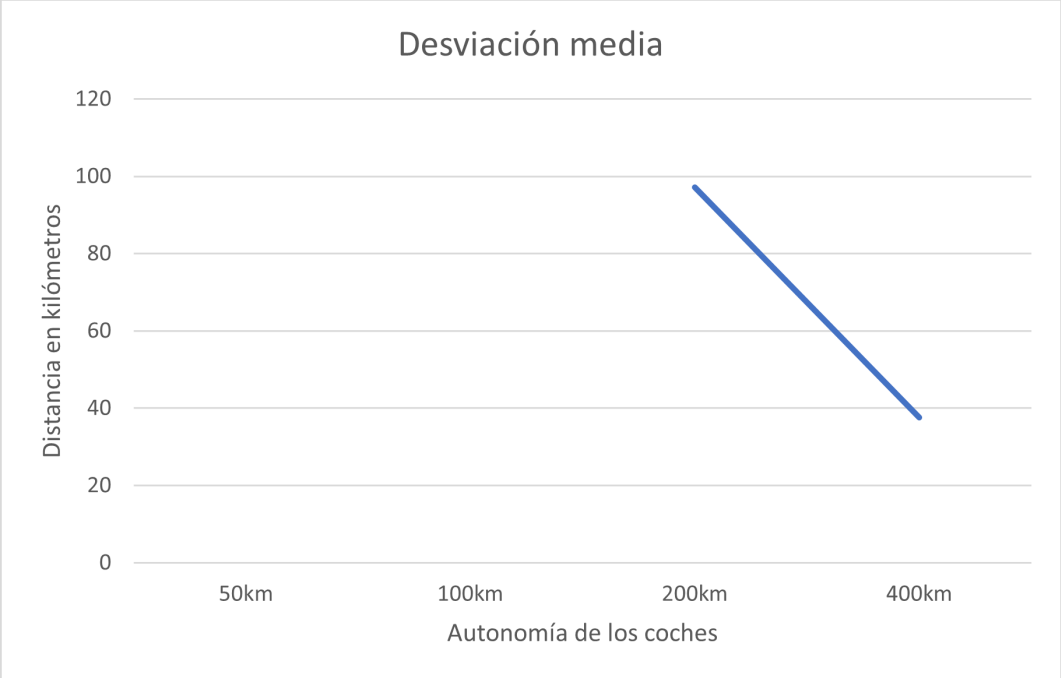


Figura 5.15: Desviación media respecto a la ruta óptima, distribución Tesla

5.5.3. Conclusiones

En resumen, la distribución de Tesla en España muestra un buen nivel de eficacia y de eficiencia cuando los vehículos que circulan por carretera tienen las autonomías más altas. Hay que destacar, antes de nada, que los experimentos que hemos analizado e interpretado, solamente contienen 42 estaciones. Por tanto, para realizar un análisis comparativo, tenemos que utilizar los resultados de esta distribución con los resultados de otras distribuciones con 50 estaciones (lo más cercano a 42). Al igual que en la estrategia de emplazamiento uniforme, esta distribución no tiene ningún resultado válido para los dos primeros experimentos con 50 estaciones. En el caso de Tesla, esto simboliza la nula cobertura de estaciones para los vehículos con los dos tipos de baterías más modestas. Por lo demás, en cuanto a las autonomías más altas, esta distribución ha arrojado un resultado mejor que la basada en el algoritmo genético. Si se puede realizar un viaje (hecho que ocurre en la gran mayoría de ocasiones), el desvío total oscilaría entre los 97 y los 38 kilómetros. Esto, para un coche eléctrico, que actualmente no tiene muchas alternativas para repostar, es una cifra extremadamente baja.

Es cierto que estos resultados son muy positivos para Tesla y sus usuarios, pero solamente lo es para ellos. Los coches fabricados por el gigante de la automoción americano son considerados como aquellos más durables del mercado, es decir, sus baterías están expresamente diseñadas para recorrer muchos kilómetros sin detenerse. Por tanto, a la hora de desarrollar su mapa de estaciones particular, la compañía apuntaba claramente a favorecer a sus clientes, diseñando una disposición ideal para sus coches de altas prestaciones. Evidentemente, los coches de menor categoría quedaron fuera de sus planes.

Por tanto, como conclusión final, la distribución de Tesla es ideal para sus propósitos: permitir viajar con total libertad a sus propios usuarios directos, no teniendo en cuenta a los vehículos que no son de su marca, en general, con baterías de menor capacidad. La distribución cumple su función a la perfección, pero desde una perspectiva global, no otorgaría accesibilidad a otros vehículos menores, dejando las posibilidades de los viajeros españoles en un gran desequilibrio.

5.6 Conclusiones globales de los experimentos

Como habíamos planificado, hasta este punto hemos dedicado un apartado a cada tipo de distribución, extrayendo los datos de las simulaciones realizadas, analizándolos e interpretando puntos fuertes y puntos más débiles. En esta sección vamos a realizar un análisis comparativo de los resultados, y para ello nos apoyaremos en la gráfica 5.16.

En la figura visualizamos todas las curvas de eficiencia de las cinco distribuciones analizadas. La distribución con peores resultados es la aleatoria, que en la mayoría de los casos presenta desviaciones medias totales más altas. Sin embargo, con cantidades altas de estaciones colocadas, es capaz de superar a la distribución radial. En total, este mal resultado nos confirma que colocar estaciones de recarga sin sentido alguno no es una buena opción, salvo que se disponga de un alto nivel de recursos económicos para construir estaciones.

La distribución radial tampoco ha dado un buen resultado, pues la mayoría de las demás estrategias de emplazamiento de estaciones tienen resultados más positivos. No obstante, tiene mejores números que la distribución aleatoria en ocasiones. Al contrario que lo que se comentaba antes, este tipo de red de estaciones resulta más favorable con pocas estaciones que con muchas. Se aprecia a primera vista viendo que es la única distribución para la que consiguen circular vehículos de 50 kilómetros de autonomía habiendo 250 estaciones en el mapa.

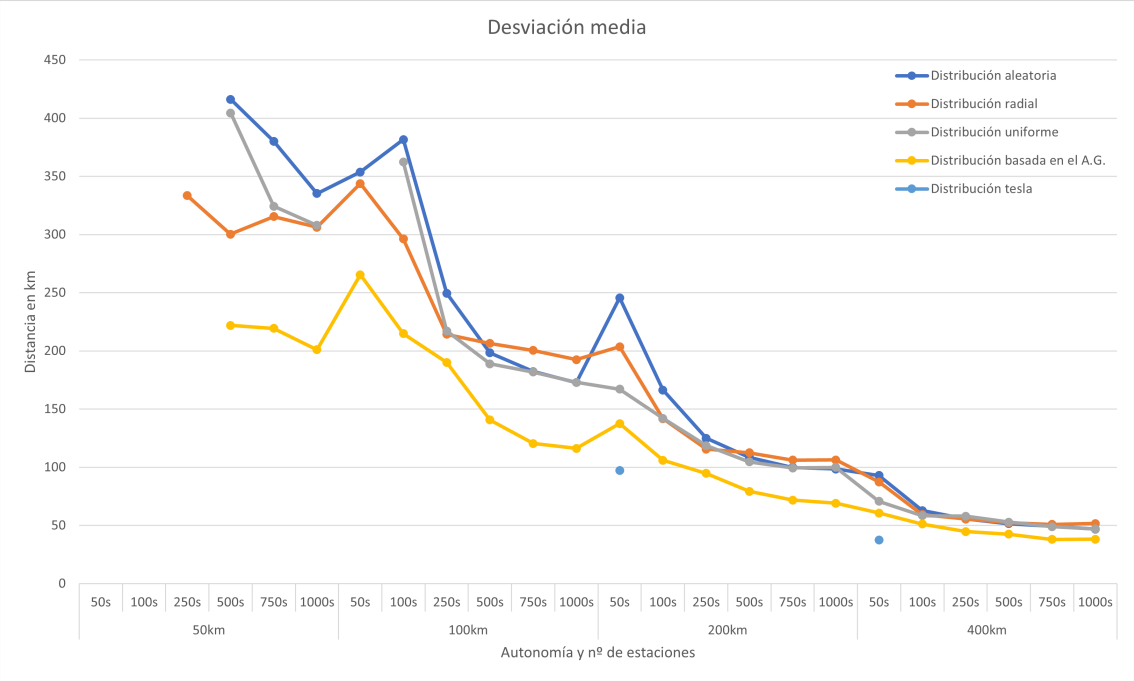


Figura 5.16: Desviación media respecto a la ruta óptima, agregado

La siguiente estrategia de localización de estaciones es la uniforme, de la que destacamos su elevado margen de mejora cuando hay pocas estaciones. En comparación con otras curvas, la de la distribución uniforme suele tener pendientes más elevadas en los primeros experimentos de cada gama de vehículos. Tal como hemos analizado en la sección anterior, esto es porque la distribución es barata, pues con pocas estaciones podemos superar rápidamente a las demás redes.

En el caso de la distribución basada en el algoritmo genético de Jordán et al. [6], esta es la distribución que ha dado los mejores resultados, pues está por debajo de las que acabamos de analizar. Su curva sigue la misma tendencia que las demás, pero siempre varios kilómetros por debajo. Conforme aumenta la autonomía de los vehículos, estas diferencias se hacen más pequeñas, pero eso se debe a la saturación de los experimentos al no poder mejorarse el resultado.

Por último, debemos fijarnos en la red de estaciones de Tesla. Los dos puntos de los que disponemos en la gráfica representan mejores resultados que los correspondientes al algoritmo genético. Esto demuestra el gran trabajo de la compañía americana a la hora de diseñar los emplazamientos de sus supercargadores, porque con pocos puntos de recarga presentan desviaciones muy bajas. El motivo, como se puede ver en los mapas de la distribución Tesla, es que los terminales se encuentran estratégicamente ubicados cerca de carreteras principales. Esto evita a los viajeros tener que cambiar su ruta para ir a recargar baterías. Sin embargo, como ya se ha discutido, con tan pocas estaciones, de momento, todo esto solo sirve para los vehículos de baterías de mayor capacidad, como sus propios modelos de coche. Por tanto, Tesla debe seguir incrementando el número de ubicaciones donde colocar estaciones de recarga.

El punto débil más fuerte de la red de Tesla, a la vista de los resultados experimentales, es que no está tan expandida como para cubrir todo el territorio español. En consecuencia, los coches eléctricos con menos capacidad no pueden utilizarla. Y es que Tesla tiene 42 estaciones de recarga por toda la red de carreteras, con 306 terminales en total[2]. Eso equivale a unos 7 terminales por estación, en promedio. Para reducir la incidencia de

[2] www.tesla.com/findus/list/superchargers/Spain

este aspecto en la eficacia de la distribución, sería conveniente que estudie un cambio en su política de reparto de terminales en futuras construcciones, para tener más puntos de recarga en el mapa con menos estaciones. Eso ayudaría a mejorar la accesibilidad universal de las estaciones en detrimento del tiempo medio de espera para recargar, que de por sí no es elevado (unos 75 segundos, según los resultados de SimFleet).

En conclusión, consideramos que la red de estaciones de recarga basada en el algoritmo genético es la más adecuada para diseñar una nueva estrategia de colocación de estaciones. Si Tesla complementara su distribución de estaciones observando los resultados del algoritmo genético, podría diseñarse una red que combine accesibilidad para vehículos eléctricos de menor gama con eficiencia a la hora de realizar un viaje. Es decir, tendríamos una estrategia de emplazamiento de estaciones en la que se pone por delante la cercanía de las estaciones a la autovía, pero teniendo en cuenta las demás necesidades de la población, representada en la información demográfica de entrada del algoritmo genético.

CAPÍTULO 6
Conclusiones

Hecho un análisis de los datos provenientes de los experimentos realizados sobre SimFleet e interpretada la progresión de las curvas de eficacia y eficiencia, damos por finalizado el estudio y podemos extraer las conclusiones de este trabajo.

En el presente Trabajo de Fin de Grado hemos barajado varias alternativas como simuladores de flotas de vehículos (Eclipse SUMO, MATSim, AnyLogic). Todos ellos eran candidatos a formar parte del desarrollo del proyecto, sin embargo, SimFleet se ha erigido como el más adecuado para el desarrollo de los experimentos por la familiaridad de los tutores del proyecto, la facilidad para moldearlo y adaptarlo a las necesidades individuales de cada usuario, el hecho de que su uso sea gratuito y su escalabilidad de cara al procesamiento de simulaciones a nivel nacional.

Después de una fase de diseño, donde se ha analizado al detalle el funcionamiento previo de SimFleet, de sus agentes y de los cambios de comportamiento que queríamos aplicar a los mismos, se han llevado a la práctica dichas ideas. Los cambios han afectado en gran medida a la figura principal de los experimentos, el agente que simboliza el vehículo eléctrico. Se ha eliminado el agente relativo al cliente y con ello todo el protocolo de interacción con el transporte. Para reflejar la realidad de los viajes largos entre ciudades, los que queremos poner a prueba, se han introducido nuevos algoritmos de búsqueda de estaciones, condiciones de terminación de los experimentos y nuevos estados en la máquina de estados, así como atributos de clase con los que calcular las estadísticas que necesitamos.

Ya con el simulador funcionando según los requisitos impuestos en la fase de diseño, se ha procedido a lanzar secuencialmente los más de cien experimentos en el simulador y a procesar los resultados de manera masiva, calculando en cada caso las métricas individuales acerca de los niveles de eficiencia y la eficacia que han mostrado las distribuciones de estaciones de recarga.

Respecto a las distribuciones creadas de manera artificial por los generadores de carga, algunas han resultado ser más idóneas que otras en ciertas circunstancias. La distribución aleatoria es la que tiene los peores números y es óptima cuando se colocan muchas estaciones (es una estrategia cara). En contrapartida, la siguiente distribución, la radial, se potencia con un número reducido de estaciones (es una estrategia barata). La red uniforme de estaciones se encuentra en un término medio, pues mejora sus cifras muy rápido cuando hay pocas estaciones, pero no tanto cuando hay muchas (rendimientos marginales decrecientes).

En cuanto a la distribución basada en los resultados del algoritmo genético desarrollado por Jordán et al.[6], esta ha demostrado ser el mejor y más flexible método de emplazamiento de estaciones, pues ocasiona las desviaciones medias totales más bajas en

todo momento. En general, la curva de eficiencia de la distribución alcanza su mejor rendimiento con 500 o 750 estaciones.

Por último, en relación con los experimentos realizados sobre la red de estaciones propietarias de Tesla en España, los datos que ofrece esta distribución son incluso mejores que los del algoritmo genético, y demuestran que la red está diseñada con buen criterio. La red de supercargadores Tesla proporciona un buen resultado porque sus terminales de carga se encuentran ubicados de manera adyacente a las carreteras principales del país, ocasionando un desvío más bajo a los conductores. Sin embargo, está centrada en dar servicio a los usuarios de coches con baterías más potentes, sus clientes. Por tanto, no es tan equilibrada como las redes generadas por el algoritmo genético. Como medidas para mejorar el rendimiento de esta red, debería de enfocarse más en dar servicio a coches eléctricos colocando más puntos geográficos en el mapa para acudir a recargar baterías en vez de acumular muchos terminales en la misma localización. El algoritmo genético podría ser un buen criterio para seleccionar la ubicación de nuevas estaciones, pues utiliza información demográfica de los españoles. De este modo puede maximizarse la satisfacción de los clientes de la red de estaciones.

6.1 Relación del trabajo desarrollado en relación con los estudios cursados

Como estudiante que ha cursado todas las asignaturas del Doble Grado en Ingeniería Informática y Administración de Empresas, son muchas las áreas de la informática que se han estudiado a lo largo de los años de carrera. Para la realización de este trabajo, sobre todo, han sido de mucha ayuda varias asignaturas de la rama de Ingeniería del Software y de Computación.

Los conocimientos más fundamentales han sido aquellos relacionados con la gestión de proyectos informáticos. Durante el presente estudio hemos mencionado en varias ocasiones que estábamos siguiendo el flujo de un proyecto informático tradicional, con sus cuatro fases: planificación, diseño, desarrollo y control. Estructurar de esta manera el avance del trabajo ha sido muy importante, y ello conlleva un profundo conocimiento de qué se realiza en cada una de ellas. Esto ha sido estudiado en la asignatura de “Ingeniería del Software”.

Las fases más determinantes para este proyecto han sido las de diseño y desarrollo. En esta parte he empleado la mayoría de conocimientos adquiridos durante los estudios, concretamente de la asignatura “Sistemas Inteligentes”. Durante el tiempo que estuve desarrollando las adaptaciones del programa, además de las explicaciones concretas y el material proporcionado por los tutores, saber qué es un agente inteligente y cómo funciona su máquina de estados, los callbacks de eventos (“Tecnologías de Sistemas en Red”), el funcionamiento del paradigma de la comunicación asíncrona (“Concurrencia y Sistemas Distribuidos”) y las estructuras complejas de datos (como los diccionarios, JSON, tablas Excel) y su manipulación, era de vital importancia. Esto último se explicó en “Estructuras de Datos y Algoritmos”.

Por otra parte, las simulaciones requerían de un sistema operativo UNIX, pues su estructura y configuración permite la apertura de muchos más documentos donde albergar la información de cada agente. Saber cómo manejar un software de este tipo es crucial, sobre todo si solo se puede acceder desde la línea de comandos haciendo uso de protocolos de conexión remota como SSH (para lanzar las simulaciones mediante un comando de ejecución) o FTP (para el envío de configuraciones y resultados de un ordenador a otro). En “Fundamentos de Sistemas Operativos” y “Gestión de Servicios Informáticos y

Tecnologías de la Información" aprendimos a utilizar el sistema UNIX, y en "Redes de Computadores" los protocolos mencionados.

Finalmente, en la fase de control o implantación del proyecto desarrollado, he mostrado mis habilidades en el análisis estadístico de una gran cantidad de datos para resumirlos de manera clara y visual. Son varias las asignaturas donde se impartieron estos conocimientos, especialmente "Introducción a la Estadística". En este punto también fueron importantes mis competencias adquiridas en el terreno del pensamiento crítico a la hora de interpretar las estadísticas resultantes.

Por último, la actualidad del tema de fondo del estudio, el del crecimiento del mercado de coches eléctricos en España y la falta de una red de estaciones de recarga en condiciones; y el análisis realizado sobre él, demuestran el conocimiento de problemas contemporáneos y su aplicación a un proyecto de investigación.

6.2 Trabajos futuros

Hay varios aspectos que, por la ya gran envergadura del presente trabajo, han quedado fuera de su ámbito. Como trabajo futuro, podría aumentarse la precisión de las simulaciones empleando un set informado de vehículos. Al igual que en el desarrollo del algoritmo genético de Jordán et al.[6], el cual emplea datos demográficos reales de la población residente, podría plantearse la búsqueda de una base de datos pública donde encontrar trayectos interurbanos que se realicen en un día cotidiano en España. El enfoque de los generadores de viajes aleatorios adaptados por el autor consistía en anteponer la cantidad a la calidad de los transportes y sus desplazamientos. Con esto, se estudia si se dota de cobertura a todo el territorio español, pero posiblemente existan zonas que no necesiten dicha cobertura, o solo en parte. Por ello, acercarse más a la realidad del tráfico en nuestro país seguro que proporciona información más precisa acerca de las necesidades de las personas.

Por los mismos motivos (estudiar la cobertura de las redes de estaciones), se han dejado de lado las métricas relativas al tiempo de los viajes. De continuar con estas investigaciones, debería hacerse un estudio adicional con los tiempos totales de viaje, el tiempo que se pasa haciendo cola en estaciones saturadas, etc. En relación a este último punto, la variación del número de terminales de carga por estación sería el parámetro a regular, teniendo en cuenta siempre el equilibrio entre bienestar ciudadano y recursos económicos disponibles para su instalación.

Por último, respecto a los generadores de carga, se podría incluir un paquete de nuevas modificaciones para mejorar la colocación de estaciones. Podría idearse un nuevo algoritmo que pondere los criterios de emplazamiento de estaciones para que sean localizadas lo más cerca posible de las autovías principales de la red de carreteras. Así, aprovecharíamos nuestros descubrimientos acerca de las fortalezas de la red de supercargadores Tesla reduciendo el nivel general de desviación total de los viajeros y aumentando, a su vez, su comodidad y satisfacción.

Bibliografía

[1] Asociación Nacional de Fabricantes de Automóviles y Camiones. Informe anual 2020, julio 2021. Obtenido de https://anfac.com/wp-content/uploads/2021/07/Informe-Anual-ANFAC-2020.pdf.

[2] Ángel Arcos-Vargas. El coche eléctrico: fortalezas y debilidades para su expansión. *Papeles de Economía Española*, tomo Infraestructuras terrestres, transporte y movilidad de personas(171):págs. 63–73, marzo 2022.

[3] Roberto Scholtes Ruiz. Impacto del vehículo eléctrico en la industria española: disrupción económica en ciernes. *Economía industrial*, tomo 411:pág. 113, 2019.

[4] Zachary A. Needell, James McNerney, Michael T. Chang y Jessika E. Trancik. Potential for widespread electrification of personal vehicle travel in the united states. *Nature Energy*, tomo 1, agosto 2016.

[5] Jaume Jordán, Pasqual Marti Gimeno, Javier Palanca, Vicente Julián y Vicent Botti. Interurban electric vehicle charging stations through genetic algorithms. En *Hybrid Artificial Intelligent Systems*, págs. 101–112. Springer International Publishing, septiembre 2021.

[6] Jaume Jordán, Javier Palanca, Elena del Val, Vicente Julian y Vicent Botti. Localization of charging stations for electric vehicles using genetic algorithms. *Neurocomputing*, tomo 452:págs. 416–423, 2021.

[7] José Ignacio García-Valdecasas Medina. La simulación basada en agentes: una nueva forma de explorar los fenómenos sociales. *Revista Española de Investigaciones Sociológicas (REIS)*, tomo 136(1):págs. 91–109, 2011.

[8] Nigel Gilbert. Agent-based models. *The Centre for Research in Social Simulation*, 01 2007. doi:10.1007/978-0-387-35973-1_39.

[9] Rafael C Cardoso y Angelo Ferrando. A review of agent-based programming for multi-agent systems. *Computers*, tomo 10(2):pág. 16, 2021.

[10] Peter Saint-Andre. Extensible messaging and presence protocol (xmpp): Core. Informe técnico, Internet Engineering Task Force, 2011.

[11] Sven Bendel, Thomas Springer, Daniel Schuster, Alexander Schill, Ralf Ackermann y Michael Ameling. A service infrastructure for the internet of things based on xmpp. En *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, págs. 385–388. 2013. doi:10.1109/PerComW.2013.6529522.

[12] William A. V. Clark y Mark Fossett. Understanding the social context of the schelling segregation model. *Proceedings of the National Academy of Sciences*, tomo 105(11):págs.

4109–4114, 2008. doi:10.1073/pnas.0708155105. Obtenido de https://www.pnas.org/doi/abs/10.1073/pnas.0708155105.

[13] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner y Evamarie Wiessner. Microscopic traffic simulation using sumo. En *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, págs. 2575–2582. 2018. doi:10.1109/ITSC.2018.8569938.

[14] Sumo at a glance. URL https://sumo.dlr.de/docs/SUMO_at_a_Glance.html.

[15] Jakob Erdmann, Robert Oertel y Peter Wagner. Vital: A simulation-based assessment of new traffic light controls. En *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, págs. 25–29. 2015. doi:10.1109/ITSC.2015.12.

[16] Fabio Gutiérrez Álvarez. *Uso de SUMO para la simulación de tráfico con usuarios vulnerables*. Trabajo de fin de carrera, Universidad Carlos III de Madrid, 2019.

[17] Andreas Horni, Kai Nagel y K. W. Axhausen. *The multi-agent transport simulation MATSim*. Ubiquity Press, London, 2016. ISBN 1-909188-77-8.

[18] Rashid A. Waraich, Gil Georges, Matthias D. Galus y Kay W. Axhausen. Adding electric vehicle modeling capability to an agent-based transport simulation. 2014.

[19] Yedi Yang, Jin Li y Qunxin Zhao. Study on passenger flow simulation in urban subway station based on anylogic. *Journal of Software*, tomo 9(1):págs. 140–146, 2014.

[20] Galina Merkuryeva y Vitalijs Bolshakovs. Vehicle schedule simulation with anylogic. En *2010 12th International Conference on Computer Modelling and Simulation*, págs. 169–174. 2010. doi:10.1109/UKSIM.2010.38.

[21] J. Palanca, A. Terrasa, S. Rodriguez, C. Carrascosa y V. Julian. An agent-based simulation framework for the study of urban delivery. *Neurocomputing*, tomo 423:págs. 679–688, 2021. ISSN 0925-2312. doi:https://doi.org/10.1016/j.neucom.2020.03.117. URL https://www.sciencedirect.com/science/article/pii/S0925231220307402.

[22] Francisco Campuzano-Bolarín, Eva Martínez-Caro y Lorenzo Ros-McDonell. Cadenas de suministro tradicionales y colaborativas. *DYNA-Ingeniería e Industria*, tomo 85(1), 2010.

[23] Cipriano Quirós, Javier Portela y Raquel Marín. Differentiated models in the collaborative transport economy: A mixture analysis for blablacar and uber. *Technology in Society*, tomo 67:pág. 101.727, 2021. ISSN 0160-791X. doi:https://doi.org/10.1016/j.techsoc.2021.101727. URL https://www.sciencedirect.com/science/article/pii/S0160791X21002025.

[24] Javier Palanca, Andrés Terrasa, Carlos Carrascosa y Vicente Julián. Simfleet: A new transport fleet simulator based on mas. En Fernando De La Prieta, Alfonso González-Briones, Pawel Pawleski, Davide Calvaresi, Elena Del Val, Fernando Lopes, Vicente Julian, Eneko Osaba y Ramón Sánchez-Iborra, eds., *Highlights of Practical Applications of Survivable Agents and Multi-Agent Systems. The PAAMS Collection*, págs. 257–264. Springer International Publishing, Cham, 2019.

[25] Pasqual Martí, Jaume Jordán, Javier Palanca y Vicente Julian. Load generators for automatic simulation of urban fleets. En Fernando De La Prieta, Philippe Mathieu, Jaime Andrés Rincón Arango, Alia El Bolock, Elena Del Val, Jaume Jordán Prunera,

João Carneiro, Rubén Fuentes, Fernando Lopes y Vicente Julian, eds., *Highlights in Practical Applications of Agents, Multi-Agent Systems, and Trust-worthiness. The PAAMS Collection*, págs. 394–405. Springer International Publishing, Cham, 2020. ISBN 978-3-030-51999-5.

[26] Pasqual Martí, Jaume Jordán, Javier Palanca y Vicente Julian. Charging stations and mobility data generators for agent-based simulations. *Neurocomputing*, tomo 484:págs. 196–210, 2022. ISSN 0925-2312. doi:https://doi.org/10.1016/j.neucom.2021.06.098. URL https://www.sciencedirect.com/science/article/pii/S0925231221016532.

APÉNDICE A
Glosario

- **Algoritmo genético**: tipo de algoritmo de optimización inspirado en los procesos de evolución natural y genética. El algoritmo comienza con un conjunto de soluciones aleatorio y en cada iteración del algoritmo (generación) se combinan (reproducción) o modifican (mutación, selección) las soluciones del conjunto con el objetivo de mejorar el conjunto de soluciones. La bondad de un conjunto de soluciones se calcula mediante una función matemática de utilidad dados una serie de factores ponderados.
- **Atributo**: en programación orientada a objetos, cada una de las variables propias de un objeto. Por ejemplo, un atributo de un coche es su autonomía máxima.
- **Bool**: tipo de datos binario que representa un valor "verdadero" o "falso". Suele emplearse en variables que representen una afirmación o una negación.
- **Buffer**: estructura de datos de tipo pila, siguiendo el esquema FIFO (*First in, first out*). Es conocido por su uso en el almacenamiento temporal de cointenido multimedia en streaming.
- **Callback**: función que se suele incluir como argumento de otra función, de manera que la primera se ejecute al completarse la segunda.
- **Código abierto**: se dice de un proyecto que permite su distribución gratuita e ilimitada a todo el que lo desee, permitiendo acceso al código, y por tanto, su edición. Es el contrario de un producto comercial, que suele requerir de una licencia pagada para utilizarse, no permitiendo su plagio ni su manipulación.
- **Control de versiones**: herramienta que administra los cambios en un sistema de archivos. Incluye un historial de versiones de un proyecto con todas las modificaciones realizadas.
- **Cortocircuito**: en Python, las condiciones booleanas se comprueban de izquierda a derecha. Cuando se detecta que ya está decidido el resultado de una sentencia booleana, detiene la evaluación y devuelve el resultado de manera temprana, dejando sin comprobar el lado derecho de la condición.
- **Dataframe**: en Python, se dice de una serie de objetos indexados por valor. Normalmente contiene una cantidad masiva de información. Su manipulación se realiza mediante el paquete *Pandas*.
- **Descentralización**: se dice de un sistema que no tiene un núcleo, y cuyo flujo se dirige de manera directa entre los miembros sin un mismo intermediario.

- **Diagrama de Voronoi**: se realiza sobre una serie de puntos en un plano. El plano se divide en tantas regiones como puntos contenga, de manera que a cada punto se le asignan los lugares del plano que están más cerca de dicho punto que de los demás.
- **Diccionario**: estructura de datos consistente en pares "clave-valor", de manera que a cada clave le corresponde un único valor o un único objeto. Tiene la ventaja de que el tiempo de lectura de un valor no depende de la posición de la clave dentro del diccionario.
- **Encapsulamiento**: en programación orientada a objetos, se refiere a la práctica de colocar todas las funciones relativas a un mismo objeto o concepto en una misma clase, aislando el código y su comportamiento de otros objetos que pudieran coexistir.
- **Función lambda**: función que no tiene nombre (anónima) y que se suele utilizar como argumento de otras funciones.
- **GeoJSON**: formato estándar para representar colecciones de elementos geográficos sencillos, como puntos, lineas y regiones de un mapa.
- **Interbloqueo**: situación en la que uno o varios agentes de un programa quedan bloqueados esperando que suceda un evento sin poder avanzar en sus funciones.
- **Internet of Things**: describe el concepto de sistemas informáticos en los que cada objeto electrónico dispone de múltiples sensores y capacidad de procesamiento de la información que recibe de su entorno, de manera que se puedan almacenar cantidades ingentes de datos, poniéndolas a disposición de todos los demás objetos de la red para que reaccionen de la manera que les corresponde.
- **Log**: línea de información de control que se imprime en la consola durante la ejecución de un programa. La palabra correspondiente a la jerga española es "diario".
- **Pandas**: librería de Python para el manejo y manipulación de dataframes y, por ejemplo, su introducción o extracción en ficheros de formato Excel.
- **Producto cartesiano**: operación matemática de conjuntos en el que se obtiene toda combinación de elementos posible de ambos conjuntos.
- **Timeout**: en programación, periodo de tiempo en el que se espera un suceso, por ejemplo, la respuesta de la llamada a una página web.
- **XML**: metalenguaje que permite la definición y almacenamiento de información en un formato legible para el ser humano. Por sus siglas en inglés, *eXtensible Markup Language*.

ANEXO B

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**			X	
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**	X			
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**		X		
ODS 10. **Reducción de las desigualdades.**				X
ODS 11. **Ciudades y comunidades sostenibles.**	X			
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**		X		
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

.

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Es una realidad que los coches de combustión tradicionales, tal como los conocemos, pueden desaparecer a medio-largo plazo. Son muchas las razones: el uso de combustibles fósiles, la contaminación, etc. Para la sociedad y el sector industrial mundial, esto supone un gran problema que tiene que ser afrontado antes de que esto suceda. Para resolverlo y mejorar la sostenibilidad del sector automovilístico, muchas empresas comenzaron hace varios años a desarrollar el vehículo eléctrico. Hablamos de un medio de transporte que no emite gases de efecto invernadero mientras se desplaza. El siguiente paso es la salida al mercado de este tipo de transporte, fase que poco a poco coge mayor fuerza por las circunstancias anteriormente enunciadas, la subida del precio de la gasolina etc. No obstante, el coche eléctrico está teniendo una buena acogida en entornos municipales, donde las distancias que se recorren son más cortas y existen más estaciones. Para poder salir de la ciudad y recorrer distancias mayores, es necesario disponer de una red de recarga rápida de baterías que quede al alcance de todo el mundo. Las baterías de estos vehículos no tienen una autonomía tan alta como los coches de diésel y gasolina. Son muchas las investigaciones que, actualmente, estudian la solución al problema de diseñar el emplazamiento óptimo de estaciones de recarga.

En todo caso, los Objetivos de Desarrollo Sostenible de la Agenda 2030 plantean varios puntos relacionados con este Trabajo de Fin de grado, en el que se ha desarrollado una herramienta para realizar experimentos sobre posibles distribuciones de estaciones de recarga por toda España:

Uno de ellos es el **objetivo 7**, que versa sobre fuentes de energía asequibles y no contaminantes. España es un país que reúne muchas condiciones climáticas para ser una potencia en la generación de energías renovables, como la eólica y la solar. Uniendo fuentes de energía verde con la utilización masiva de vehículos eléctricos se lograría reducir las emisiones de gases de efecto invernadero a mínimos. Por eso cobra mucha importancia que se introduzcan medidas para fomentar el uso de estos tipos de transportes, comenzando por aportar herramientas suficientes para que estos puedan circular sin problemas. La principal herramienta es una red de estaciones de recarga en condiciones, con la que poder apoyarse en viajes a cualquier parte de España.

En cuanto a los **objetivos 3 y 13**, la reducción de la contaminación acústica y de la polución del aire contribuiría a mejorar la salud de las personas y a paliar el fenómeno del cambio climático. Estas son otras dos preocupaciones de la sociedad en nuestros tiempos y que tiende a aumentar cada vez más. Tenemos el ejemplo de la ciudad de Madrid, donde se ha tenido que restringir la movilidad de los vehículos más contaminantes para reducir la cantidad de partículas nocivas en el aire que se respira. Además, reduciendo el cambio climático, se ayuda a cuidar especies animales en peligro de extinción, así como ciertas zonas del mundo (como los polos). Todo esto avanza hacia el cumplimiento del **objetivo 11**, ciudades y comunidades sostenibles.

La industria del automóvil tiene que reconvertirse para comenzar a fabricar coches eléctricos y sus baterías, pero las empresas tienen que especializarse también en la construcción de nuevas estaciones de carga y en el diseño de redes alternativas para ubicarlas. Esto está relacionado con el **objetivo 9**, en el que se habla de una nueva revolución industrial sostenible. Una empresa que está trabajando activamente en este objetivo es Tesla, que posee una de las redes de recarga más

potentes de España. En el presente trabajo se ha analizado su manera de ubicar las estaciones y cómo puede mejorar aún más la utilidad de sus usuarios.

En resumen, este proyecto de investigación reúne muchas características alineadas con los Objetivos de Desarrollo Sostenible, sobre todo en términos de sostenibilidad energética y la lucha contra la contaminación y el cambio climático a través de la innovación.

ETS Enginyeria Informàtica
Camí de Vera, s/n, 46022, València
T +34 963 877 210
F +34 963 877 219
etsinf@upvnet.upv.es - www.inf.upv.es