Electric Route

Resumen Ejecutivo

Blanca Alono González
Marta García Palomo
Irene Robles Cabezas
Lucía Tomaino De la Cruz

Contenido

1. Contexto del proyecto

Este proyecto es el resultado del Trabajo Fin de Máster de cuatro compañeras con perfiles muy diferentes y las ideas muy claras. Queríamos demostrar que el avance tecnológico y la sostenibilidad no sólo pueden, sino que tienen que formar parte de la misma ecuación, puesto que no debemos entender el uno sin el otro. Bajo este enfoque y unas grandes dosis de esfuerzo, ilusión y trabajo en equipo nace Electric Route.

2. Introducción, objetivos y alcance del proyecto

Durante los últimos años, el fácil acceso a la movilidad global se ha convertido en una de las principales causas de los altos índices de contaminación generados por el ser humano. Alcanzar, en este aspecto, espacios más sostenibles y limpios es el gran reto de todos en la lucha contra el cambio climático. Una de las medidas adoptadas para lograr este objetivo ha sido impulsar la movilidad sostenible y, ahí, es donde entra en juego este proyecto.

Electric Route pretende sumarse a ese cambio fomentando el uso de los coches eléctricos y, con este propósito, abordar una solución al hándicap fundamental de las rutas de larga distancia, la baja autonomía de los coches. A pesar del incremento de las ventas que está experimentando el sector del motor eléctrico, en la actualidad, siguen siendo muchos los posibles compradores interesados que, ante la falta de facilidades e información para planificar sus viajes, se echan para atrás.

Electric Route se trata de una aplicación web que, a partir de dos puntos de la Península Ibérica, origen y destino, y, seleccionado un coche eléctrico específico, recomendará una ruta óptima en tiempo para ese recorrido. En el cálculo de la ruta se tendrán en cuenta variables como la autonomía del coche, así como la carga inicial y final del mismo, de manera que, si el viaje se encuentra limitado por la batería del coche, se le recomendará al usuario un punto de recarga en cualquier parte del trayecto.

Además, en caso de no existir un punto de recarga adecuado para el coche eléctrico en cuestión, debido a la incompatibilidad con el conector, se recurrirá a una red que combina los puntos de recarga reales con puntos de recarga ficticios, situados en estaciones de servicio. A través de estas recomendaciones se sugieren localizaciones donde resultaría conveniente instalar nuevos puntos de recarga y, a su vez, se pone de manifiesto cómo la falta de estándares en cargadores, está frenando el progreso del sector.

Para el desarrollo del proyecto se han utilizado diferentes herramientas de software. Github como control de versiones, VirtualBox para el desarrollo de la aplicación en local, MySQL como sistema de gestión de las bases de datos incorporadas, Python como lenguaje de programación, Flask para crear la aplicación web, Grafana para la visualización de los datos, Docker/Docker-compose para el despliegue de la aplicación dentro de contenedores y Google Cloud Platform como plataforma para productivizar y desplegar el sistema de la aplicación en un entorno remoto. Por otro lado, cabe destacar el uso de algunas APIs de Google o de Tomtom para obtener cierta información.

En relación al conjunto de datos con el que se trabaja, proviene de diversas fuentes y se vuelca en distintas tablas de la base de datos. Una de ellas incluye información relativa a los coches eléctricos; otra, recoge los distintos puntos de recarga existentes en la Península Ibérica y, tal y como se ha mencionado con anterioridad, se obtiene información referente a las estaciones de servicio. Además, durante el planteamiento del problema, surge una nueva tabla que contiene datos de interés sobre las ciudades de la península.

3. Metodología y procedimientos

El núcleo de la aplicación se basa en el cálculo de rutas óptimas para vehículos eléctricos, para el cual se ha planteado una red compleja dirigida donde los nodos son los puntos de origen/destino y los puntos de recarga/estaciones de servicio propuestas, y, las aristas tienen un peso igual al tiempo que implica unir dos nodos. Partiendo de esto, se aborda el problema del camino más corto.

Los parámetros de entrada que necesita el modelo para funcionar son: tipo de estacionamiento (dónde realizar las paradas, puntos de recarga o estaciones de servicio), origen y destino, marca y modelo de coche, carga inicial y final de la batería.

A continuación, se describe de manera exhaustiva las distintas facetas del modelo.

3.1. Definición de la función objetivo y restricciones

El primer paso para plantear el problema como un grafo con nodos y aristas es definir tanto la función objetivo que hay que minimizar (esto es, el peso de las aristas) como las restricciones que hay que aplicar a la red para que el problema no alcance soluciones no válidas físicamente.

3.1.1. Función objetivo

La función objetivo tiene en cuenta tanto el tiempo empleado en desplazarse de un nodo a otro, así como el tiempo estimado en la parada.

Tiempo del trayecto: se calculará como el cociente entre la distancia entre nodos y una velocidad media que se define como constante (100 km/h). Hay que tener en cuenta que la distancia calculada entre dos nodos equivale a la obtenida con la función de Haversine, sin considerar la distancia por carretera. Esto implica por lo general una subestimación del tiempo del trayecto, por lo que se aplica un factor corrector del 120% para compensar este hecho.

Tiempo de parada: se obtendrá como resultado de la suma entre el tiempo de carga del coche y una estimación del tiempo de espera en la electrolinera.

- **Tiempo de carga**: es el tiempo que tarda en cargar el coche. Se calcula como el cociente entre la capacidad de la batería del coche (energía que es capaz de acumular medida en kWh) y la potencia que le proporcione el punto de recarga donde efectúa la parada (medida en kW). Además, se tiene en cuenta un rendimiento de carga del 90%, por lo que el valor de tiempo de carga obtenido se divide por este coeficiente.
- **Tiempo de espera**: se trata del tiempo que el usuario ha de esperar en la cola de la electrolinera para cargar su coche. Uno de los obstáculos con los que se ha tenido que lidiar en el cálculo de la variable tiempo de espera ha sido la escasa disponibilidad de datos reales relacionados con la afluencia de coches eléctricos en los puntos de recarga. Con el objetivo de abordar este problema bajo una cierta coherencia estadística, se ha adaptado un modelo basado en la teoría de colas. No obstante, debido al inconveniente mencionado, se han tenido que asignar valores ficticios a algunas de las variables que intervienen en el modelo en cuestión.

La formulación matemática de lo explicado es la mostrada a continuación:

$$\min t = \frac{\sum_{i=1}^{k-1} d_i}{V_{me}} + t_{parada}$$

d_i = distancia entre nodos
k=número de paradas
V_{me} = velocidad media del coche (100 km/h)
t_{parada} = tiempo de parada.

1.1.1. Restricciones

Se restringe la red sobre la que se trabaja mediante diversas restricciones en desigualdad diseñadas para cumplir requisitos, a veces imprescindibles, de cara al correcto funcionamiento del problema.

Restricciones de obligatoriedad

- **Restricción autonomía**: La distancia entre un nodo y el consecutivo ha de ser siempre menor o igual que el 90% de la autonomía del coche, puesto que si no sería imposible el cálculo de la ruta. Se establece un margen de al menos el 10% de la batería para evitar que, en ningún caso, la batería del coche pueda descargarse por completo antes de llegar a la electrolinera.

$$d_{i,i+1} \leq 0.9 * autonomía \ \forall_i$$

$d_{i,i+1}$ = distancia entre dos nodos consecutivos de la red
$autonomía$ = autonomía del coche
$\forall_i$ = para todo nodo comprendido en la red

- **Restricción de tipo de conector**: Para un coche dado (modelo y marca), hay uno o varios tipos de conectores permitidos. Por este motivo, los puntos de recarga seleccionados han de cumplir con la condición de tener un conector del tipo correcto para el coche en cuestión, por lo que se desestiman aquellos puntos de recarga de la red que no cuentan con un tipo de conector compatible.

Restricciones de no obligatoriedad

Este tipo de restricciones son implementadas como parte de un modelo más avanzado que el propuesto en base a las restricciones obligatorias, las cuales, permitirán ajustar el cálculo de las rutas a las necesidades específicas de cada usuario.

- **Restricción primera parada**: La distancia entre el nodo origen y el siguiente nodo ha de ser menor o igual que la distancia que puede recorrer el coche entre esos dos puntos en función de la carga del coche al principio del trayecto. Debe recordarse que, por defecto, el coche siempre llegará al menos con un 10% de batería al punto de recarga.

$$d_{i,i+1} \leq 0.9 * autonomía \ \forall_i$$

$d_{i,i+1}$ = distancia entre dos nodos consecutivos de la red
$autonomía$ = autonomía del coche
$\forall_i$ = para todo nodo comprendido en la red

- **Restricción última parada**: La distancia entre el nodo destino y el nodo inmediatamente anterior ha de ser menor o igual que la distancia que se permita recorrer en función de la carga final con la que desea llegar el usuario a su destino.

$$d_{k+1} \leq \frac{(100 - C_f)}{100} * (0.9 * autonomía)$$

d_{k+1} = distancia entre la última parada y el destino
$autonomía$ = autonomía del coche
C_f = carga con la que finaliza el trayecto el coche

1.2. Cálculo de la matriz de distancias

En el momento de plantear el problema como una red compleja compuesta por nodos y aristas (distancias/tiempo), surge la necesidad de calcular una matriz de distancias entre todos los puntos de la red, esto es, ciudades, puntos de recarga y estaciones de servicio. Se decide calcular esta matriz con ayuda de la distancia de Haversine.

Para su construcción, se definen dos funciones, una que toma como argumentos las coordenadas de dos puntos y, aplicando la distancia de Haversine devuelve la distancia entre ellos. Y, una segunda función, que construye iteraciones entre dos filas consecutivas de un dataframe, en este caso se devuelve con la función zip una tupla, clave-valor, con el valor del dataframe sobre el que se itera. Finalmente, dado un dataframe que contiene la información sobre todos los puntos de la red, se recorre mediante dos bucles para calcular la distancia. El dataframe resultante está compuesto por tres variables: origen, destino y distancia entre ambos puntos.

1.3. Cálculo de ruta óptima entre dos puntos

Una vez definida tanto la función objetivo y las restricciones a aplicar como la construcción de la matriz de distancias entre todos los posibles nodos, se puede abordar la descripción del algoritmo de optimización utilizado para obtener la ruta entre dos puntos.

Como se mencionaba, el problema que se trata resolver es el del camino más corto, esto es, encontrar un camino entre dos nodos tal que la suma de los pesos de las aristas que lo constituyen sea mínima. De esta manera, se plantea una red compleja dirigida donde los nodos son los puntos de origen/destino y los puntos de recarga/estaciones de servicio propuestas, y las aristas se generan entre todos los nodos de manera bidireccional. El peso de las aristas es igual al tiempo obtenido con la función objetivo.

Para realizar este proceso de generación de red compleja y optimización de la ruta, se ha recurrido a la librería *networkx*. Esta propone una serie de algoritmos de optimización de grafos tales como el *algoritmo de Dijkstra* o el de *A **.

La idea subyacente del *algoritmo de Dijkstra* consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen hasta el resto de los vértices que componen el grafo, el algoritmo se detiene.

Partiendo de esta base, el *algoritmo de A ** es una variante optimizada del de *Dijkstra* que trata de buscar un camino mejor mediante el uso de una función heurística que da prioridad a los nodos que se supone que son mejores que otros, mientras que *Dijkstra* simplemente explora todos los caminos posibles.

Tras analizar ambas opciones, se decide elegir el *algoritmo A ** como base para el optimizador de rutas de Electric Route, dado que las soluciones encontradas por éste llevan a un tiempo total de viaje equiparable al de *Dijkstra* y, sin embargo, conlleva un menor tiempo de ejecución.

2. Resto de secciones de la memoria

2.1. Herramienta Web: Electric Route

Electric Route consta de una interfaz gráfica simple e intuitiva para el usuario, que le permitirá navegar con facilidad. Se decide utilizar como FrontEnd el microframework de Python Flask, desarrollando una aplicación web que permita a los usuarios registrarse y simular sus rutas, así como acceder e interactuar con la información en la que se basa el modelo. En este apartado se detalla la estructura y la configuración de la herramienta, así como una guía al usuario.

2.2. Productivización

La productivización del sistema que permite el correcto funcionamiento de la aplicación Electric Route en un entorno operacional se basa en el uso de dos tecnologías distintas: Docker/Docker-Compose y Google Cloud Platform. Mientras que la primera permite la virtualización de aplicaciones en

contenedores de manera automática y la interacción entre distintos contenedores, la segunda proporciona una plataforma de computación remota en la nube donde desplegar el sistema.

En esta sección se explica con detalle los distintos contenedores que componen el sistema de la aplicación, y la utilización de Google Cloud Platform para realizar el despliegue de la misma.

2.3. Próximos pasos

Electric Route ya es una realidad, pero solo es el principio y aún queda mucho por hacer, por eso en esta última sección se detallan los siguientes pasos en los que se está trabajando y en los que centrarnos próximamente.

Esto abarca desde la actualización de los datos en los que se basa el algoritmo hasta la aplicación en sí misma. Mejorar la oferta de puntos de origen y destino o incluso añadir la función en tiempo real, geolocalizando la posición del usuario o incluir datos del tráfico para una estimación de tiempos más exacta.

Uno de los mayores factores limitantes de la aplicación actualmente es el tiempo de ejecución, debido al hecho de que Python es un lenguaje interpretado. Para poder llevar a cabo los puntos comentados y no impactar en el tiempo de ejecución de manera excesiva, se proponen dos opciones: o bien utilizar tecnologías tipo Hadoop/Spark que permiten procesar grandes volúmenes de datos, o bien, la introducción de las partes de código que más tiempo de ejecución conllevan en lenguaje compilado.

Por otro lado, la opción de adaptar esta aplicación al móvil sería de gran utilidad ofreciendo así un servicio más óptimo y fácil al usuario.

3. Comparativa con proyectos similares

A día de hoy, existen otros recomendadores para coches eléctricos, como son EV Navigation, Sygic GPS Navigation EV, Open Charge Map o Google Maps. No obstante, lejos de suponer un obstáculo para nosotras y aunque somos conscientes que Electric Route aún no puede competir con apps que ya están en el mercado, es una motivación para continuar con las mejoras de la aplicación, ya que consideramos que es un sector en auge.

Entre las aplicaciones comentadas no existen grandes diferencias, pues la idea global es similar, todos ellos para la planificación de la ruta tienen en cuenta los puntos de recarga compatibles con el usuario, así como los parámetros más específicos de cada coche, como son la capacidad de la batería, la potencia de recarga, el consumo promedio y el tipo de conector. Alternativamente, alguna de ellas también ofrece al usuario la opción de ajustar estos parámetros en función de su experiencia real, o la opción del pago de la recarga previamente.

Resulta satisfactorio darse cuenta que utilizamos el mismo algoritmo de optimización que google maps, A* y, con unos recursos muy limitados, haber alcanzado resultados óptimos.