Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño y aplicación de técnicas metaheurísticas en transporte multimodal

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Claes, Sammy Jonathan

Tutor: Garrido Tejero, Antonio

[2021-2022]

Resumen

Este trabajo se centra en el diseño y la implementación de un algoritmo genético que resuelve el problema de la ruta más corta y/o menos contaminante en la ciudad de Valencia y sus alrededores. Para ello, se utilizarán los medios de transporte públicos más usados de la ciudad para moverse (EMT Valencia, Metrovalencia y Valenbisi).

El algoritmo ofrecerá un camino multimodal entre dos puntos de la ciudad, con la novedad de que se tendrán en cuenta también aspectos relacionados con el medio ambiente, lo que permitirá moverse por la ciudad de una forma más sostenible.

El sistema incluye la posibilidad de moverse con un solo tipo de transporte o una combinación de ellos, además de poder elegir tres tipos de rutas: la más rápida, la menos contaminante o un equilibrio entre ambas. Asimismo, se mostrará información sobre el número de kilómetros a recorrer, el tiempo de desplazamiento y las emisiones de CO_2 emitidas a la atmósfera por cada viaje.

Pese a que el núcleo de este trabajo sea el diseño e implementación de un algoritmo genético, también se ha desarrollado una aplicación móvil básica para poder demostrar el funcionamiento del algoritmo.

Para llevar a cabo la implementación de este proyecto se ha hecho uso principalmente de las tecnologías y lenguajes Android Studio, Java, SQL y las API de Google Routes y Valenbisi. También se ha utilizado la metodología Kanban a la hora de implementar el algoritmo.

Palabras clave: Algoritmo genético, Metaheurísticas, Android, GTFS, Multimodal, CO_2, Kanban.

Abstract

This work focuses on the design and implementation of a genetic algorithm that solves the problem of the shortest and/or least polluting route of the city of Valencia and its periphery. To achieve this, the most used public transport in the city will be used to move (EMT Valencia, Metrovalencia and Valenbisi).
The algorithm will offer a multimodal path between two points of the city, with the novelty that will also consider aspects related to the environment, which will allow moving around the city in a more sustainable way.

The system includes the possibility to move with a single transport or a combination of them, in addition, the user will be able to use three different paths: the fastest, the least polluting or a balance between them. Moreover, the system will provide information about the miles of highway, travel time and the CO_2 emissions emitted to the atmosphere.

Although the core of this work is the design and implementation of a genetic algorithm, a basic mobile application has also been developed to demonstrate the functioning of the algorithm.

To carry out the implementation of this project we have made use mainly of the technologies and languages Android Studio, Java, SQL, and the APIs of Google Route and Valenbisi. The Kanban methodology has also been used to implement the algorithm.

Keywords: Genetic algorithm, Metaheuristics, Android, GTFS, Multimodal, CO_2, Kanban.

Tabla de figuras

Tablas

Tabla de contenidos

1. Introducción

Este trabajo final de grado se desarrolla en base a una propuesta del tutor. Viene motivado por la necesidad de ofrecer rutas que tengan en cuenta distintos modos de transporte en una ciudad y buscar rutas más sostenibles desde un punto de vista medioambiental. Para ello, se utilizarán los conocimientos adquiridos sobre las técnicas, entornos y aplicaciones de la inteligencia artificial que el alumnado ha aprendido durante sus estudios en la universidad.

Dichas técnicas se plantean poner en práctica para desarrollar un algoritmo que permita resolver un problema real, encontrar el mejor camino entre dos puntos de la ciudad de Valencia en base a dos objetivos, obtener la ruta más corta o menos contaminante, usando para ello un enfoque multimodal que incluya los transportes públicos disponibles dentro de la ciudad y su periferia. De esta forma, se pretende que los usuarios puedan tener el control de la ruta al mismo tiempo que sean capaces de moverse de manera más sostenible, sin perder una funcionalidad que muchos utilizan en su día a día, el poder llegar a un destino de la manera más rápida posible.

Para la realización de este trabajo se desarrollará una aplicación móvil dividida en tres partes. En la primera de ellas, se desarrollará un algoritmo genético parametrizable por el usuario. Dicho algoritmo calculará una serie de rutas que incluyan información sobre el tiempo de desplazamiento, número de kilómetros a recorrer y las cantidades de CO_2 que se emiten a la atmósfera por cada viaje y, mediante una serie de operadores, devolverá la mejor solución encontrada. La segunda parte se encargará de gestionar un servicio para obtener los horarios y las rutas de los diferentes medios de transportes de la ciudad, además de la localización de todas las estaciones que intervienen en el sistema. Para realizar este propósito, se indexarán una colección de ficheros con la información proporcionada por las empresas de transporte y se guardarán en un sistema gestor de base de datos para su uso en el algoritmo genético. Finalmente, se hará el desarrollo de una interfaz gráfica, es decir, toda la parte visual de la aplicación, para mostrar al usuario toda la información relativa sobre la ruta devuelta por el algoritmo genético. En esta parte, también se mostrará de forma gráfica, como evoluciona el algoritmo genético mediante un mapa disponible desde la pantalla principal.

1.1 Motivación

La motivación de este trabajo es doble, en primer lugar, existe una motivación personal hacia el medio ambiente. Siempre he querido poder utilizar mi formación para poder desarrollar nuevas funcionalidades que mejoren la calidad de vida de las personas, además del mundo que les rodea. En este contexto, los estudios cursados han sido clave para poder empezar a diseñar y producir proyectos que vayan en concordancia con esta visión. Por otra parte, uno de los grandes problemas que sufren las grandes ciudades es la contaminación. La movilidad con vehículos motorizados interviene directamente en esta problemática: no solo se generan atascos, sino también un impacto ambiental negativo. Si se apuesta por medios de transportes ecológicos y

sostenibles se pueden disminuir problemas como los atascos, al mismo tiempo que se consigue una incidencia positiva en la salud de la población.

Otro aspecto para destacar es el poder añadir funcionalidades nuevas a las aplicaciones convencionales de obtención de rutas. La primera de ellas es poder moverse de una forma más sostenible dentro de una ciudad, y de esta forma, evitar y reducir las emisiones de CO_2 que se emiten a la atmósfera por cada viaje. Asimismo, se desea poder centralizar los medios de transporte públicos de una ciudad y, de esta manera, poder presentar rutas en las que se contemplen una combinación de estos transportes, ya que, por ejemplo, en Valencia, estos medios pertenecen a empresas que operan en sistemas completamente diferentes. En consecuencia, las rutas ofrecidas incluyen únicamente los modos de viajes de una empresa y no se contemplan caminos en los que sea posible utilizar varios de ellos.

1.2 Objetivos

El objetivo del trabajo es diseñar y desarrollar un algoritmo genético que resuelva el problema de la ruta más corta y menos contaminante dentro de Valencia y su periferia, usando para ello, los medios de transporte públicos disponibles en la ciudad. Para poder alcanzar el objetivo principal del proyecto será necesario cumplir con una serie de objetivos específicos que se detallan a continuación:

- Plantear y formalizar el problema para definir los objetivos a optimizar.
- Diseñar e implementar los parámetros de un algoritmo genético para que sea eficiente y capaz de encontrar una solución prometedora en pocos segundos.
- Mejorar las soluciones ofrecidas por otras aplicaciones.
- Ofrecer rutas menos contaminantes.
- Ofrecer una combinación de modos de transporte con los que desplazarse por la ciudad.
- Diseñar la arquitectura y el modelo de datos para incluir la información de los distintos medios de transporte a utilizar.

El desarrollo de la parte visual de la aplicación no es un objetivo del proyecto. No obstante, se ha implementado una interfaz básica, pero completamente operativa con la finalidad de que el usuario final pueda probar la aplicabilidad del sistema.

1.3 Estructura

La memoria está organizada en siete apartados. En el primero de ellos, se encuentra la introducción, la motivación, los objetivos generales y la estructura de la memoria.

Respecto al contenido del segundo punto, se plantea la metodología empleada para el desarrollo de las distintas tareas que se han necesitado realizar para llevar a cabo este proyecto, mostrando como se ha planteado la fase de desarrollo y de pruebas para garantizar el correcto funcionamiento del algoritmo genético.

En tercer lugar, se encuentra el marco teórico, en el que se define lo que es el transporte multimodal y la huella de carbono. Asimismo, se explica en qué consiste un problema de optimización y se expondrán los elementos que intervienen en un algoritmo genético.

Por otro lado, en el cuarto punto, se expone el diseño de la solución usada en este proyecto. Para ello se presenta una instancia simple del problema que se va a resolver, proporcionando al lector una visión general de todos los elementos que intervienen en la obtención de la ruta multimodal. Asimismo, se analiza el coste teórico del problema, lo cual permite motivar el uso del algoritmo genético. Después, se presenta el estado del arte, en el que se detallan dos aplicaciones similares que existen actualmente. También se muestra una tabla comparativa entre las aplicaciones y lo que se pretende conseguir en el desarrollo del algoritmo genético. Finalmente, se detalla la obtención de los transportes públicos que se usan en el proyecto y se plantea el algoritmo genético usado con unos ejemplos simples, mostrando además una heurística diseñada por el alumno para acelerar el algoritmo genético.

Tras haber terminado con la etapa de diseño, se realizará la fase de implementación en la cual se detallarán las diversas tecnologías y herramientas utilizadas para el desarrollo de la aplicación. Una vez se haya concluido la etapa de implementación, se mostrarán los resultados obtenidos.

Finalmente, se incluye un apartado de conclusiones en el cual se analizarán los objetivos conseguidos y como el grado ha ayudado al alumno a poder realizar este trabajo. Asimismo, el lector podrá consultar la bibliografía que se ha usado para la obtención de la información necesaria para la realización de este proyecto y consultar el código desde los anexos.

2. Metodología

Este apartado detalla la metodología usada para la realización de cada uno de los objetivos del proyecto. Para este trabajo se ha utilizado la metodología Kanban. A continuación, se detallan sus aspectos principales además de los pasos seguidos para implementarla en el trabajo.

La metodología Kanban permite realizar desarrollos eficientes y efectivos. Forma parte de las metodologías ágiles y su objetivo es gestionar la realización de una serie de tareas desde su inicio hasta su finalización. Permite a las personas centrarse en las tareas a realizar y limitar la cantidad de actividades que se intentan completar. Es muy sencilla de aplicar y es un método muy visual que permite conocer el estado actual del trabajo en cada momento, además de priorizar aquellas tareas que sean más importantes. Para aplicarlo, es necesario un tablero dividido en una serie de columnas, y en estas, anotar el estado del flujo por las cuales irán pasando las tareas. También, cabe destacar que este método se basa en el desarrollo incremental, es decir, en la división del trabajo en diferentes partes, lo que permite la agilización en el proceso de producción [1].

Para llevar a cabo este método se ha utilizado la herramienta de *Azure Boards* y se ha creado un tablero dividido en las siguientes columnas:

- ***New*** (Nuevo): en esta sección se generan nuevas tareas y se dividen en una o varias tarjetas, dependiendo del tamaño de la evolución a realizar. En cada tarjeta se añade la información necesaria para conocer la carga total de trabajo que va a suponer y se describe cualquier tipo de observación.
- ***Preparation*** (Preparación): seguido de *New,* en esta sección se comprueba que la tarea está claramente definida y se investigan los aspectos necesarios para poder integrar la evolución al proyecto.
- ***Doing*** (Haciendo): una vez la evolución está clara, se lleva a cabo la integración de la tarea, que puede comprender un desarrollo o escribir una nueva sección de la memoria del trabajo.
- ***Testing*** (Pruebas): en esta sección se somete a prueba la evolución hecha en caso de que implique un desarrollo y se verifica que todas y cada una de las funcionalidades implementadas cumplen con los requisitos definidos en las tareas.
- ***Closed*** (Cerrado): cuando la tarea ha finalizado y se ha revisado que todo funciona correctamente, la tarea pasa a esta fase y se puede dar por terminada.

Cada columna dispone de un límite de tareas que puede contener (a excepción de las columnas Nuevo y Cerrado). De esta forma, se proporciona un método para no empezar nuevas tareas sin antes haber terminado con las que ya hay en curso.

Asimismo, las tareas a realizar se han dividido en una serie de elementos de trabajo. A continuación, se detallan sus características:

- ***User Story*** (Historia de usuario)*:* en esta tarjeta se proporciona una explicación general e informal de una evolución escrita desde la perspectiva del usuario final. Cada historia de usuario se divide en una descripción, que articula la funcionalidad a añadir y una serie de criterios de aceptación en formato de lista,

que enumeran las condiciones que se deben cumplir para considerar la tarea como completada. Las historias de usuario utilizadas se dividen en dos tipos; de desarrollo para las evoluciones del algoritmo y de documentación para las que guarden relación con la escritura de la memoria.

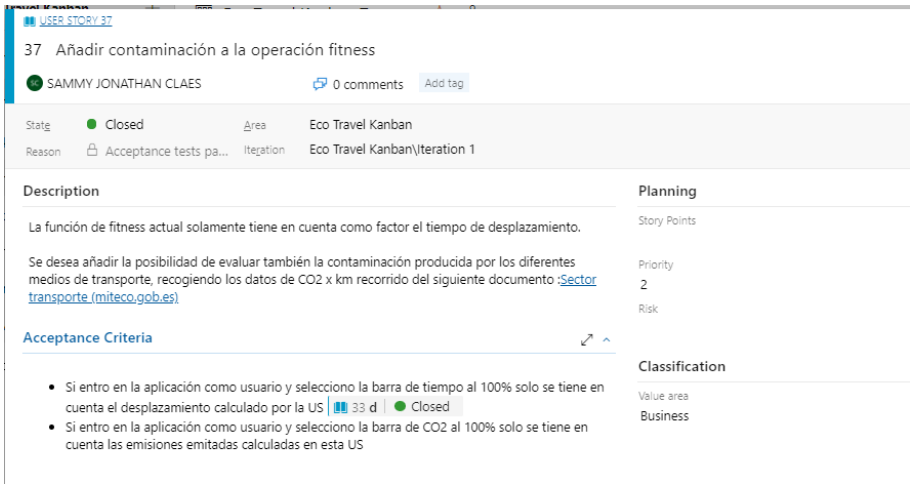

Ilustración 1. Contenido de una historia de usuario

- ***Test*** (Prueba)*:* estas tarjetas son una herramienta para hacer pruebas manuales y exploratorias sobre una evolución. Permiten definir, sin ambigüedad una serie de pasos que aseguran que cada uno de los resultados satisface las necesidades descritas en la historia de usuario. De esta forma, se identifican todos los impactos causados por un desarrollo y se localizan errores de funcionamiento en el momento de la evolución. Su ventaja es clave, se consigue una ganancia de tiempo importante, al detectar los errores cuando ocurren y no, en fases posteriores que puedan incluir una dependencia con otros módulos.

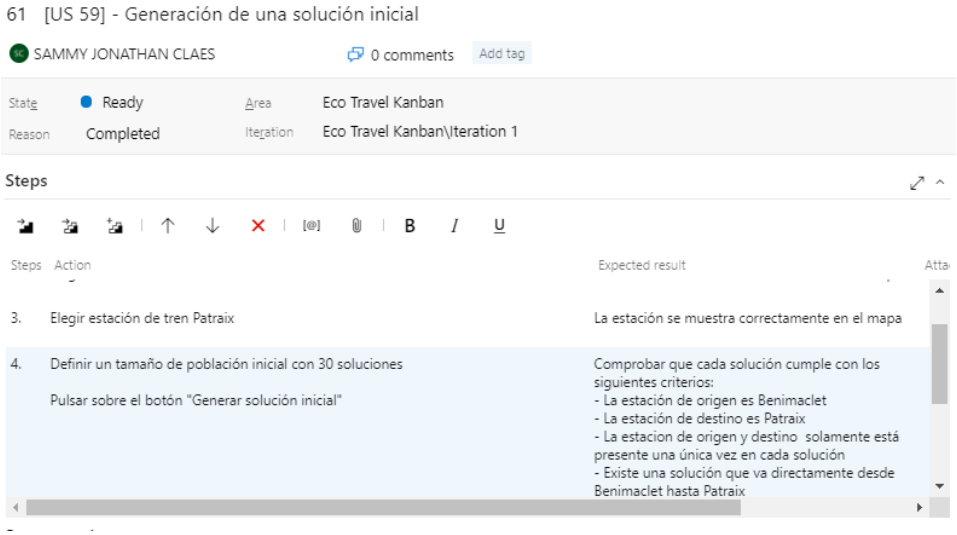

Ilustración 2. Contenido de un "test"

- ***Bug****:* en ocasiones, no se detectan todos los impactos en la fase de pruebas o sucede que un desarrollo impacta sobre evoluciones ya cerradas. Esta tarjeta permite solucionar este tipo de situaciones. Son parecidas a las historias de usuario, pero con una sección añadida para describir como reproducir el error.

- ***Task*** (Tarea): este componente se adjunta a los elementos de trabajo descritos anteriormente y se utilizan como un elemento de chequeo. Disponen de cinco estados (nuevo, activo, terminado y borrado) y sirven para tener constancia del estado del trabajo pendiente y ya finalizado.

Para llevar a cabo la implementación de estos elementos con la estrategia Kanban se ha empleado el siguiente flujo de trabajo. En primer lugar, se crean los elementos de trabajo necesarios dentro de la sección Nuevo. Si la evolución corresponde a una historia de desarrollo, se crean dos *Task (*"Dev" para la parte de desarrollo y "Test" para la parte de pruebas*)* y si corresponde a redactar aspectos sobre el trabajo, solamente una ("Doc" para la parte de documentación). Después se pasa la tarjeta a Preparación, donde se analizan aspectos relevantes sobre la evolución a tratar, y si todo está claro, el flujo continúa hacia Haciendo, donde se inicializan las *task* al estado activo. Durante esta fase, se realiza el desarrollo, la redacción de pruebas y la escritura de la memoria si procede. Finalmente, se pasa la evolución a columna de pruebas y una vez finalizada, a la sección terminado.

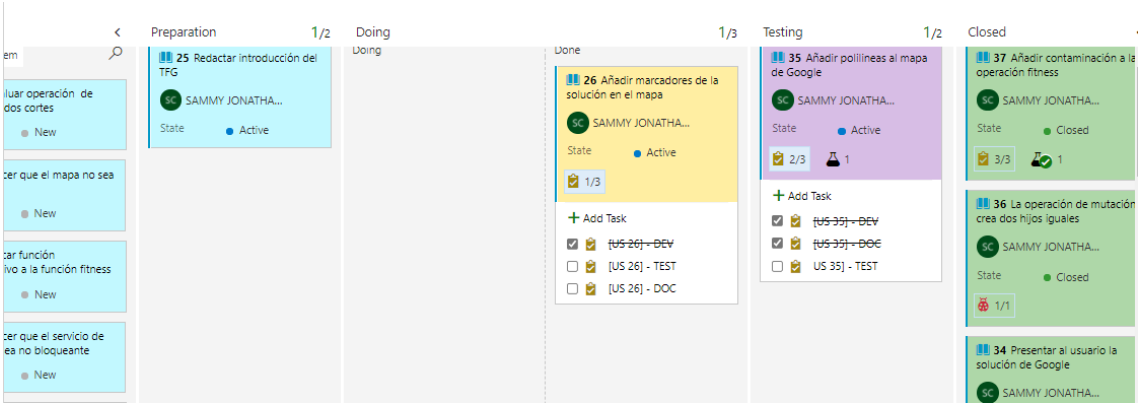

Ilustración 3. Ejemplo de tablero Kanban

3. Marco teórico

Esta sección tiene como objetivo dar las bases teóricas necesarias para que el lector comprenda de manera clara los pilares sobre los que se sustenta este proyecto. Para ello en primer lugar se definirá en qué consiste el transporte multimodal y se hablará sobre la huella de carbono y los gramos de CO_2. Asimismo, se expondrán los elementos que intervienen en un problema de optimización, y, finalmente se dará paso a explicar las bases fundamentales de un algoritmo genético.

3.1 Qué es el transporte multimodal

Para poder empezar a definir la problemática debemos de entender qué es el transporte multimodal. Podemos definirlo como el traslado de algún bien desde su procedencia hasta su destino, usando para ello, la articulación de dos o más modos de transporte [2]. En el contexto del proyecto, estos pueden ser buses, tranvías, metros y bicicletas de alquiler. Su característica principal es que se emplea el vehículo más adecuado en cada momento, usando para ello las mejores prestaciones de cada modo. A continuación, se muestran dos ejemplos:

- Un estudiante coge el tranvía desde la universidad Politécnica hasta Benimaclet y desde Benimaclet toma un metro hasta la estación de Jesús. En este caso, se trata de un desplazamiento con transporte multimodal, ya que ha habido un cambio de modo entre dos transportes.
- Un profesor coge una bicicleta desde la universidad Politécnica hasta su casa. En esta situación no se trata de un desplazamiento que implique un modo multimodal.

3.2 Huella de carbono

La huella de carbono representa el volumen total de gases de efecto invernadero que producen las actividades cotidianas del ser humano. Gran parte de las consecuencias del cambio climático son debido al aumento de estos gases, en el que se incluye el dióxido de carbono (CO_2). Dentro de este contexto, el transporte por carretera y otros medios de transporte producen una gran cantidad de CO_2, para ser exactos, el sector, transporte representa el 25% de las emisiones totales de gases de efecto invernadero en España y casi el 40% de las emisiones de los sectores difusos. Por modos de transporte, la carretera representa casi el 95% de las emisiones, mientras que la contribución de otros modos de transporte es bastante más minoritaria [3]. Así pues, el sistema de transporte empleado impacta en gran medida a nuestra huella de carbono, por lo que reducir o eliminar los gases de efecto invernadero generados a partir de nuestros desplazamientos es crucial para frenar las consecuencias del cambio climático.

Pero ¿cómo se calcula la huella de carbono? En este proyecto nos hemos basado únicamente en el dióxido de carbono, que es el gas mayoritariamente emitido por los vehículos, aunque no el único. Existen otros como el monóxido de carbono (CO) o el óxido de nitrógeno (NOx). Para el cálculo de la huella de carbono, se ha utilizado como métrica, los gramos de CO_2 que emite un vehículo por cada kilómetro recorrido. En el caso de los modos de transporte usados en este proyecto, la bicicleta es el medio de

transporte más ecológico, ya que no contamina, y, por lo tanto, no emite ningún gas a la atmósfera. Lo mismo ocurre con el tranvía y el metro en Valencia, estos transportes son eléctricos, y desde 2018 funcionan con energía 100% renovable [4]. En el caso del autobús, sí que existen emisiones. EMT Valencia, la principal entidad que da servicio de transporte público en la ciudad dispone de varios modelos no eléctricos. Para obtener su consumo de carburante, se pueden consultar bases de datos oficiales como la creada por el Instituto para la diversificación y Ahorro de la Energía (IDAE), que muestra información detallada sobre su consumo (ver ilustración 4). Para vehículos más nuevos, el cálculo es directo, ya que el factor de emisión aparece expresado en gCO2 /km. En caso de que el modelo de vehículo no se encuentre entre los datos disponibles se puede estimar a partir del dato "*litros de combustible consumido por km recorrido*" que aparece en las especificaciones técnicas de un vehículo, por lo que, multiplicando el número de kilómetros recorridos por este factor podemos obtener una huella de carbono aproximada al hacer uso de estos transportes.

Modelo	Clasificación Energética	Consumo (*)		Emisiones (gCO_2/km) (**)	
		Mínimo	Máximo	Mínimo	Máximo
Mercedes-Benz CLA Shooting Brake (X118) 180 aut.	B	6,1	6,8	139,0	154,0

Ilustración 4. Información sobre un vehículo en IDAE

3.3 Optimización

Como se ha comentado en la introducción, uno de los objetivos del proyecto es poder encontrar un camino en base a dos objetivos, la ruta más corta o menos contaminante. Estos objetivos buscan finalidades diferentes. Seguramente para poder llegar más rápido a un destino sea necesario utilizar modos de transportes contaminantes, mientras que para la ruta menos contaminante sea necesario evitarlos utilizando la bicicleta o yendo a pie. Para resolver esta problemática es necesario hablar de optimización. En esta sección se pretende explicar al lector, lo qué es un problema de optimización y sus características principales. Asimismo, se mostrará una manera sencilla para resolver objetivos contradictorios, la cual, ha sido usada en el desarrollo del algoritmo genético.

Todos los días, nos enfrentamos a problemas que requieren encontrar la mejor solución posible, como puede ser la obtención de la ruta más corta para llegar al trabajo. En el campo de la ingeniería, el término de optimización es usado para resolver estos problemas complejos. La optimización puede ser definida como el proceso de maximizar o minimizar el valor de una función, escogiendo sistemáticamente un conjunto de valores de unas variables, dentro de un espacio de búsqueda permisible. Concretamente, un problema de optimización está formado por los siguientes elementos:

- **Variables de decisión**: corresponden a cantidades numéricas cuyos valores deben ser encontrados para poder alcanzar la solución óptima. Se suelen denotar como X_i con i = 1, 2, …, n y permiten modelizar los elementos del problema.

- **Restricciones**: en los problemas de optimización las restricciones son impuestas por las características del ambiente o por la disponibilidad de los recursos, como por ejemplo restricciones de tiempo o de capacidad. Estas restricciones describen la dependencia entre el valor de las variables decisión a encontrar y el espacio de búsqueda permisible.
- **Funciones objetivo**: la forma de conocer que tan buena es una propuesta, es por medio de uno o varios criterios que permiten evaluar las variables decisión. Estos criterios expresados como funciones son llamadas funciones objetivo y suelen denotarse como $f(X)$.

Asimismo, dependiendo del problema, es posible que se necesite encontrar una solución basada en uno o varios objetivos. Dependiendo del problema podemos encontrar de dos tipos:

- **Optimización mono-objetivo**: cuando es necesario resolver un problema cuyo único fin es encontrar el mejor valor de una única función objetivo, se habla de optimización mono-objetivo. Dicho valor dependerá del tipo de problema a resolver, habiendo casos en el que se desea encontrar un valor mínimo o máximo. A continuación, se presenta su definición formal:
  Encontrar un vector:
  $$\vec{X} * = [\, x_1^*, \quad x_2^*, \ldots, x_n^*]$$
  que satisfaga las restricciones:
  $$G_i(\vec{x}) \geq 0, donde\ i = 1, \ldots, m$$
  $$H_j(\vec{x}) = 0, donde\ j = 1, \ldots, p$$
  Y optimice la función vectorial:
  $\boldsymbol{max}\, f(\vec{x})$ maximizar una función objetivo
  o
  $\boldsymbol{min}\, f(\mathrm{x})$ minimizar una función objetivo
  Siendo:
  $\vec{X} = [\, x_1\ ,\ x_2\ , \ldots, x_n\,]$ el vector de variables decisión.
- **Optimización multiobjetivo**: en los problemas de optimización multiobjetivo, se busca la optimización simultánea de más de una función objetivo. Mientras que en la optimización mono-objetivo la tarea es encontrar una solución que optimice el valor de una única función, en este tipo de problemas se pretende encontrar un conjunto de soluciones que sirvan para todas las funciones. Dichas funciones usualmente están en conflicto e individualmente buscan objetivos diferentes. Por ejemplo, si en el cálculo de una ruta deseamos el camino más corto a la vez que el camino menos contaminante, el camino más rápido podría ser coger un helicóptero mientras que el menos contaminante ir a pie. Alcanzar los dos objetivos a la vez es imposible, por lo que el término optimizar significa encontrar una solución que de un valor aceptable a todas las funciones. A continuación, se presenta su definición formal:
  Encontrar un vector:
  $$\vec{X} * = [\, x_1^*, \ x_2^*, \ldots, x_n^*]$$
  que satisfaga las restricciones:
  $$G_i(\vec{x}) \geq 0, donde\ i = 1, \ldots, m$$
  $$H_j(\vec{x}) = 0, donde\ j = 1, \ldots, p$$

Y optimice la función vectorial:

$\boldsymbol{max}\ \mathrm{F}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}) \ldots f_k(\vec{x})]$ maximizar una serie de funciones objetivo

o

$\boldsymbol{min}\ \mathrm{F}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}) \ldots f_k(\vec{x})]$ minimizar una serie de funciones objetivo

Siendo:

$\vec{X} = [x_1, x_2, \ldots, x_n]$ el vector de variables decisión

Como es de imaginar, los problemas de optimización multiobjetivo son más complicados de resolver, más aún cuando existen objetivos contradictorios. Para resolverlos se pueden utilizar dos enfoques. El primero de ellos es emplear alguna técnica para definir lo que es una buena solución, como por ejemplo con la técnica de Pareto, que consiste en que una asignación es óptima si puede mejorar la situación de un objetivo sin hacer que empeore la situación de los otros. Otra estrategia es transformar un problema multiobjetivo a un problema mono-objetivo usando para ello un conjunto de parámetros definidos por el usuario y es el enfoque utilizado en este proyecto.

De forma general, un problema multiobjetivo, tiene como finalidad encontrar una solución por medio de una estrategia de toma de decisiones, es decir, al final del proceso se busca obtener una solución que se ajuste a las necesidades del usuario. Si el usuario define esas necesidades a priori, se puede usar una estrategia basada en preferencias. Consiste en lo siguiente:

En esta estrategia se transforma un problema multiobjetivo a uno mono-objetivo mediante un vector de preferencias, que al multiplicarlo por $F(\vec{x})$ convierte el problema en un problema de optimización mono-objetivo. A continuación, se muestra un ejemplo simple:

Sea P un problema de optimización multiobjetivo formado por los siguientes elementos:

$$\boldsymbol{min}\ f_1(\vec{x})$$

$$\boldsymbol{min}\ f_2(\vec{x})$$

$$G_i(\vec{x}) \geq 0, donde\ i = 1, \ldots, m$$

$$H_j(\vec{x}) = 0, donde\ j = 1, \ldots, p$$

Podemos transformar P en un problema de optimización mono-objetivo añadiendo dos pesos y sumando las funciones:

$$F(\vec{x}) = w_1 * f_1(\vec{x}) + w_2 * f_2(\vec{x}), donde\ w_1 + w_2 = 1$$

3.4 Algoritmos genéticos

Los algoritmos genéticos son métodos adaptativos, generalmente usados en problemas de búsqueda y optimización de parámetros, basados en la reproducción sexual y en el principio de la supervivencia del más apto. Son algoritmos de búsqueda basados en la mecánica de selección y de la genética natural. Combinan la supervivencia del más apto

entre estructuras de secuencias con un intercambio de información estructurado, aunque aleatorizado, para constituir así un algoritmo de búsqueda que tenga algo de las genialidades de las búsquedas humanas [5].

Para alcanzar la solución a un problema se parte de un conjunto inicial de individuos, llamado población, generado de manera aleatoria. Cada uno de estos individuos representa una posible solución al problema. Estos individuos evolucionarán tomando como base la selección natural, y se adaptarán en mayor medida tras el paso de cada generación a la solución requerida [5].

A grandes rasgos, su funcionamiento es el siguiente, se define una población inicial con un número finito de individuos. Cada uno de estos individuos tendrá asociado un *fitness* que cuantifica su validez como solución al problema [5]. De esta población se seleccionarán una serie de padres que se reproducirán y crearán nuevos hijos, que con cierta probabilidad podrán mutar. A continuación, se seleccionarán los individuos que sobrevivirán en la siguiente generación y se repetirá este proceso durante un número determinado de iteraciones. En las próximas secciones se definirán los elementos que forman parte de un algoritmo genético.

3.3.1 Codificación de problemas

Cualquier solución potencial a un problema puede ser representada mediante una serie de parámetros. El conjunto de todos los parámetros se codifica en una cadena de valores denominada cromosoma.

El conjunto de los parámetros representado por un cromosoma particular recibe el nombre de genotipo, y contiene la información necesaria para la construcción del organismo, es decir, la representación real del organismo, denominada fenotipo [5]. Por ejemplo, la información mostrada en un grafo, con sus vértices, aristas y pesos sería una representación del fenotipo. Mientras que su codificación en una lista representaría el genotipo.

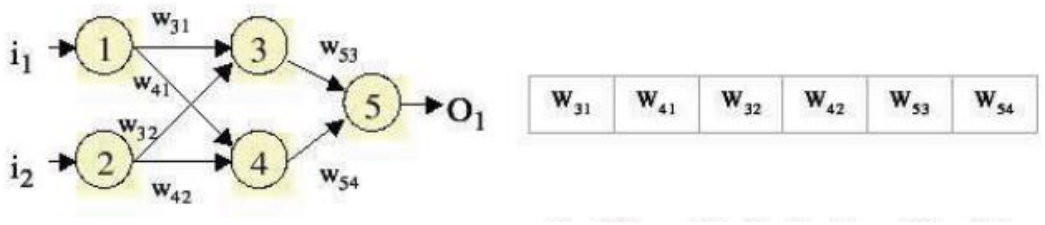

Ilustración 5. Codificación de un grafo [5]

3.2.1 Operadores Genéticos

Para el paso de una generación a la siguiente se aplican una serie de operadores genéticos. Los más empleados son los operadores de selección, cruce, copia y mutación [5]. A continuación, se verán con mayor detalle.

3.2.1.1 Operadores de selección

Los algoritmos de selección serán los encargados de escoger qué individuos van a disponer de oportunidades de reproducirse y cuáles no. Puesto que se trata de imitar lo que ocurre en la naturaleza, se ha de otorgar un número de oportunidades de reproducción a los individuos más aptos. Por lo tanto, la selección de un individuo estará relacionada con su valor de ajuste (*fitness*). No se debe, sin embargo, eliminar por completo las opciones de reproducción de los individuos menos aptos, pues en pocas generaciones la población se volvería homogénea [5]. A continuación, se presentarán los operadores más usados:

- **Selección por ruleta**: a cada uno de los individuos de la población se le asigna una parte proporcional en base a su ajuste y se crea una ruleta, de tal forma que la suma de todos los porcentajes sea la unidad. Los mejores individuos recibirán una porción de la ruleta mayor que la recibida por los peores. Para seleccionar al individuo basta con generar un número aleatorio del intervalo [0...1] y devolver el individuo situado en esa posición de la ruleta [5].
- **Selección por torneo**: la idea principal de este método consiste en realizar la selección en base a comparaciones directas entre individuos. Existen dos versiones de selección, determinística y probabilística. En la versión determinística se selecciona al azar un número p de individuos y de entre estos, se selecciona el más apto. La versión probabilística únicamente se diferencia en el paso de selección del ganador del torneo. En vez de escoger siempre el mejor, se genera un número aleatorio en el intervalo [0..1], y si es mayor que un parámetro f (fijado para todo el proceso evolutivo) se escoge el individuo más apto y en caso contrario el menos apto [5].

Elegir uno u otro método de selección determinará la estrategia de búsqueda del algoritmo Genético. Si se opta por un método con una alta presión de selección, por ejemplo, cuando participan muchos individuos en la selección por torneo y, por lo tanto, peores individuos tienen menos posibilidades de ser escogidos, la búsqueda se centra en encontrar las mejores soluciones actuales. Por el contrario, optando por una presión de selección menor se deja el camino abierto para la reproducción de nuevas regiones del espacio de búsqueda [5].

3.2.1.2 Cruce

Una vez seleccionados los individuos, estos son recombinados para producir la descendencia que se insertará en la siguiente generación. Los diferentes métodos de cruce podrán operar de dos formas diferentes. Si se opta por una estrategia destructiva los descendientes se insertarán en la nueva población, aunque sus padres tengan mejor ajuste. Por el contrario, utilizando una estrategia no destructiva la descendencia pasará a la siguiente generación únicamente si supera el *fitness* de los padres (o de los individuos a reemplazar). La idea principal del cruce se basa en que, si se toman dos individuos correctamente adaptados al medio y se obtiene una descendencia que comparta genes de ambos, existe la posibilidad de que los genes heredados mejoren el *fitness* de los respectivos padres. Si el cruce no agrupa las mejores características en uno de los hijos y la descendencia tiene un peor ajuste que los padres, no significa que se esté dando un paso atrás, ya que en posteriores cruces puede ocurrir que los genes

dispersos sean parte de una buena solución [5]. Para realizar el cruce se seleccionan dos padres de la población actual y se intentan reproducir con cierta probabilidad Pc. Si el cruce tiene éxito, se recombinan sus genes y se genera la descendencia.

Existen multitud de algoritmos de cruce. Sin embargo, se detallarán los analizados para la realización de este trabajo:

- **Cruce de 1 punto**: es la más sencilla de las técnicas de cruce. Una vez seleccionados dos individuos se cortan sus cromosomas por un punto seleccionado aleatoriamente para generar dos segmentos diferenciados en cada uno de ellos: la cabeza y la cola. Se intercambian las colas entre los dos individuos para generar los nuevos descendientes. De esta manera ambos descendientes heredan información genética de los padres [5].

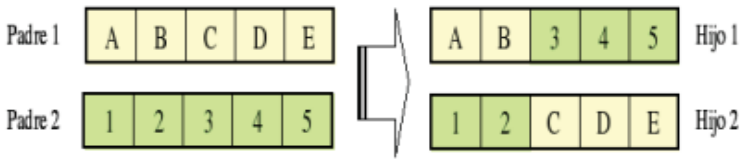

Ilustración 6. Cruce de 1 punto [5]

- **Cruce de 2 puntos**: se trata de una generalización del cruce de 1 punto. En vez de cortar por un único punto los cromosomas de los padres como en el caso anterior, se realizan dos cortes. Deberá tenerse en cuenta que ninguno de estos puntos de corte coincida con el extremo de los cromosomas para garantizar que se originen tres segmentos. Para generar la descendencia se escoge el segmento central de uno de los padres y los segmentos laterales del otro padre [5].

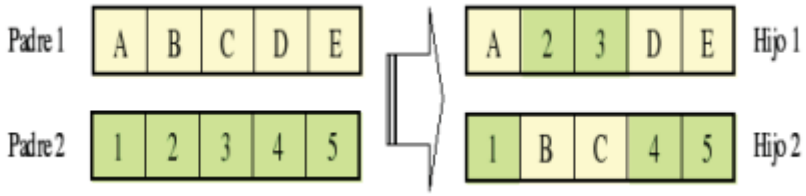

Ilustración 7. Cruce de 2 puntos [5]

- **Cruce uniforme**: El cruce uniforme es una técnica completamente diferente de las vistas hasta el momento. Cada gen de la descendencia tiene las mismas probabilidades de pertenecer a uno u otro padre. Aunque se puede implementar de muy diversas formas, la técnica implica la generación de una máscara de cruce con valores binarios. Si en una de las posiciones de la máscara hay un 1, el gen situado en esa posición en uno de los descendientes se copia del primer padre. Si por el contrario hay un 0 el gen se copia del segundo padre. Para reproducir el segundo descendiente se intercambian los papeles de los padres, o bien se intercambia la interpretación de los unos y de los ceros de la máscara de cruce [5].

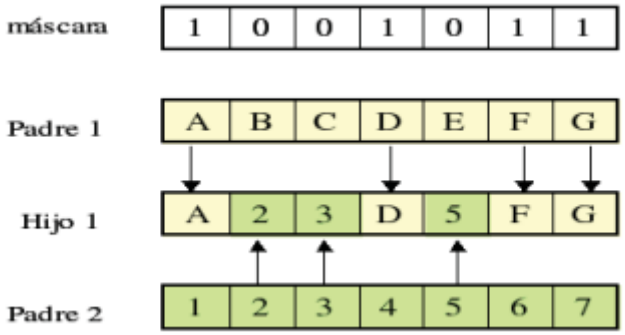

Ilustración 8. Cruce uniforme [5]

3.2.1.3 Mutación

La mutación es un mecanismo que permite realizar cambios en los genes del individuo. Generalmente, se varía un solo gen de los hijos generados, aunque se pueden seleccionar los individuos directamente de la población actual y mutarlos antes de introducirlos en la nueva población, la mutación se suele utilizar de manera conjunta con el operador de cruce. Para ello, una vez ejecutado el cruce, si tiene éxito, uno de los descendientes, o ambos, se muta con cierta probabilidad Pm. Se imita de esta manera el comportamiento que se da en la naturaleza, pues cuando se genera la descendencia siempre se produce algún tipo de error, por lo general sin mayor trascendencia, en el paso de la carga genética de padres a hijos [5].

La probabilidad de mutación es muy baja, generalmente menor al 1%. Esto se debe sobre todo a que los individuos suelen tener un ajuste peor después de ser mutados. Sin embargo, se efectúan mutaciones para garantizar que ningún punto del espacio de búsqueda tenga una probabilidad nula de ser examinado [5].

3.2.1.4 Reemplazo

Cuando se trabaja sobre una única población, sobre la que se hacen selecciones e inserciones, deberá tenerse en cuenta que para insertar un nuevo individuo deberá de eliminarse previamente otro de la población [5]. Existen diferentes métodos de reemplazo:

- **Aleatorio**: se reemplaza cualquier individuo de la población por el hijo generado.
- **Reemplazo de padres**: los hijos generados reemplazan a los padres.
- **Reemplazo de similares**: una vez obtenido el ajuste de la descendencia se reemplaza aleatoriamente aquellos individuos con un *fitness* similar.
- **Reemplazo de los peores**: se seleccionan los peores individuos de la población y se reemplazan por la descendencia.

3.2.1.5 Evaluación

Para el correcto funcionamiento de un Algoritmo Genético se debe de poseer un método que indique si los individuos de la población representan buenas soluciones al problema planteado. De esto se encarga la función de evaluación (o *fitness)*, que establece una medida numérica de la bondad de una solución. Esta medición se utiliza en el mundo de los Algoritmos genéticos para controlar la aplicación de los operadores genéticos. Es decir, permitirá controlar en número de selecciones, cruces, copias y mutaciones llevadas a cabo [5].

La aproximación más común consiste en crear explícitamente una medida de ajuste para cada individuo de la población [5].

3.2.1.6 Algoritmo principal

En la ilustración 9 se puede observar la estructura básica de un Algoritmo genético. En primer lugar, se genera la población inicial, típicamente de manera arbitraria, y se realiza la evaluación de cada individuo de la población. A continuación, a partir de los resultados obtenidos por la primera generación, se seleccionan los padres de la población que serán usados para producir a la descendencia que acabará formando parte de la población en la siguiente generación, mediante los operadores de selección, cruce y mutación. Con la nueva población generada, se repetirán los pasos anteriores una y otra vez hasta que se cumpla algún criterio de finalización.

```
Inicializar población actual aleatoriamente

MIENTRAS no se cumpla el criterio de terminación
       crear población temporal vacía

  MIENTRAS  población temporal no llena
        seleccionar padres
        cruzar padres con probabilidad Pc
        SI se ha producido el cruce
            mutar uno de los descendientes con probabilidad Pm
            evaluar descendientes
            añadir descendientes a la población temporal
        SINO
            añadir padres a la población temporal
        FIN SI
  FIN MIENTRAS

  aumentar contador generaciones
  establecer como nueva población actual la población temporal

FIN MIENTRAS
```

Ilustración 9. Pseudocódigo de un AG [5, p. 6]

4. Diseño de la solución

Esta sección será empleada para definir y resolver el problema planteado en este proyecto. Aunque en la introducción se ha definido de manera breve que se quiere obtener la ruta multimodal más rápida o menos contaminante entre dos puntos de la ciudad de Valencia y su periferia, usando para ello un conjunto de transportes públicos, creemos necesario añadir en este capítulo un ejemplo simple para que quede expresado de forma clara y explícita todos los elementos intervinientes en el cálculo de la ruta, al mismo tiempo que motivamos el uso del algoritmo genético. A continuación, mostraremos un estudio sobre las aplicaciones ya existentes en el mercado y se mostrará la solución que se implementará en este proyecto.

4.1 Un problema simple

Imaginemos que estamos cerca de la Universidad Politécnica de Valencia y en los alrededores hay un conjunto de estaciones. Concretamente, existen tres estaciones de tranvía, dos de autobús y dos de Valenbisi. Asimismo, existe un usuario que quiere ir desde el punto verde hasta el punto naranja (ver ilustración 10), necesitando obtener una ruta en base a sus preferencias.

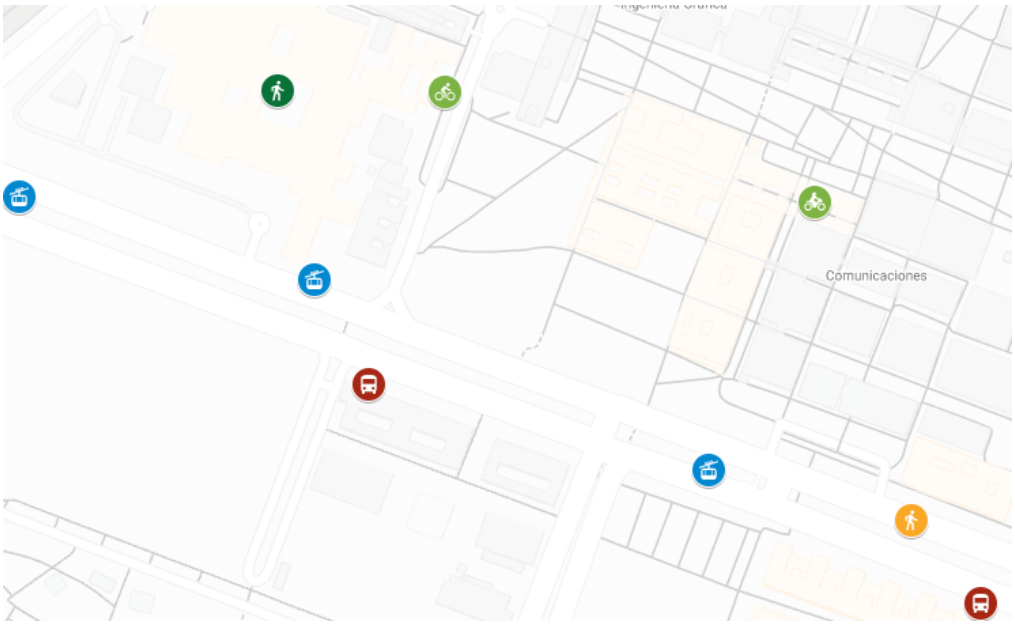

Ilustración 10. Mapa de estaciones

Dicha información puede ser representada visualmente mediante un grafo no dirigido G = (V, A), donde los vértices representan a las estaciones y las aristas las conexiones entre estas. En la ilustración 11 se muestra su representación. Por simplicidad solo se han pintado las aristas entre estaciones del mismo tipo, pero implícitamente, el grafo es completo, ya que existe un camino entre cada par de vértices, que corresponde a ir andando desde una estación hacia otra.

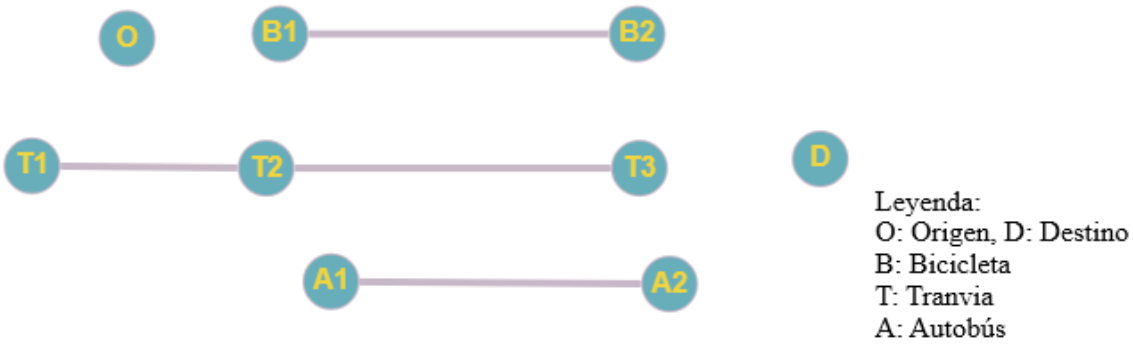

Ilustración 11. Representación de las estaciones

A su vez, cada vértice, dispondrá de una información dependiendo de su tipo de estación y de la ruta a obtener. Si el usuario busca la ruta más rápida, en los vértices de tipo autobús o tranvía será necesario tener un horario de paso para saber a qué hora llega el transporte. Asimismo, cada arista deberá de tener un valor que indique el tiempo de desplazamiento entre dos estaciones. Dicho tiempo de desplazamiento dependerá del par de estaciones conectadas por la arista, si son del mismo tipo es posible que haya un transporte que conecte ambas estaciones, sin embargo, si el tiempo de espera al transporte es mayor que el tiempo yendo a pie, se debería de considerar la segunda opción como parte de la solución. En caso de necesitar la ruta menos contaminante haría falta una métrica bien definida para calcular los gramos de CO_2 que se emiten al usar un determinado tipo de transporte.

En resumen, para resolver el problema será necesario disponer de los siguientes elementos:

- Hora de salida del usuario.
- Ubicación de origen y destino.
- Localización exacta de todas las estaciones.
- Una tabla de horarios por cada estación que no sea de tipo bicicleta.
- Número de kilómetros y tiempo de desplazamiento más corto entre dos estaciones.

De esta forma, teniendo todos los elementos bien definidos, el problema se reduce en obtener el camino óptimo del grafo entre los vértices O y D en base a los objetivos definidos. Así pues, podemos representar posibles soluciones factibles de este problema mediante tuplas. A continuación, se muestran tres ejemplos:

- **(O, D)**: Ir desde el origen hacia el destino directamente andando.
- **(O, B1, B2, D)**: Ir andando desde el origen hasta B1, coger una bicicleta en B1 hasta B2 e ir andando desde B2 al destino.
- **(O, T1, T2, A1, A2, D):** Ir andando desde el origen hasta T1, coger el tranvía hasta T2, ir andando hasta A1, coger el bus hasta A2 e ir andando hasta el destino.

Aunque existen muchos caminos, todos tienen elementos en común. La ruta empieza en el origen y termina en el destino. Además, cada estación solo puede aparecer una

única vez en la solución, ya que, de no ser así, se formarían bucles en el camino. De esta forma, por ejemplo, la tupla (O, B1, B2, B1, B2, D) no sería una solución válida.

De este modo, ya solamente nos resta diseñar un algoritmo capaz de obtener la ruta óptima. Si pensáramos en un diseño simple, bastaría con realizar una búsqueda de fuerza bruta que pruebe todas las posibles combinaciones y recorra todos los caminos hasta dar con la situación óptima. Esto sería posible si tuviéramos un número reducido de estaciones, pero conforme vaya aumentando el tamaño del problema, es probable que, en el peor de los casos, el tiempo de resolución del algoritmo aumente considerablemente con el número de estaciones, para ser exactos, en un factor de O((n-1)!), el factorial del número de estaciones, por lo que incluso con tan solo cientos de estaciones se tardarían bastantes años en encontrar la solución óptima.

Este tiempo de cómputo es fácilmente demostrable. Supongamos que disponemos de 5 estaciones, las estaciones 1,2,3,4,5 y queremos obtener la ruta más rápida para ir desde la estación 1 hasta la 5. Un algoritmo de fuerza bruta empezaría el recorrido desde la estación 1, y una vez elegida la primera, barajaría cuatro posibilidades más, las tuplas (1, 2), (1, 3), (1, 4), (1, 5). En este punto dispondríamos ya de una solución, la tupla (1, 5), pero no podemos saber si esta solución es la óptima hasta no haber analizado las otras tres tuplas. En la siguiente iteración ¿Cuántas posibilidades habría?, Ya hemos elegido dos estaciones, por lo que, para cada tupla, se tendría que probar un camino con las tres estaciones restantes, por ejemplo, la tupla (1,2) generaría las tuplas (1, 2, 3), (1, 2, 4) y (1, 2, 5). Si siguiéramos de esta forma, hasta que solo quedara una estación por visitar, podríamos comprobar que se han generado 4x3x2x1 rutas posibles, esto es el factorial de (5-1), que generalizando a n estaciones es (n-1)!

De esta manera, si en lugar de tener cinco estaciones tuviéramos 100, el total de rutas posibles a analizar sería de más de tres millones seiscientos mil. Si tuviéramos un ordenador capaz de procesar una ruta por segundo, tardaríamos más de un mes en encontrar la solución a este problema. ¿Entonces, como podemos resolver este problema con un tamaño de estaciones considerable? Una posible solución es con el uso de algoritmos heurísticos y de aproximación, donde se incluyen los algoritmos genéticos. Con un buen diseño se puede llegar a una buena solución rápidamente, pudiendo encontrar soluciones para problemas extremadamente grandes (miles de estaciones) en un tiempo razonable, con una alta posibilidad de que no se alejen mucho de la solución óptima.

4.2 Estado del arte

En esta sección del proyecto se mostrarán las tecnologías existentes en el mercado que se asemejan al propósito de este proyecto. En particular, se hablará acerca de las aplicaciones similares que existen en el mercado, con sus respectivas características y se justificarán las mejoras que aporta este proyecto a las tecnologías ya existentes.

4.2.1 Aplicaciones similares

Existen una multitud de aplicaciones destinadas a la planificación de rutas que permiten ir desde un sitio hacia otro. A continuación, se presentarán dos de ellas, Google *Maps* y Emt Valencia.

4.2.1.1 Google Maps

Google *Maps* es una aplicación de mapas en la web que ofrece un planificador de rutas para poder desplazarse en coche, motocicleta, transporte público, a pie, bicicleta y avión. La compañía ofrece un servicio multiplataforma, permitiendo su uso en una gran cantidad de dispositivos como son móviles *Android*, *iPhone's* y navegadores web. Para usarlo es tan sencillo como seleccionar un origen, un destino y el modo de transporte con el que desplazarse. Una vez introducidos todos los parámetros se le mostrará al usuario la mejor ruta. Asimismo, si existen varias rutas, la ruta más rápida aparecerá en azul y el resto en gris. También se le mostrará al usuario unas indicaciones en formato de itinerario para desplazarse por la ruta [6].

Ofrece una multitud de servicios, de las cuales destacaremos las relacionadas con este trabajo. En primer lugar, además de mostrar la ruta más rápida también se le muestra al navegante una ruta alternativa que ahorra el consumo de combustible gastad. Por lo tanto, permite la reducción de emisiones de CO_2 emitidas a la atmósfera. Para ello tiene en cuenta diferentes factores que afectan al consumo de combustible y a las emisiones de CO_2, como son, el consumo medio de combustibles de los vehículos de una región o los patrones de parada y reanudación de la marcha en el tráfico [6]. Asimismo, cuándo el usuario selecciona la opción de transporte público puede optar por elegir una hora de salida y los modos de transporte con los que desea viajar (autobús, tren, metro y tranvía). Finalmente, también dispone de un modo de transporte multimodal cuando el usuario selecciona la opción de transporte público, permitiendo combinar los diferentes servicios de transporte con opciones de ciclismo y servicios de transporte privados [7].

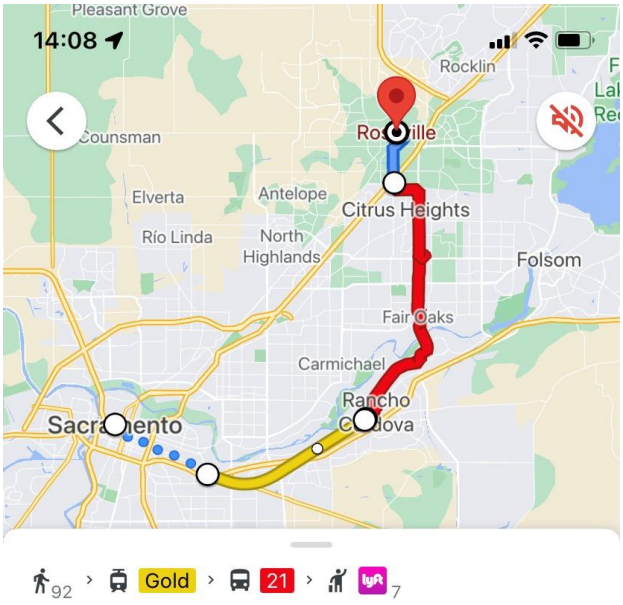

Ilustración 12. Ejemplo de ruta multimodal en Google Maps

Su principal inconveniente reside en que algunas indicaciones de Google Maps están en desarrollo y su disponibilidad está limitada [8]. Aunque se prevé su despliegue en 2022, en Europa no es posible obtener la ruta ecológica mencionada en el párrafo anterior y, por otro lado, tampoco se pueden obtener rutas multimodales que incluyan los servicios principales de alquiler de bicicletas públicas de la ciudad de Valencia [7].

4.2.1.2 EMT Valencia

La empresa Municipal de Transportes de Valencia, también conocida como EMT Valencia es una entidad que proporciona servicios de transporte público en la ciudad de Valencia y sus alrededores. Entre estos servicios, se incluye un planificador de rutas multiplataforma disponible para móviles *Android*, *iPhone* y cualquier navegador web.

Su operatividad es parecida a la de *Google Maps,* con la diferencia de que solo ofrece rutas para la ciudad de Valencia y las poblaciones de Alboraya, Alfara del Patriarca, Burjassot, Moncada, Tavernes Blanques, Sueca y Vinalesa. Asimismo, los medios de transporte ofrecidos para la obtención de la ruta son su flota de autobuses, los transportes de Metrovalencia, y el servicio de bicicletas Valenbisi.

Para obtener una ruta, el usuario pulsa sobre los medios de transporte con los que quiere desplazarse y selecciona un origen y un destino perteneciente a uno de los municipios mencionados anteriormente. Una vez seleccionados, el motor de búsqueda ofrecerá varias soluciones de desplazamiento, así como un itinerario a seguir, que incluye localizaciones, horarios de transporte y el ahorro de CO_2 conseguido al utilizar esa ruta [9].

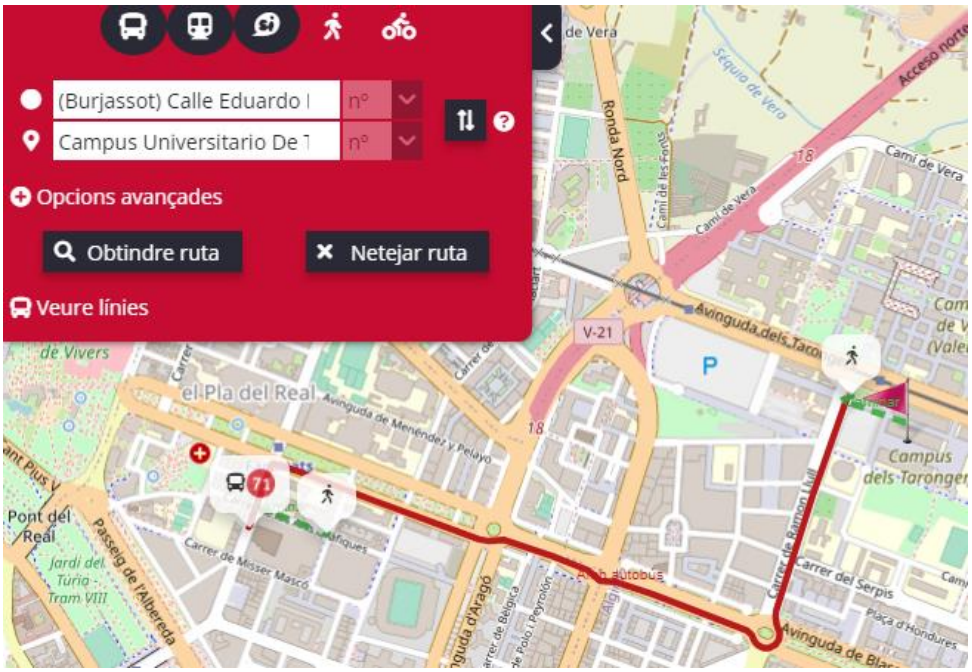

Ilustración 13. Ejemplo de ruta en EMT Valencia

Sus principales desventajas son que no ofrece rutas que incluyan varios modos transporte. Si se eligen varios de estos modos, al ciudadano se le muestra una lista de rutas con cada transporte de manera individual. En consecuencia, se pierden itinerarios

que puedan ser más cortos que los presentados por la aplicación. Asimismo, en algunas ocasiones no se muestra la ruta más corta para la combinación de transportes seleccionados y tampoco se le indica al usuario la manera en la que se ha calculado el ahorro de CO_2 con la ruta elegida.

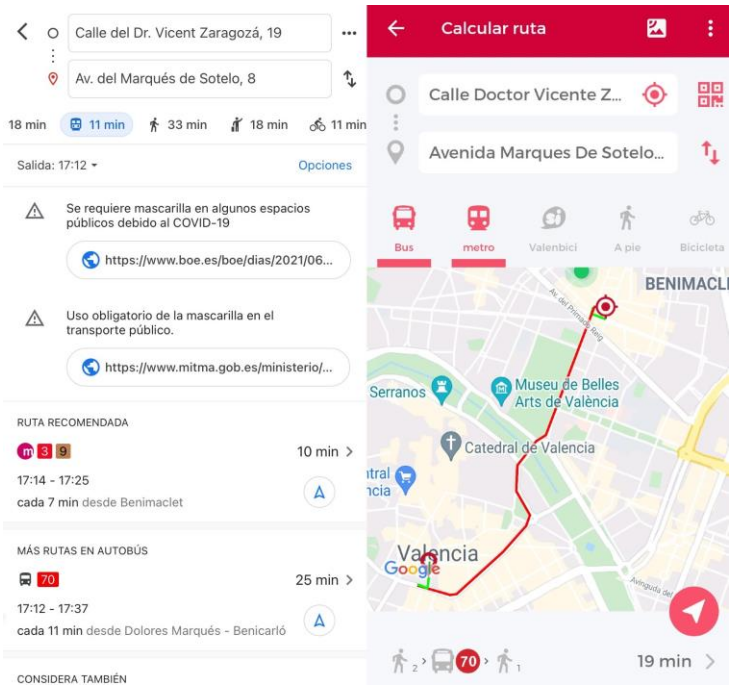

Ilustración 14. Comparativa de rutas entre las dos aplicaciones (11 minutos frente a 19 minutos)

4.2.2 Mejora propuesta

En la sección anterior se ha recogido información acerca de las aplicaciones que tienen una funcionalidad parecida a la que se desea conseguir en este proyecto. En este apartado se pretende justificar las mejoras que aportará el desarrollo del algoritmo genético. En primer lugar, se quiere permitir la selección de varios modos de transporte con los que desplazarse, esto es, el usuario podrá seleccionar aquellos transportes con los que ir de un punto a otro. Asimismo, también se realizarán los desarrollos necesarios para encontrar soluciones multimodales con los servicios de transporte públicos ubicados en la ciudad de Valencia. De esta forma se unificarán todos los servicios de las dos aplicaciones en una sola. Finalmente, se añadirá una nueva característica, el poder seleccionar el tipo de ruta que mejor se adapte a las necesidades del ciudadano. Para ello, el usuario podrá elegir si desea encontrar la ruta más rápida, sin tener en cuenta aspectos ecológicos, la ruta más ecológica o, por el contrario, una ruta que tenga en cuenta ambos factores. Esto permitirá obtener rutas parecidas a la solución más rápida, pero teniendo en cuenta las emisiones de CO_2 producidas. A continuación, se muestra una tabla que resume lo expuesto en el párrafo:

	Google Maps	EMT Valencia	Aplicación propuesta
Permite la selección de varios modos de transporte con los que desplazarse	Sí	Sí	Sí
Encuentra soluciones multimodales	Sí	No	Sí
Tiene en cuenta factores ecológicos	No	Sí	Sí
Tiene en cuenta todos los medios de transporte públicos de la ciudad.	No	Sí	Sí
Permite seleccionar el tipo de ruta que más se adapta al usuario (ecológica o rápida)	No	No	Sí

Tabla 1. Comparación de las diferentes aplicaciones

4.3 Solución propuesta

En esta sección se definirá una solución al problema propuesto. Para ello, se empezará por mostrar los medios de transportes usados en este proyecto y de qué manera obtener toda su información (horarios, localizaciones de las estaciones, número de bicicletas en una estación, etc.). Después mostraremos el diseño de un algoritmo genético, basado en los trabajos propuestos por [10], [11] y veremos un ejemplo del algoritmo genético diseñado. Finalmente, se mostrará una solución creada por el alumno para reducir el número de estaciones a visitar durante el cálculo de una ruta, lo que permitirá reducir el espacio de búsqueda del algoritmo genético y, en consecuencia, hacer que sea más rápido.

4.3.1 Obtención del transporte público

Valencia dispone de una amplia red de transportes públicos que pone a disposición de sus ciudadanos. Los siguientes modos serán utilizados para el proyecto:

- **EMT Valencia:** La Empresa Municipal de Transportes de Valencia es una entidad que proporciona un servicio de transporte público por medio de autobuses en la ciudad de Valencia y su periferia (Alboraya, Alfara del Patriarca, Burjassot, Moncada, Tavernes Blanques, Sueca y Vinalesa). En la tabla siguiente se recogen sus características principales:

N.º de líneas	56
N.º de autobuses	481
N.º de paradas	900

Tabla 2. Flota de EMT Valencia

- **Metrovalencia**: Metrovalencia es un servicio de transporte público ofrecido por la empresa pública FGV (Ferrocarriles de la Generalitat Valenciana) que opera en Valencia y su área metropolitana por medio de tranvías y metros. En la tabla siguiente se recogen sus características principales:

N.º de líneas	9
N.º de tranvías/trenes	106
N.º de paradas	214

Tabla 3. Flota de Metrovalencia

- **Valenbisi**: Valenbisi es el servicio de alquiler de bicicletas públicas implantado en la ciudad de Valencia desde 2010. Su funcionamiento es muy simple, el usuario se dirige a cualquier estación con bicicletas disponibles y una tarjeta de abonado. A continuación, introduce su pin y retira una bicicleta que podrá usar durante un tiempo máximo de 30 minutos. Una vez haya acabado, es suficiente con dejar directamente la bicicleta en una borneta de otra estación. Sus características son las siguientes:

N.º de estaciones	275
N.º de bicicletas	2750

Tabla 4. Comparación de las diferentes aplicaciones

La información de estos modos de transporte se elegirá de distintos sitios. Para empezar, se obtendrá toda la información necesaria de EMT Valencia y Metrovalencia desde una especificación de datos llamada Especificación de alimentación de tránsito general (GTFS) que pone a disposición del usuario, un conjunto de ficheros que definen un formato común para los horarios de transporte público y su información geográfica relacionada. Dicho ficheros permiten que las empresas de transporte publiquen sus datos de forma pública. Son usados por Google y una multitud de empresas, por lo que su uso es bastante extendido y fiable. Existen de dos tipos, estáticos y en tiempo real. El primero de ellos contiene datos para unas determinadas fechas. Son actualizados cada cierto tiempo por las empresas de transporte público, por lo que es responsabilidad del programador obtener su última versión. El segundo de ellos ofrece un componente en tiempo real que contiene predicciones de llegada o avisos de servicio entre otras funcionalidades. En la ilustración 15, se muestra un ejemplo del conjunto de estos archivos.

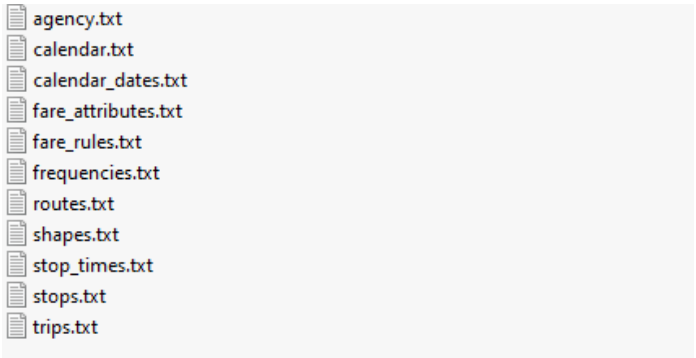

Ilustración 15. Ejemplo GTFS

Para la realización de este proyecto hemos hecho uso únicamente de la información estática, proveída por el Ayuntamiento de Valencia. Concretamente, para obtener los

elementos que participarán en el cálculo de la ruta se han utilizado los siguientes archivos:

- **calendar.txt:** Indica las fechas de disponibilidad del servicio. Cada línea del archivo contiene un identificador único, las fechas y los días en los que está disponible el servicio. Por ejemplo, en la ilustración 16 se define que la primera línea con el servicio 000 estará disponible solo entre semana, mientras que la segunda, únicamente el sábado. Ambos servicios funcionan desde el 07/01/2020 y son válidos hasta el 31/12/2022.

```
service_id,monday,tuesday,wednesday,thursday,friday,saturday,sunday,start_date,end_date
000,1,1,1,1,1,0,0,20200107,20221231
100,0,0,0,0,0,1,0,20200107,20221231
```

Ilustración 16. Ejemplo de fichero calendar.txt

- **trips.txt:** Indica las rutas existentes en el servicio, entendiendo el concepto de ruta como una secuencia de dos o más paradas por la que transcurre un vehículo. Cada ruta dispone de un identificador de ruta único, un identificador de servicio y un identificador de viaje. Por ejemplo, en la ilustración 17 se puede observar que la primera línea indica que la ruta TO-MS con identificador 2202850 está disponible para las fechas especificadas en el servicio 0, mientras, que la segunda, señala otra ruta para el servicio 000. De esta forma, la ruta 2202850 solo estará disponible los sábados, mientras que la segunda se realizará entre semana.

```
route_id,service_id,trip_id
TO-MS,100,2202850,Marítim - Serrería
MR-DL,000,2202849,Dr. Lluch
```

Ilustración 17. Ejemplo de fichero trips.txt

- **stop_times.txt**: Proporciona información sobre un viaje y las paradas por las que transcurre. Cada viaje dispone de un identificador de ruta y las horas, en la que un vehículo llega a una parada y sale de ella, además de su secuencia. Como se puede observar en la figura 18, la tercera línea indica que el viaje 2202850 empieza en la parada con identificador 131 a las 07:19 de la mañana, con salida inmediata. A continuación, llega a la parada 130 dos minutos después y la parada 129 un minuto después.

```
trip_id,arrival_time,departure_time,stop_id,stop_sequence
2202850,07:19:00,07:19:00,129,3,0,0
2202850,07:18:00,07:18:00,130,2,0,0
2202850,07:16:00,07:16:00,131,1,0,0
```

Ilustración 18. Ejemplo de fichero stop_times.txt

- **stops.txt:** Define las paradas en las que los vehículos recogen o dejan a los pasajeros. Para ello, se dispone de un identificador de parada, su nombre y su localización mediante coordenadas. En la ilustración 19 se muestra un ejemplo.

```
stop_id,stop_name,stop_lat,stop_lon,zone_id
129,Orriols,39.4931488037,-0.3676636219
130,Estadi del Llevant,39.4949188232,-0.3655419946
```

Ilustración 19. Ejemplo de fichero stops.txt

De esta manera, mediante estos ficheros disponemos de toda la información necesaria para calcular las rutas multimodales de un problema. Con los ficheros calendar.txt y trips.txt se pueden obtener los viajes disponibles para una fecha y hora concreta. El fichero stop_times.txt proporciona toda la información de estos viajes de manera detallada, por lo que es posible obtener el tiempo de desplazamiento entre dos paradas conectadas mediante la secuencia. Finalmente, con el archivo stops.txt sabemos dónde se localizan las paradas, por lo que es posible calcular la distancia entre ellas para obtener el factor de CO_2 por kilómetro recorrido.

Por otro lado, la información de las bicicletas Valenbisi se recogerá desde unos ficheros ofrecidos por la empresa JCDecaux. Dicha empresa dispone de un servicio web para poder obtener información dinámica sobre las estaciones de servicio y el número de bicicletas disponibles. Para ello solo hay que registrarse en su página https://developer.jcdecaux.com/ y obtener una clave API. A partir de ese momento podremos acceder a unos ficheros en formato JSON con toda la información. Concretamente, por cada entrada en el fichero dispondremos de estos campos:

- **Number**: corresponde a un número de estación, identificándola de manera única.
- **Address**: indica el nombre de la estación y su dirección exacta.
- **Position**: indica las coordenadas de la estación en formato latitud y longitud.
- **Available_bike_stands**: indica el número de huecos libres para poder dejar las bicicletas en la estación.
- **Available_bikes**: indica el número de bicicletas disponibles en la estación.
- **Status**: Muestra si la estación está operativa (*OPEN*, *CLOSE*).
- **Last_update**: señala la hora en la que se hizo la última actualización del sistema. De esta forma, es posible saber si ha habido algún cambio en el flujo de bicicletas de la estación.

Para organizar la información de las estaciones se utilizará una base de datos relacional, utilizando para ello un diagrama de entidad-relación para el diseño de todas las tablas de la base de datos. En la ilustración 20 se muestra el diagrama que se ha empleado en el diseño de la aplicación. Concretamente, se indexarán, todas las estaciones de los ficheros en una misma tabla con un identificador único y con su tipo de estación (bus, tren, bicicleta). Asimismo, cada estación de tipo bus o tren dispondrá de sus viajes, para poder recuperar de manera cómoda los horarios que nos interesen.

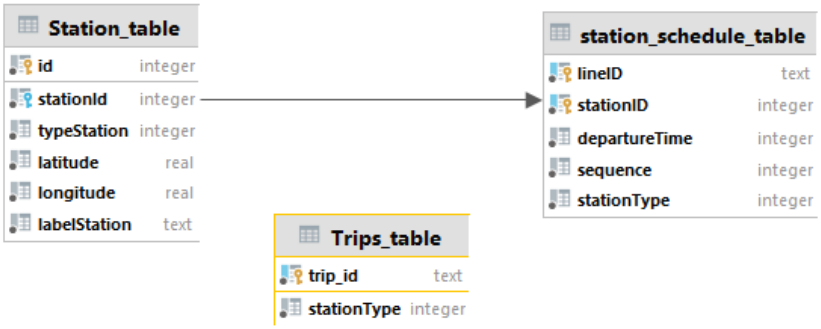

Ilustración 20. Diagrama entidad relación

4.3.2 Características del Algoritmo genético

Una vez que se dispone de las estaciones es posible empezar a trabajar con el algoritmo genético. Sus características se tomaron como referencia de los trabajos presentados por [10], [11], los cuales demostraron resultados óptimos en sus investigaciones para problemas muy similares al que se pretende resolver. A continuación, presentaremos el algoritmo propuesto con los siguientes parámetros, a la vez que clarificaremos su uso mediante un ejemplo simple:

- Tamaño de la población: 4
- Probabilidad de cruce (pc): 0.80
- Probabilidad de mutación (pm): 0.05
- W1 = 0.8
- W2 = 0.2
- Número de iteraciones: 10

Como se ha comentado en la sección 3.4 de este documento, el primer paso para el diseño de un algoritmo genético es codificar el fenotipo en un genotipo. En este caso, el fenotipo son las estaciones del sistema y las conexiones entre estas. Dichas estaciones serán codificadas mediante un identificador único en forma de listas. Asimismo, cada posición de la lista representará un orden de visita en la ruta, por ejemplo, en una lista de 4 componentes, la posición 0 será el inicio del viaje, la posición 1 el primer transporte elegido y la posición 2 el segundo transporte conectado por el primero. Para ilustrar un ejemplo, utilizaremos las mismas estaciones vistas en la figura 11 del problema simple (O, B1, B2, T1, T2, T3, A1, A2, D).

A continuación, se ha de generar la inicialización de la población, que consiste en la creación de los individuos. Cada individuo de la población representa un posible camino desde el vértice O hasta el vértice D, siendo la longitud de estos, variable entre 2 y N, dependiendo de la trayectoria, donde N es el total de estaciones del sistema. Esto es porque nunca se necesita más que N estaciones para formar una ruta. Asimismo, la creación de cada individuo de la población se hace de forma aleatoria, es decir, un individuo se va construyendo, empezando por el inicio. A continuación, se conecta de forma aleatoria con otro nodo del sistema (en este caso, todas las estaciones incluyendo

el destino, ya que es posible ir andando a cualquier punto) y se repite el proceso hasta que el nodo elegido sea D. En la tabla 5 se muestra un ejemplo:

	Población inicial
1	[O, D]
2	[O, B1, A2, A1, T2, D]
3	[O, T1, T1, T3, T2, A1, D]
4	[O, T3, B1, A2, T3, T2, B2, B1, D]

Tabla 5. Generación de la población inicial

El segundo paso, es iniciar una función de reparación para cada individuo. El objetivo de esta función, tal como lo expresa [10] es la de arreglar los individuos, de manera que los caminos inválidos sean factibles. Concretamente, se implementa para eliminar las estaciones que forman un bucle en un camino. La ilustración 21 muestra un ejemplo de la función de reparación.

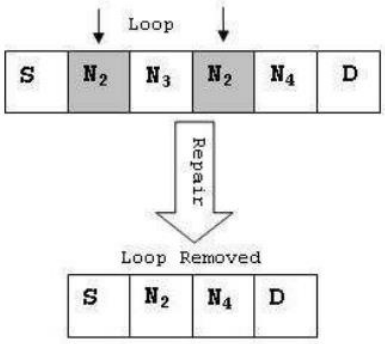

Ilustración 21. Función de reparación

Como podemos observar, los individuos 3 y 4 presentan bucles en sus genes. Concretamente, el individuo 3 presenta un solo bucle (T1), mientras que el individuo 4 presenta dos (T3 y B1). Aplicando la función de reparación se obtienen las siguientes correcciones:

	Población tras la función de reparación
1	[O, D]
2	[O, B1, A2, A1, T2, D]
3	[O, T1, T3, T2, A1, D]
4	[O, T3, T2, B2, B1, D]

Tabla 6. Población inicial tras haber aplicado la función de reparación.

El siguiente paso es efectuar la evaluación de aptitud (*fitness)* que tiene por objetivo calcular el coste total de un individuo, en base a los dos objetivos propuestos. El coste de un individuo se calcula sumando todas las conexiones entre las estaciones codificadas del individuo. La siguiente ecuación muestra el *fitness* del individuo x_i.

$$x_i = \sum_{k=0}^{n-1} min\,[w1 * tiempo(k, k+1) + w2 * contaminacion(k, k+1)]$$

Donde x_i representa al individuo, n el número total de estaciones del individuo y k el elemento k-ésimo del individuo, Asimismo, $tiempo(k, k+1)$ representa el tiempo de desplazamiento entre dos estaciones, $contaminacion(k, k$ +1) la contaminación producida y w1, w2 los pesos de importancia que se le da a la ruta más corta y a la ruta menos contaminante.

Asimismo, $min\ [w1 * tiempo(k, k+1) + w2 * contaminacion(k, k$ +1)] representa una función que calcula cuál es la mejor ruta en base a los dos objetivos entre k y k+1. Funciona del siguiente modo:

- En caso de que las estaciones sean del mismo tipo, si son estaciones de autobús o tren, se comprueba si existe una conexión entre las dos estaciones. Por ejemplo, en el individuo 3 de la tabla 6, T3 Y T2 están conectados directamente por una arista, por lo que existe una conexión inmediata, pero T1 Y T3 también lo están, ya que, aunque no estén conectados directamente, pertenecen a la misma línea y se puede llegar a ellos mediante un tren. De esta forma, si hay una conexión se calculan dos rutas, la que corresponde a ir de k hasta k +1 en transporte y la ruta andando entre esos dos nodos. En la primera ruta, se calcula $tiempo(k, k+1)$ como la suma del tiempo de espera en k hasta que llegue el transporte, más el tiempo de trayecto hasta k+1, después se obtienen los kilómetros de la ruta, y con ello, se calcula $contaminacion(k, k+1)$. En la segunda ruta se calcula $tiempo(k, k+1)$ como el tiempo de desplazamiento al ir andando desde k hasta k+1. Como en este caso la contaminación producida es de 0 (véase sección 3.2 de este documento) solo se tiene en cuenta el tiempo de desplazamiento. Finalmente, se multiplica las dos rutas por los factores w1 y w2 y se escoge la ruta con el valor más bajo.
- Si las estaciones son del mismo tipo, pero de tipo bicicleta, entonces $tiempo(k, k+1)$ se calcula como el tiempo que se tarda en ir en bicicleta desde k hasta k + 1. En este caso, $contaminacion(k, k+1)$ También vale 0, por lo que no se tiene en cuenta el factor contaminante.
- En caso de que las rutas sean de diferente tipo, la ruta se calcula como el tiempo de ir andando desde k hasta k + 1.

De esta forma, es posible obtener hasta tres tipos de rutas. Si w1 o w2 valen 0, la evaluación de la aptitud solo tiene en cuenta un tipo de ruta, la ruta más rápida o la menos contaminante. En caso contrario, se obtiene una ruta mixta en base a las preferencias del usuario.

Para ilustrar este ejemplo, imaginemos que disponemos de una matriz de adyacencia con el valor de aptitud calculado para cada conexión entre dos nodos:

-	O	B1	B2	T1	T2	T3	A1	A2	D
O	-	2	6.5	2	2.5	7	4	8	10
B1	2	-	4	3.5	2	5	3	6	7.5
B2	6.5	4	-	7	4.5	1.5	4.5	3	2.5
T1	2	3.5	7	-	3	7	4	7.5	9
T2	2.5	2	4.5	3	-	4.5	1.5	4.5	5
T3	7	5	1.5	7	4.5	-	4	1.5	2
A1	4	3	4.5	4	1.5	4	-	4	5.5
A2	8	6	3	7.5	4.5	1.5	4	-	3

D	10	7.5	2.5	9	5	2	5.5	3	-

Tabla 7. Matriz de adyacencia de las conexiones

Aplicando la matriz a los individuos, obtenemos el siguiente valor de aptitud para cada uno:

Población tras la evaluación de aptitud		**Valor de aptitud**
1	[O, D]	15
2	[O, B1, A2, A1, T2, D]	2 + 6 + 4 + 1.5 + 5 = 18.5
3	[O, T1, T3, T2, A1, D]	1 + 7 + 4.5 + 1.5 + 5.5 = 19.5
4	[O, T3, T2, B2, B1, D]	7 + 1.5 + 4.5 + 4 + 7.5 = 24.5

Tabla 8. Población inicial tras fitness

Una vez calculado el valor de aptitud de cada individuo, se ordenan de menor a mayor.

Población tras la evaluación de aptitud		**Valor de aptitud**
1	[O, D]	15
2	[O, B1, A2, A1, T2, D]	18.5
3	[O, T1, T3, T2, A1, D]	19.5
4	[O, T3, T2, B2, B1, D]	24.5

Tabla 9. Población inicial ordenada

A partir de este punto ya disponemos de la población inicial. El siguiente paso es realizar la operación de cruce. Para ello, debemos seleccionar dos individuos como padres para que se efectúe. En la selección de los padres, [11] menciona que el mejor operador genético para obtener un oportuno nivel de eficiencia del algoritmo es el de selección por torneo. Por ello, se empleó este esquema de cruce para la selección del primer padre. Para ello, se escogen al azar dos individuos de la población (comprobando que sean diferentes). Supongamos que son los individuos 3 y 4. De entre ellos, el que demuestra mejor valor de aptitud es el individuo 3, por lo que es escogido como primer padre de la operación de cruce. El segundo padre se selecciona al azar, de esta forma, se libera un poco de presión en la selección, asegurando de que los individuos con un peor valor de aptitud tengan más posibilidades de reproducirse. En este caso, supongamos que elegimos al individuo 2.

Una vez seleccionados los padres, hay que elegir si se produce o no el cruce, para ello se genera un número al azar en el rango [0, 0.80] y, si el número generado es menor que la tasa de cruce Pc, se procede a la ejecución de cruce. En caso contrario, se repite el proceso de selección de padres. Supongamos que si ocurre en este caso.

El siguiente paso es elegir el método de cruce. [10] proponen un cruce de un punto con una pequeña modificación, que consiste en que los padres deben de tener al menos una estación en común. A continuación, elegir una estación aleatoria de entre todas las comunes, y utilizar esta como punto de cruce. Una vez decidido el punto de cruce, se aplica el método usual de la operación de cruce de un punto y se eliminan posibles bucles con la función de reparación. Un ejemplo de esta operación se muestra en la figura 22.

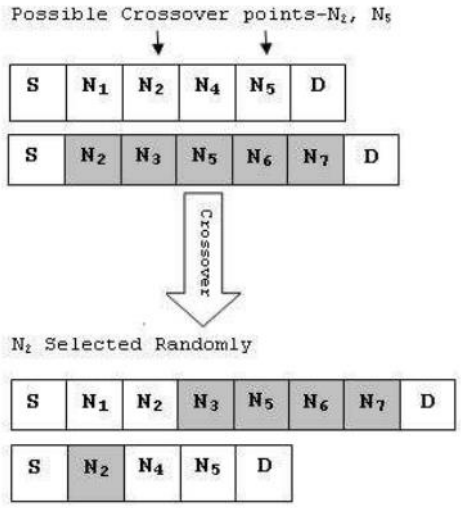

Ilustración 22. Ejemplo de operación de cruce

Aunque al principio se empleó esta operación, existían problemas con uso. El principal es que se conseguía una población homogénea a partir de la tercera o cuarta generación, por lo que no se generaban nuevas soluciones. Esto era debido a que, con muchas estaciones, no se encontraban muchos individuos con estaciones en común, por lo que en el proceso de reproducción siempre intervenían las mismas soluciones. Es por ello, por lo que finalmente, se optó por usar la técnica tradicional del punto de un cruce, que funciona de la siguiente manera: en primer lugar, se retiran del individuo las estaciones de inicio y destino. A continuación, se elige un punto de corte seleccionado aleatoriamente en los dos individuos (asegurándonos de que no sea el primer o último elemento de la lista) llamados cabeza y cola y se intercambian las cabezas y las colas de los padres, para generar dos nuevos descendientes. Finalmente, se vuelve a introducir la estación de origen y destino y se aplica la función de reparación para eliminar posibles bucles. En la tabla 9 se muestra un ejemplo con los individuos 2 y 3.

Padres	Hijos	Reparación de individuos
[O, B1, A2, A1, T2, D], punto de corte: posición 2.	[O, B1, A2, B2, B1, D]	[O, B1, D]
[O, T3, T2, B2, B1, D], punto de corte: posición 3	[O, T3, T2, B2, A1, T2, D]	[O, T3, T2, D]

Tabla 10. Resultado de la operación de cruce

Una vez generada la descendencia es momento de realizar la operación de mutación de uno de los hijos. Para ello, primeramente, se genera un número aleatorio entre [0, 0,05], si este número es menor que Pm, la mutación tiene éxito. A continuación, se eligen dos posiciones de cromosoma, llamadas P1 Y P2, siendo P1 <= P2 y se genera un nuevo subcamino aleatorio entre esas dos posiciones. Finalmente, se utiliza la función de reparación, si procede, y se calcula el valor de aptitud de los dos hijos, tanto si uno de ellos ha sido mutado como si no. Un ejemplo de la mutación se muestra en la ilustración 23.

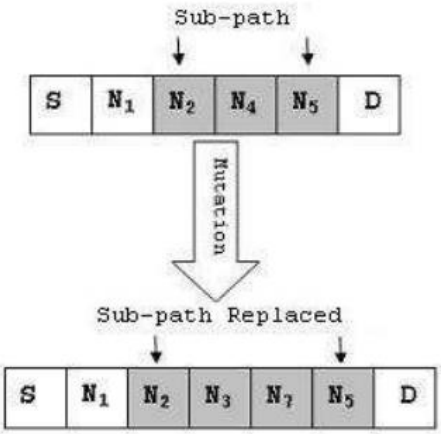

Ilustración 23. Ejemplo de operación de mutación

Supongamos que la mutación ha tenido éxito y que se elige al segundo hijo del proceso de cruce. Imaginemos también que se han generado P1 y P2 y que estos valen 1 y 2 respectivamente. De esta forma el subcamino a mutar es [T3, T2]. Aplicando el algoritmo de inicialización se obtiene el cromosoma [T1, T3, B], el cual es reemplazado a partir de la posición 1 del hijo mutado. En la tabla 10 se muestra el resultado.

Hijo por mutar	Resultado	Reparación del individuo
[O, T3, T2, D]	[O, T1, T3, B]	No procede

Tabla 11. Resultado de la operación de mutación

Finalmente, se calcula la función fitness para los dos hijos, obteniendo dos descendientes. En la tabla 12 se muestra un ejemplo, en el que se puede observar cómo los hijos generados presentan un mejor valor de aptitud que sus padres y del resto de soluciones.

Población tras la evaluación de aptitud		Valor de aptitud
1	[O, B1, D]	2 + 7.5 = 9.5
2	[O, T1, T3, B]	2 + 7 + 2 = 11

Tabla 12. Valor de aptitud de los hijos generados.

En el último paso se ha de insertar los individuos generados en la población. Para este problema se ha elegido trabajar con una única población de tamaño fijo, por lo que, para insertar un nuevo individuo se ha de eliminar previamente a otro. Para ello, se ha elegido utilizar una estrategia de reemplazo de los peores, que consiste en seleccionar los dos peores individuos de la población y reemplazarlos por la descendencia. El resultado de esta operación se muestra en la tabla 13.

Población tras la primera iteración		Valor de aptitud
1	[O, D]	15
2	[O, B1, A2, A1, T2, D]	18.5
3	[O, B1, D]	9.5
4	[O, T1, T3, B]	11

Tabla 13. Población resultante de la primera iteración.

Observemos, que tras esta primera iteración los individuos generados ya mejoran incluso al mejor individuo de la población inicial A partir de este punto, hay que decidir si la realización del algoritmo genético prosigue o se detiene. Como condición de parada, se ha elegido utilizar un número máximo de iteraciones y comparar la mejor solución obtenida hasta el momento con la solución ofrecida por Google (en cuestión de tiempo). En caso de que la mejor solución sea igual o mejor que la de Google, el algoritmo se detiene. Si no es el caso, el algoritmo prosigue hasta alcanzar el número máximo de iteraciones, repitiendo los pasos descritos anteriormente.

4.3.3 Reducción de estaciones a visitar

Como se ha visto en la sección 4.1 de este documento, el número de estaciones a visitar tiene un impacto crucial en la obtención de la solución óptima. Asimismo, en el capítulo 4.3.1 se ha presentado el número de estaciones que tienen cada uno de los tres tipos de transporte utilizados. Para ser exactos, disponemos de 1389 estaciones entre los tres servicios de transporte, por lo que encontrar una solución óptima se vuelve imposible. También, en la inicialización del algoritmo genético, se ha hablado de que el tamaño de las listas puede tener como mucho N estaciones, siendo N, 1389 en nuestro caso.

Generar soluciones con tantos nodos no es una buena práctica. Es poco probable encontrar caminos representativos del problema. Por ejemplo, si el usuario está interesado en ir desde la universidad politécnica hasta el centro de Valencia, con todas las estaciones presentes, una solución posible podría ser ir desde la Universidad hasta Patraix, y desde ahí, ir al centro. Lo cual no tiene sentido. Para solucionar este problema, se ha diseñado un algoritmo que limita las estaciones a visitar durante la ejecución del algoritmo genético, reduciendo para ello, el factor N antes de la inicialización del algoritmo. Consiste en lo siguiente:

En primer lugar, se obtiene la longitud y latitud exacta de los puntos de inicio y destino. A continuación, se traza una línea recta de *m* metros y se obtiene el punto intermedio entre las dos estaciones. Después se dibuja un círculo de diámetro *m*, tomando como punto central el punto situado en *m/2.* Finalmente, para cada estación del sistema (llamémosle s) se calcula la distancia que hay entre s y el punto central. Si la distancia entre la estación s y el punto central es mayor que el radio del círculo, entonces se desecha en el cálculo del algoritmo. En la ilustración 24 se muestra el proceso gráficamente:

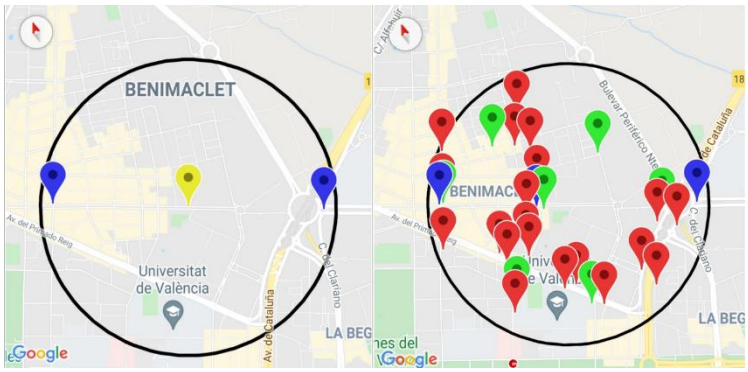

Ilustración 24. Ejemplo de reducción de estaciones

5. Implementación

A lo largo de esta sección, se detallará toda la información referente a la implementación del proyecto. Para ello, se empezará hablando de las tecnologías y herramientas empleadas y posteriormente se mostrará la aplicación móvil desarrollada y la estructura del proyecto. Finalmente, se mostrarán varios resultados y se compararán con las rutas ofrecidas por las aplicaciones de *Google Maps* y Emt Valencia.

5.1 Tecnologías y herramientas

En este apartado, como se ha comentado anteriormente, procederemos a detallar las diferentes tecnologías usadas durante la implementación del proyecto.

Android Studio

Android Studio es el entorno de desarrollo estándar para el desarrollo de aplicaciones en Android [12]. Además de disponer de un editor de códigos completo y varias herramientas potentes, ofrece funciones que aumentan la productividad cuando se desarrollan aplicaciones para Android, como son las siguientes:

- Un sistema de compilación flexible, permitiendo compilar y ejecutar aplicaciones al mismo tiempo.
- Un emulador rápido y cargado de funciones para disponer de dispositivos móviles virtuales en el ordenador.
- Un editor de diseño para crear la interfaz gráfica de la aplicación mediante archivos XML.
- Inserción de código en la aplicación que se encuentra en ejecución, sin necesidad de reiniciarla.
- Herramientas de depuración avanzadas.
- Toda la tecnología disponible en Java 8 (expresiones lambda, gestión multihilo, Java Time, etc.).

Room

Room es una librería de base de datos producida por el equipo de Google Android que simplifica la tarea de trabajar con bases de datos. Es tan fácil de usar, que en unos minutos se puede disponer de una base de datos lista para ser usada. Funciona como un servicio en segundo plano que actúa de forma intermediaria con la aplicación. Por lo que no bloquea la ejecución del programa al realizar consultas u otras operaciones. Asimismo, abstrae el modelo de datos en simples clases Java, haciendo que el programador no necesite tener en cuenta aspectos como son el modelado de tablas o la definición de claves.

SQL

SQL es un lenguaje de programación que permite realizar operaciones sobre las bases de datos relacionales. Mediante este lenguaje se efectúa la gestión de los datos, permitiendo la creación, borrado, y modificación de tablas. Se ha utilizado junto a Room para indexar los archivos GTFS en el modelo de datos presentado en la sección 4.3.1 y

ejecutar consultas un poco más avanzadas, como puede ser, comprobar si dos estaciones de autobús o tren pertenecen a una misma línea.

Google Maps Platform

Maps de Google permite crear experiencias simples y personalizadas para acercar el mundo real a los usuarios a través de mapas estáticos y dinámicos. Ofrece un Kit de desarrollo de software para añadir mapas a una aplicación de *Android.* También es posible ofrecer información adicional sobre las ubicaciones del mapa y facilitar la interacción con el usuario mediante el uso de marcadores, polígonos y superposiciones. Para poder hacer uso de esta herramienta, es necesario registrarse en https://developers.google.com/ y obtener una clave API. Asimismo, el servicio es totalmente gratuito para *iOS* y *Android*, por lo que es una herramienta muy útil para trabajar con mapas en *Android*.

Google Directions API

La API *Directions* de Google es un servicio web que ayuda a los usuarios a encontrar el mejor trayecto hasta su destino. Con esta API es posible obtener rutas para varios modos de transporte, como son el transporte público, la bicicleta, el auto o el ir a pie. Su uso se hace mediante peticiones HTTP en la que se incrustan una serie de parámetros, como son el inicio, el destino, el modo de transporte a emplear o la hora de salida. Asimismo, cuando se procesa la petición, devuelve una dirección formateada con varios parámetros, como son el tiempo de desplazamiento, los kilómetros recorridos, el itinerario a seguir o una polilínea del trayecto, que puede ser utilizada junto a *Maps* de Google para dibujar el recorrido en el mapa. Para poder hacer un uso de esta herramienta, es necesario registrarse en la misma página que *Google Maps Platform* y obtener una clave API. También cabe destacar, que, aunque presente muchas ventajas, su uso es de pago, por cada petición lanzada se le cobra al usuario 0.005 USD (5.00 USD por cada 1000 peticiones) [13]. No obstante, el desarrollador dispone de 200 USD de crédito mensual gratuito, el cual es suficiente para la mayoría de los usuarios.

Java Client for Google Maps Services (JVGM)

JVGM es una librería cliente desarrollada para Java que actúa como un intermediario entre las API's de Google y Java. Simplifica la obtención y el procesamiento de rutas obtenidas por *Google* ofreciendo un patrón de diseño Singleton, por lo que, mediante la instancia de una clase, se pueden realizar y obtener peticiones directamente en código Java. Se pueden encontrar más detalles de esta librería en https://github.com/googlemaps/google-maps-services-java.

5.2 Aplicación desarrollada

En esta sección se documenta el algoritmo genético que se ha desarrollado. Es importante recordar que la implementación de la parte usuario no es un objetivo principal del proyecto, por lo que se ha decidido crear una aplicación simple y sencilla con la finalidad de que se tenga una interfaz totalmente funcional con la que calcular las diferentes rutas. Asimismo, para la implementación de la aplicación se han efectuado las siguientes asunciones:

- Se asume el correcto funcionamiento de las Api's de Google, Valenbisi y del cliente Java para *Google Maps*. Asimismo, se asume que la información transferida por estos servicios es correcta.
- Se asume que la información contenida en los ficheros GTFS está actualizada y que se incluye información veraz sobre las paradas y horarios.
- Se asume que solo hay un modelo de autobús en Valencia. Concretamente, el modelo "*Renault Citybus*", que corresponde al vehículo más desplegado por parte de la empresa EMT Valencia [14]. Se ha hecho está asunción debido a que, en el cálculo de la ruta, no ha sido posible obtener los modelos de autobuses que circulan para un horario concreto. Dicho vehículo, emite 69 gramos de CO_2 por cada kilómetro recorrido según la información ofrecida por IDAE.

De la misma manera, también se han realizado las siguientes simplificaciones:

- El origen y el destino serán siempre una estación del sistema.
- La hora de salida del usuario corresponde a la hora actual del sistema cuando se inicia el algoritmo.
- Cuando el algoritmo tenga que evaluar la función de aptitud entre dos estaciones de tipo bicicleta, asumirá que siempre hay bicicletas en la estación de origen, y huecos disponibles en la estación de destino.

Por otro lado, para el desarrollo de la aplicación se ha hecho uso de la tecnología ofrecida por *Android*, siguiendo para ello las buenas prácticas dictadas en su página web oficial [15]. En la ilustración 25 se muestra la apariencia de la interfaz gráfica:

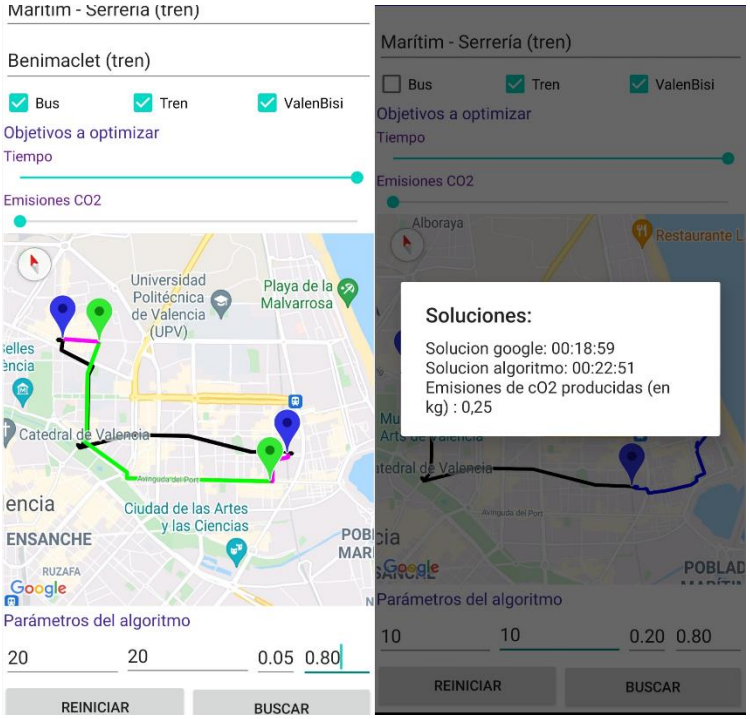

Ilustración 25. Interfaz usuario de la aplicación

Su uso es muy sencillo. Cuando se entra por primera vez en la aplicación, se indexan todas la estaciones e información sobre los horarios. Este proceso es bloqueante y no es posible realizar ninguna acción hasta haber sido finalizado. Una vez se tenga toda la

información, es posible introducir el origen y el destino mediante dos listas desplegables, que disponen de una función de autocompletar. También, se muestra en todo momento la localización de ambas estaciones, mediante el mapa incrustado en la aplicación. A continuación, se pueden seleccionar los transportes con los que moverse. Si no se selecciona ninguno, por defecto toda la ruta se calculará a pie. También se ofrecen dos barras de búsqueda, para que sea posible seleccionar la importancia de los objetivos a optimizar. Dichas barras, deben de sumar siempre el 100%, por lo que darle importancia a un objetivo, disminuye la importancia del otro. De esta forma, se puede seleccionar la ruta más rápida, la menos contaminante o una combinación de ambas. Asimismo, también se pueden seleccionar los parámetros población, iteraciones, probabilidad de cruce y probabilidad de mutación del algoritmo genético. Finalmente, se puede empezar con el cálculo dándole al botón de buscar, y reiniciar los campos y la obtención de la ruta en cualquier momento pulsando el botón de Reiniciar.

Al finalizar el cálculo, se mostrará un recuadro con la solución obtenida (tiempo de desplazamiento y gramos de CO_2 emitidos por viaje). Asimismo, se podrá ver el tiempo de desplazamiento que ofrece Google para el mismo trayecto a la misma hora. Una vez se cierre el *popup* se mostrará el mapa una serie de elementos. En primer lugar, estarán pintadas todas las estaciones que intervienen en la solución a modo de marcadores. Si el marcador es rojo, significa que la estación es de tipo autobús, si es de color azul significa que la estación es de tipo tren y si es verde significa que la estación es de tipo Valenbisi. Asimismo, cada par de marcadores estará conectado por una ruta de color rosa, roja, azul o verde. En caso de ser rosa significa que el trayecto entre ambos marcadores se realiza a pie. Por el contrario, si es roja, el trayecto se hará en bus y es azul si se hace en tren. Finalmente, una línea verde, significa un desplazamiento en bicicleta. Por otra parte, también se ha pintado una ruta de color negro, que corresponde con la mejor ruta que ofrece Google para el mismo problema.

5.3 Estructura del proyecto

Este apartado detalla la organización y estructura de carpetas utilizadas durante la implementación del proyecto. De esta forma, el lector puede orientarse en el código si así lo desea. Asimismo, en los anexos se proporcionará todo el código fuente disponible desde la plataforma de *GitHub*.

A tal efecto, en la figura 26 se muestra el contenido de la estructura del proyecto. Concretamente, en el paquete *Presentation* se muestran todos los archivos que tienen que ver con la interfaz gráfica. Dentro de esta carpeta, en *MapsActivity* se cargan todos los elementos gráficos y se realiza la llamada a la implementación del algoritmo genético. Dicho algoritmo se puede encontrar en la carpeta *Algorithms*. Asimismo, en las carpetas *Database y Poco* se ubica toda la lógica necesaria para guardar en la base de datos todo el modelo de datos presentado en el diseño de la solución. Finalmente, en la carpeta *Utils* se puede encontrar el indexador de estaciones y varias utilidades desarrolladas durante la aplicación.

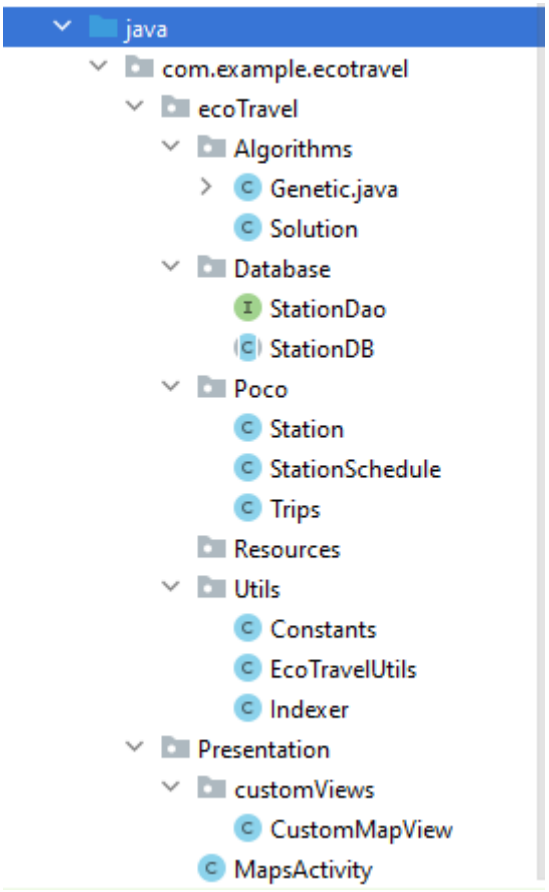

Ilustración 26. Estructura de directorios

5.4 Pruebas y resultados

En esta sección se mostrarán las pruebas ejecutadas para obtener los mejores valores posibles para algunos de los parámetros que afectan al rendimiento del algoritmo genético diseñado. Asimismo, se enumerarán las dificultades a la que nos hemos visto expuestos para poder obtener dichos valores. A continuación, se presentarán los resultados obtenidos para diferentes tipos de rutas y se compararán con los resultados ofrecidos por las dos aplicaciones analizadas.

5.4.1 Análisis de parámetros

No existe ningún método o regla que permita estimar los parámetros de un algoritmo genético, ya que estos dependen de muchos factores, como pueden ser el tipo de problema, los operadores elegidos o la longitud de los cromosomas. Por ejemplo, tal y como lo expresa [10] en su problema, el tamaño de la población tiene un gran impacto en la ejecución del algoritmo. Un tamaño de población pequeño derivará en un tiempo de ejecución más corto. Por otro lado, un tamaño de población grande hará que se exploren más zonas y ayudará a mantener la diversidad de la población. Estos dos objetivos son contradictorios, por lo que es necesario encontrar un equilibrio entre ambos. Asimismo, en su artículo, también comentan que el tamaño de la población ha sido el parámetro más influyente a la hora de encontrar un buen compromiso entre un tiempo de ejecución corto y la obtención de una buena ruta. Para rutas cortas, con

solamente dos individuos es posible alcanzar una buena solución. Por el contrario, para rutas más largas y complejas, un tamaño de población de 20 o 30 individuos puede llegar a ser una buena opción.

Para este trabajo, hemos decidido realizar un análisis de estos parámetros de forma manual, es decir, observando como un cambio en estos, afecta al desempeño del algoritmo genético. Asimismo, decidimos efectuar las pruebas solamente sobre el tamaño de la población, ya que, aunque nos hubiera gustado realizar un análisis más exhaustivo, durante la realización del trabajo nos encontramos con dos limitaciones que expondremos a continuación:

- La primera limitación fue debida al coste económico que existe con el uso de la API de *Google Routes*. Como se ha comentado en la sección 5.1, al usuario se le ofrecen 200 USD de crédito gratuito mensual, lo que equivale a 40.000 solicitudes por mes. Por lo que, si quisiéramos, por ejemplo, generar una población inicial con un tamaño de 100 individuos, cada uno con aproximadamente 30 estaciones, podríamos ejecutar este acto tan solo trece veces antes de agotar el saldo.
- El segundo problema con el que nos hemos encontrado también ocurrió por el uso de *Google Routes*. Cada petición a su API tarda hasta 3 segundos en procesarse, dependiendo de lo saturado que esté su servicio, por lo que es un cuello de botella importante para el tiempo de ejecución del algoritmo. De esta forma, por ejemplo, en el intento de resolver una ruta, cuando su servicio estaba saturado, con una población de 20 individuos y 20 iteraciones, nos encontramos con un tiempo de resolución de 4 minutos y 16 segundos. De este tiempo, 4 minutos y 6 segundos fueron destinados a ejecutar y a esperar peticiones del servicio de *Google Routes*. De esta manera, un aumento considerable en el tamaño de la población o en las iteraciones puede llegar a acarrear a un tiempo de resolución demasiado alto. Aunque se estudió la alternativa de usar otras tecnologías, no encontramos ninguna que realizara una función similar.

Es por ello por lo que decidimos realizar el análisis únicamente sobre el tamaño de la población. También es importante destacar que la estimación se puede aplicar de la misma forma para todos los parámetros del algoritmo genético por separado. Por lo que, sirve como una base para esta sección.

Ahora bien, procederemos al análisis del parámetro. Para ello, primeramente, justificaremos la decisión de analizar el tamaño de la población y no otros parámetros. En primer lugar, este parámetro es el que más influencia tiene en el tiempo de ejecución, como se ha comentado anteriormente, ya que, será el que más llamadas hará al servicio de *Google* (sobre todo en la inicialización del algoritmo). Su estudio es fundamental. El número de generaciones es un parámetro igual de relevante para el tiempo de ejecución (cuantas más iteraciones, más tiempo). Si elegimos un valor muy pequeño, obtendremos soluciones poco precisas. En cambio, un valor muy grande hará que se produzca, posiblemente, una población homogénea a partir de un número de iteraciones y hará que se siga iterando sobre una población que no pueda evolucionar más. Es por ello por lo que preferimos definir un número de 100 iteraciones y parar el algoritmo cuando se detectara la homogeneidad, asumiendo el riesgo de que esta no se produjera. A partir de este punto ya solo nos quedan dos parámetros, las probabilidades de cruce

y mutación. Estas, tienen un gran impacto en la exploración y explotación del algoritmo. En otras palabras, la probabilidad de cruce afecta a la búsqueda de soluciones, localizando zonas prometedoras del problema, mediante la presión selectiva o la competición entre los padres, por ejemplo. Por otro lado, la tasa de mutación introduce diversidad en la población, permitiendo que se exploren zonas nuevas. De esta forma, es necesario encontrar un equilibrio entre estos dos factores. En la resolución de este problema decidimos utilizar una probabilidad de mutación de 0.05, esperando que, en promedio, un 5 % de los individuos mutaran. Asimismo, usamos una tasa de cruce de un 70%, para que individuos con peor ajuste, tuvieran cierta probabilidad de reproducirse.

De esta forma, se usará un AG con unos parámetros por defecto, excepto para las estaciones de inicio y destino, que serán escogidas de forma aleatoria entre las estaciones existentes del sistema. Asimismo, se probará con diferentes tamaños de población y se analizará como afectan al rendimiento del algoritmo genético. El resumen de los parámetros puede observarse en la siguiente tabla:

Parámetros	**Valor**
Estación de origen	Aleatorio
Estación de destino	Aleatorio
Transportes usados	Tren, autobús y Valenbisi
Hora de salida	15:00
Tamaño de población	{2,5,10,20,50,80,100}
Número de generaciones	100 o hasta homogeneidad
Probabilidad de mutación	0.05 (5%)
Probabilidad de cruce	0.70 (70%)
Tipo de ruta	Ruta multimodal más corta

Tabla 14. Valores del algoritmo genético para las pruebas

Asimismo, para cada tamaño de población dado, se realizará 10 veces el mismo cálculo de la ruta y se contará las veces en la que la solución ofrecida por el algoritmo mejora u ofrece un tiempo similar a la de *Google Maps* (entendiendo como similar, aquellas rutas que ofrezcan un incremento en el tiempo de desplazamiento no superior a 5 minutos). También, se mostrará el tiempo medio de las ejecuciones para cada tamaño de la población. Los resultados obtenidos se muestran en las figuras 27 y 28.

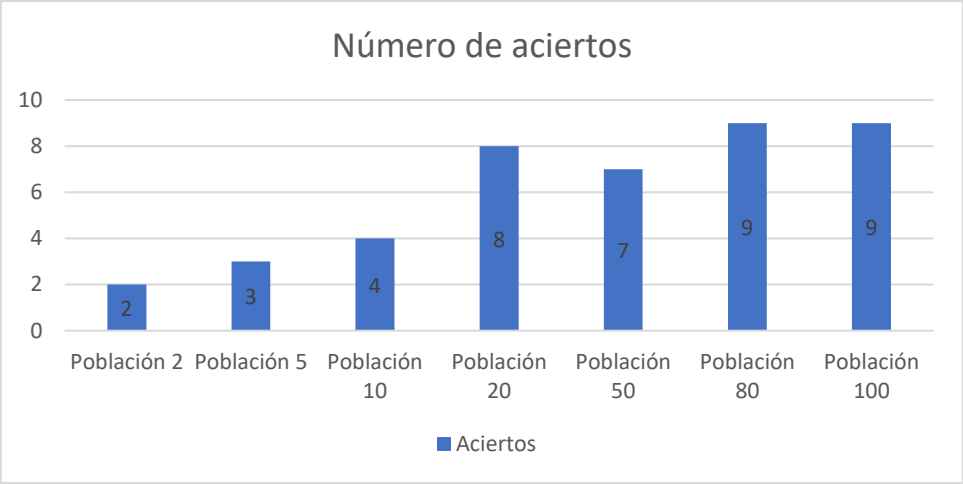

Ilustración 27. Gráfico de aciertos.

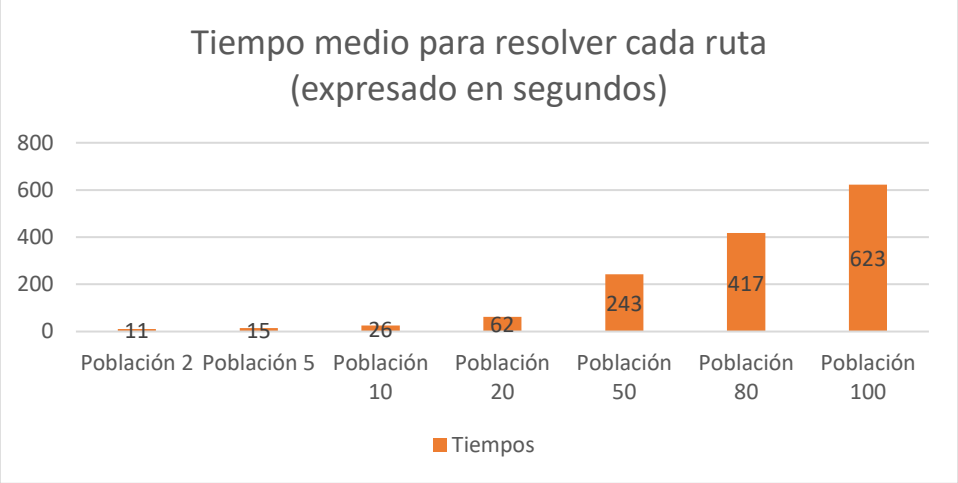

Ilustración 28. Gráfico de tiempos

En la figura 28 podemos observar que no hay mucha mejora para tamaños de población reducidos. Esto se debe a la poca variabilidad que existe en la población. También podemos ver que, a partir de los 20 individuos y hacia arriba, el AG es capaz de encontrar soluciones aproximadas, iguales o mejores que las propuestas por *Google*. Por otro lado, en la figura 28 podemos contemplar como efectivamente el tamaño de la población afecta al tiempo de ejecución. Disparándose a partir de 50 individuos. Por lo tanto, vistos los resultados, elegiremos un tamaño de población de 20 individuos, ya que es el mejor candidato si tenemos en cuenta el ratio de número de aciertos y del tiempo.

5.4.2 Resultados obtenidos

A continuación, mostraremos los resultados de varias rutas calculadas por el AG y las compararemos con las soluciones ofrecidas por Google y Emt Valencia. Para calcular las rutas de Google se ha utilizado el servicio de Google *Route*. Construyendo una petición con los parámetros origen, destino y hora de salida utilizados por el algoritmo genético. Dicho servicio, devuelve el tiempo de desplazamiento entre los dos puntos y una representación del camino a seguir. Esta información se mostrará directamente en la aplicación, pintando en el mapa, una polilínea negra con el recorrido ofrecido por

Google. También cabe destacar, que la ruta ofrecida por Google *Route* es la misma que se le muestra al usuario cuando utiliza la aplicación de Google *Maps*. Por otro lado, para comparar nuestra ruta con la ofrecida por EMT Valencia, no nos ha quedado más remedio que insertar los parámetros manualmente en su aplicación. Finalmente, en la tabla 15 se muestra un resumen de los parámetros utilizados en el algoritmo genético.

Parámetros	**Valor**
Estación de origen	Elegida por el usuario
Estación de destino	Elegida por el usuario
Transportes usados	Tren, autobús y Valenbisi
Hora de salida	Hora del sistema al lanzar la búsqueda
CO_2 emitido por el tren/tranvía	0 gramos por kilómetro recorrido
CO_2 emitido por la bicicleta	0 gramos por kilómetro recorrido
CO_2 emitido por el Bus	69 gramos por kilómetro recorrido
Tamaño de población	20
Número de generaciones	100 o hasta homogeneidad
Probabilidad de mutación	0.05 (5%)
Probabilidad de cruce	0.70 (70%)
Tipo de ruta	Ruta multimodal más corta y/o ruta multimodal menos contaminante

Tabla 15. Valores finales del AG.

De este modo, hemos empezado por calcular rutas que minimicen únicamente el tiempo de llegada (ruta multimodal más corta), con los tres tipos de transportes disponibles. El cálculo de estas rutas equivale a las que ofrecen Google *Maps* y EMT Valencia. En las ilustraciones 29 y 30 mostramos dos ejemplos. En la primera de ellas, se ha calculado el recorrido entre las estaciones de metro Benimaclet y Serrería. Como puede observarse, el tiempo de desplazamiento equivale a 8 minutos y 22 segundos, tanto para el algoritmo genético como para la solución ofrecida por Google. Esto significa que ambas rutas son iguales (este hecho también puede observarse porque la polilínea de color azul superpone a la de color negro). De esta forma, se propone coger el tranvía para desplazarse entre estas dos estaciones. Por otro lado, EMT Valencia sugiere una ruta de 22 minutos en autobús (aunque esté el icono de metro seleccionado). En la ilustración 29 hemos obtenido los mismos resultados, pero para obtener un camino entre dos estaciones que están en la zona de poblados Sur, a las afueras de Valencia. De esta forma, el algoritmo es capaz de encontrar la misma solución que la de Google y además mostrar los kilogramos de CO_2 producidos por el desplazamiento en autobús. Por otro lado, la aplicación de EMT Valencia propone el mismo camino, sin embargo, con un tiempo de duración considerablemente mayor.

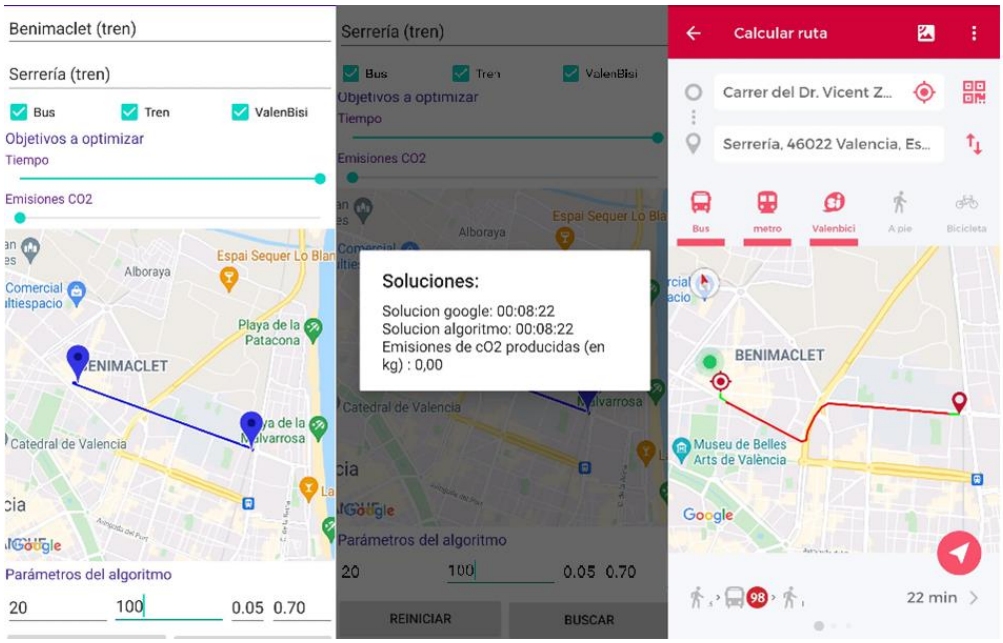

Ilustración 29. Ruta más corta dentro de Valencia

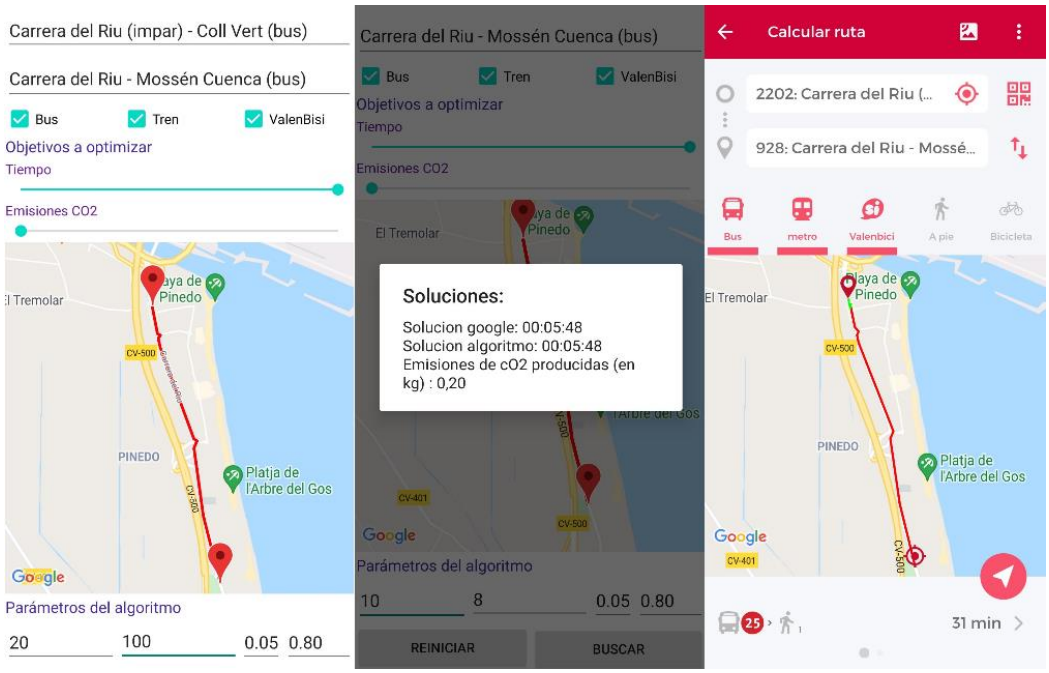

Ilustración 30. Ruta más corta en la periferia.

El siguiente tipo de ruta que hemos puesto a prueba ha sido la ruta menos contaminante. En la ilustración 31 mostramos un ejemplo. En dicha ruta se recomienda ir desde la estación de tranvía “Universitat politécnica” hasta una estación de autobús en la Malvarrosa. Asimismo, hemos desactivado la posibilidad de ir en tren. De este modo, como se puede observar, al algoritmo no le queda más remedio que escoger rutas que incluyan desplazamientos en bicicleta y a pie (ya que el autobús es un medio contaminante). También observamos como se intentan priorizar los desplazamientos en bicicleta frente al ir a pie. Como comentamos en el diseño de la solución, siempre se

intenta ir en el transporte más rápido. De esta forma, se le indica al usuario de ir andando hasta una estación de Valenbisi cercana, luego ir en bicicleta hasta un punto en la Malvarrosa, y finalmente ir andando hasta el destino, con un tiempo de 25 minutos y 17 segundos. Un tiempo muy próximo a la ruta elegida por la aplicación EMT. Por otra parte, Google *Maps* ofrece una solución de 15 minutos y 33 segundos. Utilizando para ello el metro, ya que, este servicio solo calcula rutas que minimicen el tiempo de llegada y además no tiene en cuenta el servicio de Valenbisi en sus cálculos.

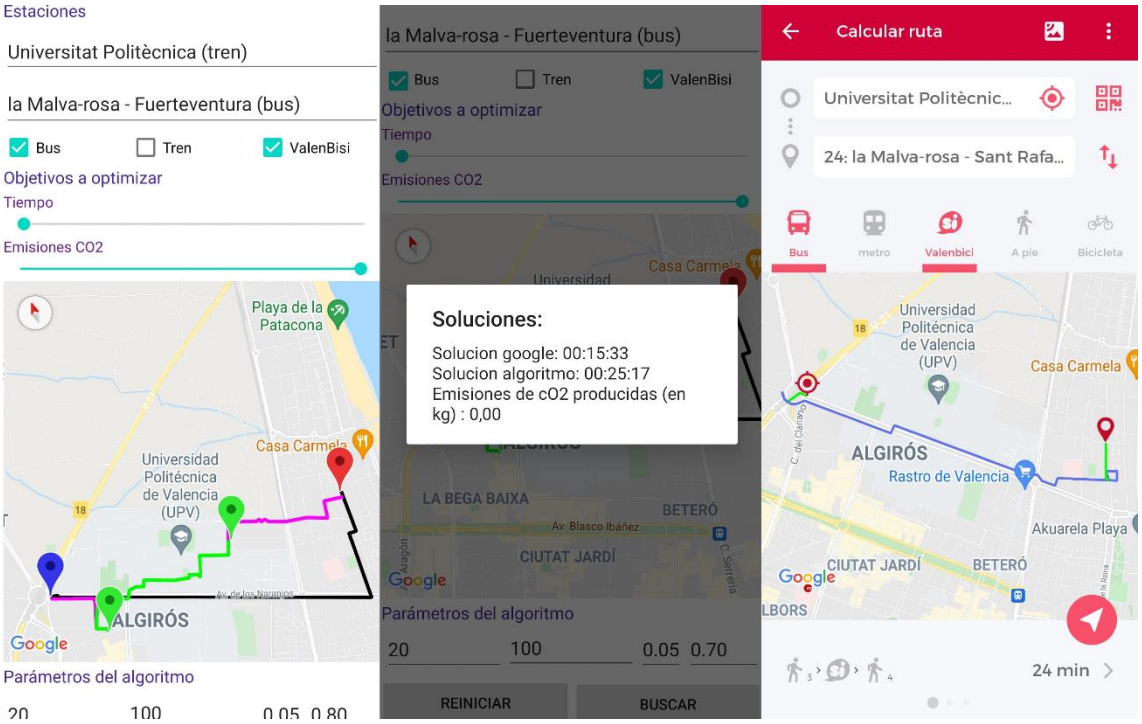

Ilustración 31. Ruta menos contaminante

Asimismo, en ocasiones, el algoritmo genético también es capaz de mejorar las soluciones ofrecidas por los otros servicios. En el recorrido de la ruta de la ilustración 32, se propone una ruta que va desde la estación de metro “Vicent Zaragozá” hasta una estación de Valenbisi, ubicada cerca de la estación “Maritim-Serrería”. Como se puede observar, tanto el algoritmo genético como Emt Valencia muestran que el mejor recorrido es ir en bicicleta. Sin embargo, nuestro algoritmo elige un camino mucho más corto. Por otro lado, como Google no utiliza el servicio de Valenbisi para calcular sus rutas, sugiere hacer el recorrido íntegramente en tranvía hasta la estación de “Maritim-Serreria”. Con una duración de 26 minutos y 5 segundos.

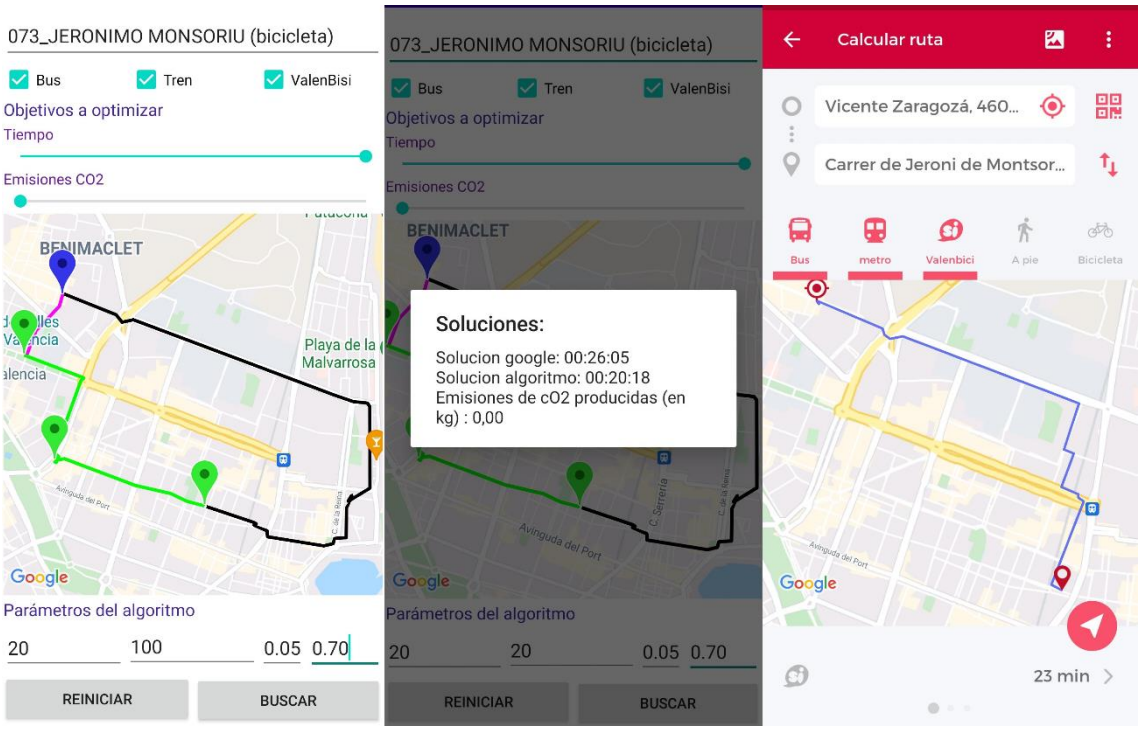

Ilustración 32. Ruta mejorada

Finalmente, el algoritmo también es capaz de encontrar rutas multimodales, las cuales se pueden observar en distancias más largas. En la ilustración 33 se muestra un ejemplo, mostrando, además una configuración en la que se prioriza más la contaminación que el tiempo (80% frente a un 20%). Como puede observarse, nuestra solución es idéntica a la de Google, con la diferencia de que, después de coger el tranvía se hace el recorrido restante en bicicleta. Por otro lado, la aplicación de EMT Valencia recomienda el recorrido de la ruta, íntegramente en bicicleta, superando el tiempo de desplazamiento de las otras dos aplicaciones.

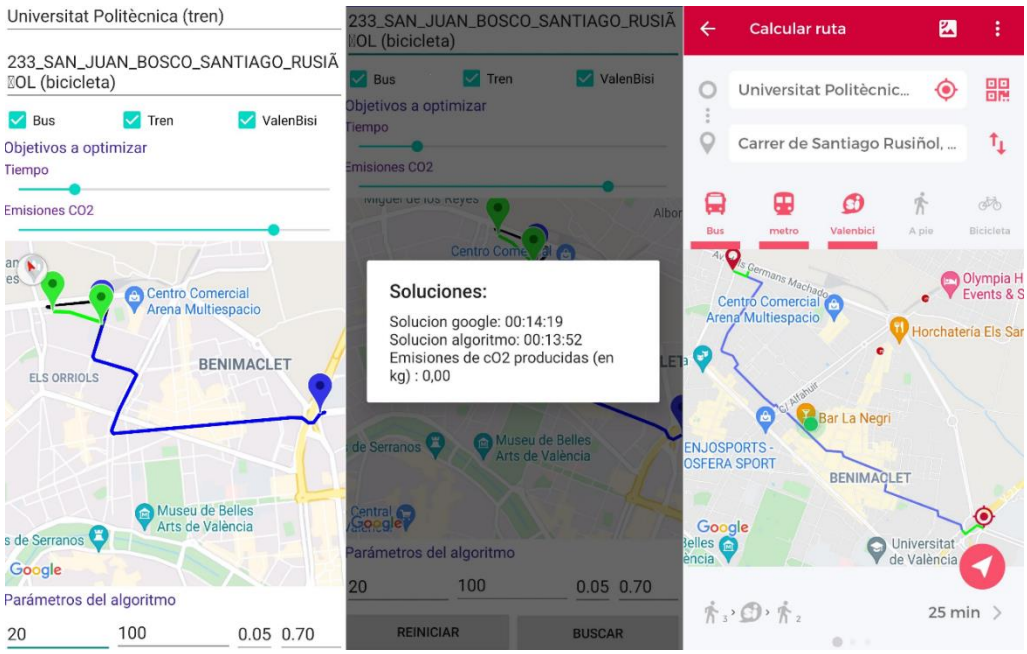

Ilustración 33. Ruta multimodal

6. Conclusiones y relación del trabajo desarrollado con los estudios cursados.

Para terminar esta memoria, en este apartado veremos el cumplimiento de los objetivos propuestos. También se verán las áreas que no se han podido investigar por falta de tiempo disponible para la realización del TFG. Asimismo, se recomendarán ampliaciones o mejoras a implementar que puedan enriquecer este proyecto. Finalmente, se expondrán cuáles han sido los conocimientos que han servido como un pilar para la realización de este proyecto.

6.1 Conclusiones

El principal objetivo de este proyecto era el de desarrollar un algoritmo genético que permitiera obtener tres tipos de rutas multimodales. La ruta más rápida, la menos contaminante o una que tuviera en cuenta ambos factores. Podemos dar por cumplido este objetivo, vistos los resultados obtenidos en el apartado 5.4.2. Concretamente, hemos conseguido formalizar el problema, además de plantear los dos objetivos contradictorios a optimizar. Asimismo, hemos conseguido mejorar las soluciones ofrecidas por otras aplicaciones y cumplido con los objetivos expuestos en la tabla 1. Por otro lado, también hemos conseguido centralizar los diferentes sistemas de transportes de la ciudad en un modelo de datos muy sencillo y eficiente. De esta forma, el algoritmo es capaz de encontrar rutas multimodales con estos transportes e incluso mejorar el tiempo de llegada de algunos caminos.

No obstante, es cierto que no hemos podido cumplir completamente con el diseño de los parámetros del algoritmo genético, sobre todo, por las limitaciones a las que nos hemos visto expuestos al utilizar la API de Google *Routes*. En los últimos dos meses hemos llegado a gastar totalmente el saldo gratuito de este servicio. Principalmente, en la realización de pruebas y la estimación del valor del tamaño de la población del AG. Aunque hemos comprobado como afectaba el tamaño de la población al tiempo de ejecución y escogido un parámetro conveniente para nuestro problema, no hemos tenido en cuenta otros criterios como puede ser estimar el número de generaciones o elegir la tasa de cruce. Asimismo, existen otras métricas aparte del tiempo de ejecución que se pueden tener en cuenta, como, por ejemplo, estudiar la mejora del fitness con el paso de cada generación.

En conclusión, aunque hayamos conseguido buenos resultados, hacen falta más estudios para ver que parámetros se adaptan mejor en el cálculo de la ruta. Aprovechando que, en el mes de diciembre, tendremos de nuevo saldo disponible para usar la API de Google *Route*, aprovecharemos para realizar pruebas más completas y mostrarlas en la defensa del proyecto.

6.2 Trabajos futuros

Aunque se hayan cumplido los objetivos buscados, todavía hay una serie de mejoras que se pueden investigar para que esta aplicación sea más completa y eficiente. Algunos de los cambios y mejoras por estudiar podrían ser:

- Analizar otras técnicas metaheurísticas enfocadas en resolver problemas de optimización. Existen muchas aparte de los algoritmos genéticos, como pueden ser la optimización por enjambre de partículas, por colonias de hormigas o búsquedas Tabú. Aunque parte de la idea original del proyecto era estudiar e implementar varias de estas técnicas, por cuestiones de tiempo no ha sido posible. Esto abre la puerta para analizar el problema desde la perspectiva de otras técnicas.
- Realizar un cálculo más exhaustivo del cálculo de la huella de carbono. Para el diseño del algoritmo genético solo se ha tenido en cuenta el factor de CO_2 de los vehículos utilizados. Aunque es una buena métrica, es verdad que hay muchos otros factores que intervienen en la huella de carbono. Incluir un cálculo más detallado sobre esta parte podría ser una mejora por ejecutar.
- Encontrar alternativas a la API de Google *Routes,* ya que, como se ha mostrado en el capítulo de implementación, se ralentiza considerablemente la ejecución del algoritmo genético, por lo que el uso de las alternativas podría mejorar el tiempo que se tarda en obtener una ruta multimodal.
- Incluir modos de transporte privados como el coche o la motocicleta, adaptando el algoritmo genético para que se tengan en cuenta.
- Mejorar la interfaz gráfica y mostrar un itinerario de la ruta a seguir, ya que, aunque no era un objetivo principal del proyecto, es conveniente de realizar para ofrecer una mejor experiencia de usuario.

6.3 Relación del trabajo con los estudios cursados

Muchas asignaturas han servido de base para la realización de este trabajo de fin de grado, se podría decir incluso, que muchas han aportado su grano de arena en la resolución del problema planteado. De manera general, los estudios de formación básica han permitido al alumno obtener diferentes destrezas sobre los lenguajes de programación. Asimismo, asignaturas más complejas han sido de gran ayuda también. Ya que han permitido al alumno enfrentarse a un problema nuevo. Mediante las herramientas obtenidas durante la realización del grado se ha conseguido resolver poco a poco el problema planteado. A continuación, y sin incluir aquellas asignaturas de formación básica, mostraremos aquellas materias que han proporcionado los conocimientos básicos para la realización de este trabajo:

- **Técnicas, Entornos y Aplicaciones de Inteligencia Artificial:** en esta asignatura se le presentó al alumno diversas técnicas y métodos de la Inteligencia Artificial de una manera muy práctica, permitiendo reconocer y resolver nuevos tipos de problemas. El estudio de esta asignatura ha sido clave para la realización de este trabajo, ya que, al alumno, se le introdujo los conocimientos básicos sobre metaheurísticas y algoritmos genéticos. Asimismo, el alumno aprendió a identificar aquellos problemas que no tuvieran solución en tiempo polinómico y dio, una base para resolverlos.
- **Ingeniería del Software:** en esta asignatura se realizó un proyecto en *Azure devops* y se le introdujo al alumno las nociones básicas sobre las metodologías ágiles, por lo que ha servido como base para realizar la metodología del proyecto.

- **Bases de datos y sistemas de información:** esta asignatura capacitó al alumno para el uso avanzado y el diseño básico de bases de datos relacionales, enseñando al alumno un estudio teórico de los principios, modelos y metodologías de diseño de las bases de datos relaciones. De esta forma, gracias a esta asignatura pudimos plantear y desarrollar el modelo de datos sin problema alguno.
- **Sistemas inteligentes:** en esta asignatura se impartió un conocimiento general sobre cómo funcionan ciertas técnicas relacionadas con la IA. Asimismo, se le capacitó al alumno los conceptos de algoritmos de búsqueda, lo cual ha permitido entender el coste computacional del problema planteado.
- **Desarrollo de Aplicaciones para Dispositivos Móviles:** En esta asignatura se estudió el desarrollo de aplicaciones móviles, particularizando dicho desarrollo en el ámbito de Android. Ha permitido al alumno obtener las destrezas necesarias para familiarizarse con la herramienta de Android *Studio* y la programación móvil.

7. Referencias

[1] "Metodología Kanban: en qué consiste y cómo utilizarla | APD." https://www.apd.es/metodologia-kanban/ (accessed Nov. 03, 2021).

[2] "Transporte Multimodal." https://www.transeop.com/blog/transporte-multimodal/29/ (accessed Nov. 03, 2021).

[3] "Sector transporte en España." https://www.miteco.gob.es/es/cambio-climatico/temas/mitigacion-politicas-y-medidas/transporte.aspx (accessed Nov. 14, 2021).

[4] "El metro de Valencia y el tranvía de Alicante usan energía 100% renovable." https://ielektro.es/2018/05/11/metro-valencia-tranvia-renovable/ (accessed Nov. 14, 2021).

[5] M. Gestal, "Introduccion a los Algoritmos Geneticos," 2013. [Online]. Available: https://www.researchgate.net/publication/237812449

[6] "Obtener indicaciones y ver rutas - Ordenador - Ayuda de Google Maps." https://support.google.com/maps/answer/144339?hl=es&co=GENIE.Platform%3DDesktop (accessed Nov. 14, 2021).

[7] "Travel your first and last mile with Google Maps." https://blog.google/products/maps/travel-your-first-and-last-mile-google-maps/ (accessed Nov. 14, 2021).

[8] "3 new ways to navigate more sustainably with Maps." https://blog.google/products/maps/3-new-ways-navigate-more-sustainably-maps/?_ga=2.25448830.293984643.1636914567-1279784916.1635675896 (accessed Nov. 14, 2021).

[9] "EMT Valencia." https://movil.emtvalencia.es/ciudadano/index.php?option=com_content&view=article&id=275:ayuda-gcalcula-tu-ruta&catid=80&lang=es (accessed Nov. 14, 2021).

[10] S. C. Nanayakkara, D. Srinivasan, L. W. Lup, X. German, E. Taylor, and S. H. Ong, "Genetic algorithm based route planner for large urban street networks," 2007. doi: 10.1109/CEC.2007.4425056.

[11] A. Moratilla, E. Fernández, J. J. Sánchez, and B. Vicario, "Selección óptima de operadores para el tratamiento de problemas VRP con Algoritmos Genéticos," 2014.

[12] "Introducción a Android Studio | Desarrolladores de Android." https://developer.android.com/studio/intro?hl=es-419 (accessed Nov. 20, 2021).

[13] "Directions API Usage and Billing | Google Developers." https://developers.google.com/maps/documentation/directions/usage-and-billing (accessed Nov. 21, 2021).

[14] “Evolución.” https://www.emtvalencia.es/ciudadano/index.php?option=com_content&view=article&id=120&Itemid=152&lang=es (accessed Nov. 22, 2021).

[15] “Guía de arquitectura de apps | Desarrolladores de Android.” https://developer.android.com/jetpack/guide?hl=es-419 (accessed Nov. 21, 2021).

8. Anexos

Código de la aplicación:

samcl94/EcoTravel (github.com)