Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño y aplicación de técnicas metaheurísticas para control de tráfico

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Marcos Hernández Yáñez

Tutor: Antonio Garrido Tejero

Curso 2020-2021

Resumen

Cada año millones de euros son desperdiciados a nivel mundial a causa de la congestión de tráfico. Los retrasos provocados por esta congestión se traducen en una serie de consecuencias: reducción de la productividad, aumento de la contaminación ambiental y malgasto de recursos, como, por ejemplo, de los combustibles fósiles. Una de las formas más rentables para reducir este tipo de retrasos es la de optimizar los tiempos de ciclo de los semáforos. Una acción que no requiere de apenas cambios en su infraestructura, sino de la implementación de un pequeño reajuste en los recursos existentes.

Este proyecto tiene por objeto la creación y posterior estudio de una herramienta para la optimización del tiempo de los ciclos semafóricos de una red de carreteras mediante el uso de un algoritmo genético. Dada una red y las rutas de los vehículos que circulan sobre la misma, se busca reducir los tiempos de espera, el consumo de combustible y las emisiones de gases contaminantes y nocivos en dichas vías urbanas.

Palabras clave: optimización, SUMO, metaheurísticas, tráfico, algoritmo genético.

Abstract

Every year millions of euros are lost in the global economy due to traffic congestion. The delays caused by this congestion lead to a series of consequences: reduced productivity, increased environmental pollution and waste of resources, such as, for example, fossil fuels. One of the most cost-effective ways to reduce these delays would be optimizing the cycle timings of the traffic lights. An action that it does not require changes to the infrastructure, but only to implement a small readjustment in the existing resources.

This work consists of the creation and subsequent study of a tool for optimizing the timing of traffic light cycles of a road network using a genetic algorithm. Given a network and the routes of the vehicles that circulate on it, it aims to reduce waiting times, fuel consumption and emissions of polluting and harmful gases in urban areas.

Keywords: optimization, SUMO, metaheuristics, traffic, genetic algorithm.

Resum

Cada any milions d'euros són desperdiciats a nivell mundial a causa de la congestió de trànsit. Els retards provocats per aquesta congestió es tradueixen en una sèrie de conseqüències: reducció de la productivitat, augment de la contaminació ambiental i desaprofitament de recursos, com, per exemple, dels combustibles fòssils. Una de les formes més rendibles per a reduir aquest tipus de retards és la d'optimitzar els temps de cicle dels semàfors. Una acció que no requereix de quasi canvis en la seua infraestructura, sinó de la implementació d'un xicotet reajustament en els recursos existents.

Aquest projecte té com a objecte la creació i posterior estudi d'una ferramenta per a l'optimització del temps dels cicles semafòrics d'una xarxa de carreteres per mitjà de l'ús d'un algorisme genètic. Donada una xarxa i les rutes dels vehicles que hi circulen sobre la mateixa, es busca reduir els temps d'espera, el consum de combustible i les emissions de gasos contaminants.

Paraules clau: optimització, SUMO, metahurístiques, trànsit, algorisme genètic.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

1. Introducción

El tráfico es uno de los principales problemas en las grandes ciudades. El número de vehículos en circulación y el de sus conductores se incrementa cada año. Sin embargo, la infraestructura que se encarga de su gestión y circulación no se adapta a estos cambios. Esto acaba ocasionando un incremento de la congestión del tráfico urbano agravando los efectos negativos que derivan del mismo.

Cuando el nivel de congestión del tráfico es alto, se producen atascos y retrasos en el viaje de los conductores. Por ejemplo, Barcelona es una de las ciudades españolas con más congestión de tráfico. En ella la duración de los viajes se vio incrementada en un 29% [1] en el 2019 a causa de los atascos. Además, otro dato, es que los conductores de las grandes urbes del mundo pasan una media de 2 a 4 días al año esperando en los atascos [2]. Esto también acaba suponiendo una pérdida de productividad de los trabajadores que acuden a su puesto laboral por las mañanas y se encuentran con el tráfico en hora punta. Durante estos tiempos de espera, los vehículos siguen consumiendo combustible de forma inútil, provocando un incremento en su uso y en las emisiones. Estos aumentos provocan pérdidas de cientos de euros anuales a los conductores españoles y de millones a las ciudades al no poder gestionar el combustible de una forma más eficiente.

Las emisiones procedentes de los vehículos contienen diversos gases y partículas contaminantes que afectan en diferente medida al medio ambiente y a la salud de las personas. El CO2 es un gas de efecto invernadero que agrava los efectos del calentamiento global y, por lo tanto, tiene un impacto medioambiental. Otros elementos como el monóxido de carbono (CO), los óxidos de nitrógeno (NOx), los hidrocarburos y las partículas sólidas tienen un impacto nocivo en la salud, que al entrar en contacto con el organismo pueden dañar el hígado, producir afecciones cardiovasculares o provocar la aparición de tumores que deriven en cáncer (principalmente pulmonar). Según la OMS [3] cada año se producen más de 4 millones de muertes por exposición a la contaminación del aire, por lo que debe llevarse un exhaustivo control de la calidad de este, especialmente en las grandes ciudades, y tomar medidas en caso de que se sobrepasen los valores recomendados. Estas medidas suelen repercutir en el tráfico al ser una de las principales fuentes de contaminación del aire en zonas urbanas.

Diferentes estudios han concluido que la acción de realizar pequeños ajustes en la temporización de las señales de tráfico puede llegar a aportar grandes beneficios. Por ejemplo, en Portland, Estados Unidos, se utilizó un software de optimización sobre las 135 intersecciones de 16 calles de la ciudad, consiguiendo así ahorrar más de 6,5 millones de litros anuales de gasolina, y reduciendo en 15,5 toneladas las emisiones de CO2 [4]. También, en el estado de Arizona se realizó un reajuste de la temporización de las señales de dos carreteras que conectan con la I-17, acción con la que se espera ahorrar un total de 350.000 horas de viaje a los motoristas [5].

Figura 1 Contaminación del aire por tráfico. [6]

1.1 Motivación

Como ya se ha comentado en la introducción, el tráfico en la actualidad tiene un gran impacto a nivel económico, medioambiental y sanitario, y aun así se siguen utilizando, en su gran mayoría, normas, sistemas y métodos tanto de diseño como de planificación creados hace muchos años, que están envejeciendo y quedándose obsoletos. Es por ello por lo que debemos empezar a aprovechar las nuevas técnicas y herramientas que nos brindan las nuevas tecnologías y ponerlas en práctica para facilitar y mejorar el proceso de construcción y gestión de la infraestructura de tráfico.

En este proyecto nos enfocaremos en la mejora de la gestión. Concretamente en la optimización de los tiempos de fase y de ciclo de los semáforos, creando una herramienta que permita evaluar y facilitar la elección de estos tiempos, lo que puede llegar a ser una tarea muy compleja, al tener en cuenta un gran conjunto de intersecciones semafóricas. Buscamos conseguir establecer combinaciones de tiempo que mejoren la sincronización de los semáforos y el flujo de la circulación.

Además, se plantea un método que no requiere de cambios en la infraestructura actual, algo que sí es requerido al utilizar otros enfoques que necesiten de la información proporcionada en tiempo real de sensores externos para funcionar correctamente. Solo se realizará una modificación de los tiempos, que se puede hacer con relativa facilidad.

1.2 Objetivos

El objetivo principal de este proyecto es el de crear una herramienta con la que podamos reducir los tiempos de espera, el consumo de combustible y las emisiones de los vehículos. Mejorando así el flujo global del tráfico y reduciendo los atascos en entornos urbanos. La herramienta utilizará un algoritmo genético para la resolución del problema de optimización. Sus parámetros de entrada serán una red de carreteras, como puede ser la de una ciudad real o la de un proyecto futuro, y las rutas generadas para los vehículos, que deberán ser similares a las reales o a las esperadas para resultados más realistas. Los datos más relevantes para el correcto funcionamiento del

algoritmo genético serán recogidos a partir de las simulaciones realizadas en un simulador de microtráfico.

Finalmente, realizaremos un análisis de los resultados obtenidos, comparándolos con los valores iniciales.

1.3 Impacto esperado (Objetivos de Desarrollo Sostenible)

Los objetivos de desarrollo sostenible (ODS) fueron establecidos en 2015 por la ONU. Están formados por 17 objetivos principales y una serie de metas para llevar a cabo cada uno de ellos. Una correcta implementación y resolución del problema daría como resultado una herramienta que aportaría a los siguientes objetivos:

- **Objetivo 3 (Salud y bienestar):** uno de los objetivos principales del trabajo es el de reducir la contaminación del aire en zonas urbanas y, por lo tanto, el número de muertes y enfermedades producidas por esta, minimizando las emisiones de gases nocivos para la salud. También al haber un mejor flujo del tráfico se reducirían los accidentes de tráfico. Reducir los tiempos de espera también ayudaría a disminuir los niveles de estrés de los conductores que en casos extremos podría hacerlos vulnerables a otras enfermedades psicológicas.
- **Objetivo 8 (Trabajo decente y crecimiento económico):** reducir el tiempo perdido en los atascos causaría un incremento en la productividad de los trabajadores y, por lo tanto, de los ingresos de las empresas. Además, también se reduciría el gasto económico anual en combustibles al hacer un mejor uso de ellos.
- **Objetivo 9 (Industria, innovación e infraestructuras):** fomentamos la innovación al utilizar nuevas técnicas y tecnologías para abordar problemas de actualidad e intentar resolverlos. También, se busca facilitar el futuro diseño y evaluación de las redes de carreteras, en lo referente a la asignación y elección de tiempos de ciclo semafóricos.
- **Objetivo 13 (Acción por el clima):** el CO2 es un gas contaminante de efecto invernadero procedente de las emisiones de los vehículos. Minimizando el consumo de combustible ayudamos a reducir estas emisiones y a mitigar sus efectos en el cambio climático

1.4 Estructura de la memoria

La memoria está distribuida en los siguientes capítulos:

1. **Introducción:** en este capítulo hemos realizado una breve introducción al problema que vamos a abordar, la motivación para la realización del proyecto y los objetivos principales que se busca cumplir.
2. **Estado del arte:** aquí detallaremos cómo funcionan los semáforos en la actualidad y abordaremos alguna terminología sobre ellos que se utilizará posteriormente en la memoria. También hablaremos sobre la inteligencia artificial en el transporte, haciendo hincapié en las técnicas metaheurísticas, y finalizaremos realizando una introducción sobre el simulador de tráfico de SUMO.

3. **Fundamentos de los algoritmos genéticos:** capítulo en el que se realizará una introducción a los algoritmos genéticos, los conceptos básicos, su estructura y componentes.
4. **Implementación de la solución:** explicaremos cómo están implementados los semáforos en el simulador SUMO, cómo se ha llevado a cabo la obtención de los datos de entrada necesarios para la herramienta, y cómo se ha realizado la implementación del algoritmo genético.
5. **Pruebas y resultados:** en este capítulo se analizarán los resultados de las pruebas sobre el algoritmo genético que nos permiten obtener buenos parámetros para su ejecución. Después se procederá a realizar una comparación de los resultados finales entre los programas generados por el algoritmo genético y los generados por defecto por el algoritmo SUMO Cycle Program Generator que viene con SUMO.
6. **Conclusiones y trabajos futuros:** finalmente en este apartado veremos si hemos cumplido con los objetivos propuestos en la introducción y haremos una lista con los posibles trabajos futuros.

2. Estado del arte

En este capítulo vamos a empezar realizando una breve explicación sobre ciertas propiedades de los semáforos, su funcionamiento y los tipos de semáforos que hay en la actualidad. Después, hablaremos sobre el uso que está teniendo la inteligencia artificial actualmente en el transporte, centrándonos un poco más en las técnicas metaheurísticas. Finalmente, haremos una breve introducción al simulador de tráfico SUMO, donde hablaremos de sus características principales y los motivos de su elección.

2.1 Semáforos en la actualidad

Un semáforo es un dispositivo de señalización que se sitúa en una intersección para regular el tráfico. Los semáforos se suelen encontrar formando conjuntos semafóricos en las intersecciones, donde cada uno de ellos tiene la responsabilidad de controlar el paso de los vehículos de una vía concreta, y tienen una serie de propiedades:

- **Ciclo semafórico**: es el tiempo que tarda en repetirse una misma situación o fase en un grupo semafórico tras haber realizado una secuencia de maniobra completa de los semáforos conectados a un mismo regulador [7]. Este ciclo se repite infinitamente siguiendo una serie de tiempos predefinidos o actuando respecto a datos proporcionados del entorno.
- **Fase**: es cada una de las divisiones por las que está formada un ciclo en la que la configuración de los colores del conjunto semafórico permanece invariable. [7]

Por ejemplo, considerando una intersección de dos calles que mostramos en la Figura 2, donde llamaremos a la calle horizontal, calle A, y a la vertical, calle B, un ciclo semafórico completo podría estar constituido por las siguientes fases:

- **Primera fase:** la calle A está en verde lo que permite el tránsito del tráfico, mientras que la calle B está en rojo y no lo permite.
- **Segunda fase:** la calle A cambia a ámbar lo que detiene el paso de los vehículos que aún no hayan entrado en la intersección y de esta forma se prepara para cambiar a rojo. La calle B permanece en rojo.
- **Tercera fase:** la calle B pasa a estar en verde y la calle A a rojo.
- **Cuarta fase:** la calle B cambia a ámbar, preparándose así para cambiar a rojo. Tras esta fase el ciclo semafórico habrá finalizado y volverá a repetirse la misma secuencia desde el inicio.

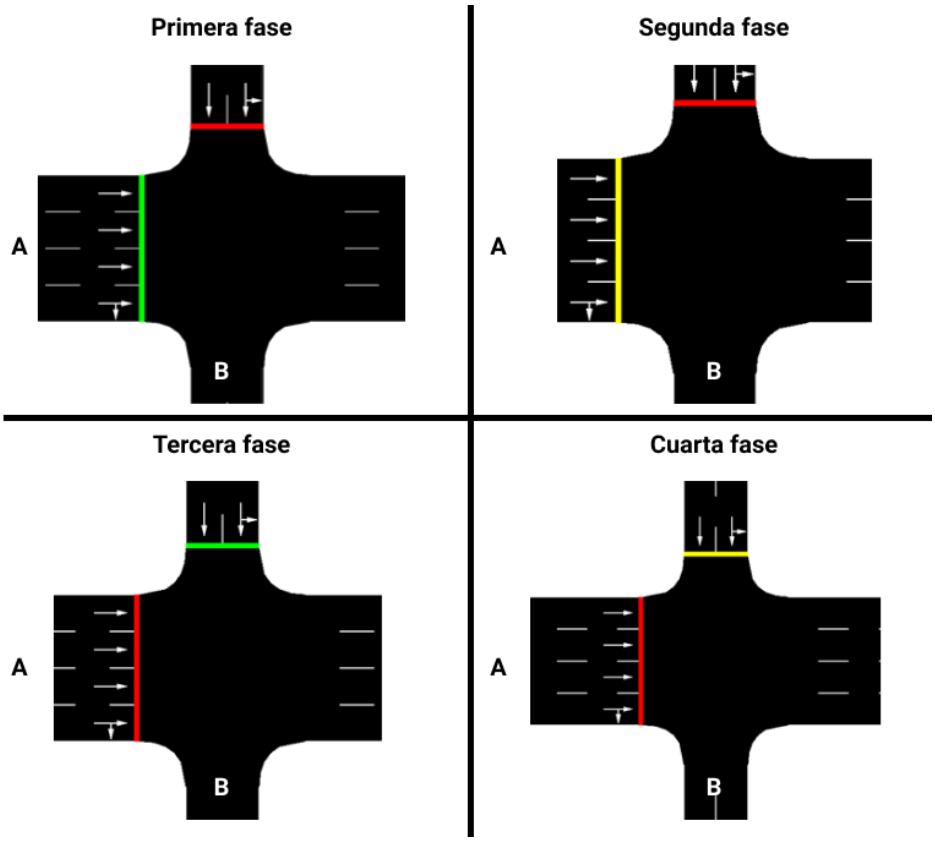


Figura 2 Ejemplo fases semafóricas

En la actualidad se utilizan mayoritariamente dos tipos de señales semafóricas:

- De **tiempo fijo**, este tipo de señales son las más simples [8]:
  - Cada fase tiene un tiempo fijado previamente.
  - Los ciclos se ejecutan siempre en un mismo orden y en bucle.
  - La duración de las fases y el ciclo se mantiene siempre igual.
  - No responden a la demanda del tráfico.
  - Mantenimiento e instalación de bajo coste.
- **Actuadas**, este tipo de señales son más sofisticadas [8]:
  - Ajustan los tiempos de fase y de ciclo en función del estado del tráfico.
  - Requieren de sensores externos que les informe del estado de las vías que forman la intersección.
  - Suelen priorizar la vía que tenga una mayor densidad de tráfico.
  - Mantenimiento e instalación de mayor coste y dificultad.

La mayoría de las señales utilizadas actualmente son las de tiempo fijo, ya que son más económicas y requieren de un menor mantenimiento. Las actuadas solo suelen utilizarse en zonas en las que haya un gran volumen de tráfico de forma regular y donde puedan generar mayor impacto en el flujo del tráfico, siendo así más rentables en este tipo de situaciones.

2.2 Inteligencia artificial en el transporte

Las tecnologías inteligentes están siendo cada vez más introducidas en diferentes aspectos de nuestras vidas y de nuestro entorno. Uno de ellos es el transporte, donde estas tecnologías buscan mejorar la movilidad de las personas y las mercancías, al mismo tiempo que incrementan la seguridad en las carreteras y reducen el impacto negativo en el medio ambiente. Podemos ver muchos ejemplos de uso de estas nuevas tecnologías en distintas aplicaciones en la actualidad. Desde la creación de vehículos inteligentes de conducción autónoma, a la gestión, creación y mantenimiento de infraestructuras del tráfico de maneras más innovadoras.

En este trabajo nos vamos a centrar en la optimización de las intersecciones controladas por luces de tráfico, que es un tema en el que actualmente ya se han realizado una gran cantidad de proyectos e investigaciones empleando distintas técnicas y enfoques. Por ejemplo, se han utilizado técnicas de aprendizaje automático para entrenar a redes neuronales, a elegir cuál es la fase semafórica más correcta para aplicar en una intersección en un momento determinado [9], o a controlar la duración de las fases [10]. Todo ello a partir de información sobre el estado actual de la intersección, que puede obtenerse mediante el uso de sensores sobre la vía. También se han utilizado técnicas metaheurísticas, donde según un artículo [11] la mayoría de estos proyectos trabajaron en la mejora del modelo de congestión alrededor de intersecciones y cruces, y la optimización en modelos de ciudades reales. Las técnicas metaheurísticas más utilizadas para llevar a cabo este proceso de optimización son, la optimización por enjambre de partículas, la optimización por colonia de hormigas y los algoritmos genéticos.

2.2.1 Técnicas metaheurísticas

Una metaheurística es un proceso que nos permite encontrar, generar o seleccionar una heurística (técnica para resolver un problema) concreta que nos proporcione una buena solución a un problema de optimización, especialmente en problemas de optimización combinatoria. En los últimos años, las metaheurísticas se están convirtiendo en alternativas exitosas a procedimientos clásicos de resolución de problemas de optimización que incluyen en su formulación matemática información incierta, estocástica o dinámica. Esto se debe a que estos métodos clásicos, pese a que garanticen encontrar la solución óptima, suelen tener un gran coste computacional cuando se enfrentan a problemas de alta complejidad y con un gran espacio de búsqueda. Mientras que, por otro lado, los métodos basados en metaheurísticas son capaces de encontrar soluciones óptimas, o lo suficientemente cercanas a las óptimas, en un tiempo mucho menor [12]. También se suelen aplicar cuando no existe otro método para obtener una solución a un problema [13]. Algunas de estas técnicas están inspiradas en procesos de la naturaleza, como la evolución o el comportamiento de ciertos animales.

2.2.1.1 Optimización por enjambre de partículas

La optimización por enjambre de partículas, también conocida por sus siglas en inglés PSO (Particle Swarm Optimization), es un método que optimiza un problema

tratando de mejorar una solución candidata de forma iterativa respecto a una medida de calidad determinada. A partir de una población de soluciones candidatas denominadas partículas, estas se mueven por el espacio de búsqueda de acuerdo con una fórmula matemática simple que afecta a la velocidad y posición de la partícula. Los movimientos de las partículas están influenciados por su posición local y por las mejores posiciones conocidas en el espacio de búsqueda. Este proceso suele guiar al enjambre a las mejores soluciones [14].

La optimización por enjambre de partículas está atribuida originalmente a los investigadores Kennedy, Eberhart y Shi, y principalmente tenía el propósito de simular el comportamiento social, intentando realizar una representación del movimiento de los organismos en un banco de peces o en una bandada de pájaros.

Esta técnica metaheurística ha sido utilizada varias veces para intentar encontrar programas de ciclos de luces de tráfico eficientes. Un ejemplo es este estudio [15] en el que se utiliza PSO para realizar este proceso de optimización con el objetivo de mejorar el flujo del tráfico sobre zonas urbanas de ciudades españolas.

2.2.1.2 Optimización por colonia de hormigas

La optimización por colina de hormigas, también conocida por sus siglas en inglés ACO (Ant Colony Optimization), es una técnica probabilística para resolver problemas computacionales y, normalmente, utilizada para buscar buenos caminos en problemas de grafos o de enrutamiento de vehículos. Este método está inspirado en el comportamiento de las hormigas y su comunicación basada en feromonas, que permite a las hormigas encontrar los caminos más cortos. Las hormigas dejan unas feromonas por donde pasan, estás feromonas pueden ser seguidas por el resto de las hormigas y desaparecen con el paso del tiempo. Cuanto más corto sea el camino mayor será la concentración de feromonas en él, por lo que será más utilizado por el resto de las hormigas de la colonia [16]. Las hormigas artificiales (que son los agentes individuales de este sistema multi agente utilizado en el algoritmo) se mueven por el espacio de parámetros que representa todas las soluciones, registran las posiciones y la calidad de sus soluciones. El camino elegido por las hormigas futuras estará determinado por la calidad de las soluciones anteriores, para que en iteraciones posteriores más hormigas sean capaces de encontrar mejores soluciones.

Este algoritmo fue propuesto inicialmente en 1992 por Marco Dorigo con el que intentaba buscar caminos óptimos en grafos basándose en el comportamiento de las hormigas [17].

La optimización por colonia de hormigas también ha sido uno de los métodos más utilizados para la optimización de señales de tráfico. Se ha observado en sus resultados ser más eficiente que el uso de semáforos actuados, especialmente cuando hay alta demanda de tráfico y para redes de carreteras complejas [18].

2.2.1.3 Algoritmos genéticos

Se realizará una explicación extensa sobre este método en el siguiente capítulo de la memoria: Fundamentos de los algoritmos genéticos. Aquí hablaremos de porqué

hemos escogido esta metodología para llevar a cabo el proceso de optimización en este trabajo. Todo se debe a una serie de razones, entre las que cabe destacar la gran flexibilidad que tiene esta técnica a la hora de resolver problemas. Otras de estas razones son:

- Pueden usarse en un gran número de problemas de optimización pertenecientes a distintos campos de la ciencia.
- Poseen un paralelismo inherente que les proporciona una muy buena habilidad de búsqueda y de gran rapidez. Esto produce muy buenos resultados en espacios de búsqueda grandes y en problemas de gran complejidad.
- Podemos ajustar sus parámetros para encontrar un equilibrio entre la exploración y la explotación de soluciones.
- Existe una enorme cantidad de documentación sobre su funcionamiento e implementación.

2.3 Introducción a SUMO (*Simulation of Urban MObility*)

SUMO es un simulador de tráfico gratuito y de código abierto que permite la simulación de diferentes medios de transporte de forma microscópica [19]. Esto quiere decir que simula de forma individual el movimiento de cada uno de los vehículos dentro del entorno, basándose en teorías de seguimiento de automóviles y de cambios de carril. Este tipo de modelo de simulación nos permite evaluar de forma efectiva situaciones con una congestión alta de tráfico, y analizar variables a nivel global e individual de cada vehículo que son características muy necesarias para la realización de este trabajo. Además, es capaz de trabajar con redes de gran tamaño y con un gran número de vehículos de forma eficiente.

Otras características por las que se ha decidido utilizar SUMO:

- La existencia de una interfaz de comunicación con la simulación llamada TraCI (Traffic Control Interface) que utiliza una arquitectura cliente/servidor, y permite consultar el valor de las variables de los distintos objetos simulados, así como manipular el comportamiento de la simulación y de estos objetos. El cliente se puede implementar en *Python* mediante el uso de las librerías de TraCI.
- La gran cantidad y calidad de la documentación de SUMO que se puede encontrar en su página web [20], que facilita el aprendizaje de su funcionamiento.
- Incluye una serie de herramientas que facilitan la creación, conversión y manipulación de redes de tráfico.
- La flexibilidad que ofrece a la hora de ejecutar una simulación. Nos permite modificar y elegir entre una serie de parámetros en función de los resultados que estemos buscando.
- La posibilidad de ejecutar las simulaciones en una interfaz gráfica (sumo-gui) para así poder visualizar el comportamiento de los vehículos dentro de la red de carreteras. Podemos ver esta interfaz en la Figura 3. Muy útil para resolver problemas derivados de la topología de la red y consultar parámetros de los distintos objetos de la simulación en cada instante de tiempo.

- Los ficheros de configuración de las simulaciones, de topología de las redes de carreteras y de rutas de los vehículos se encuentran en formato XML, por lo que existe una gran variedad de herramientas con las que podemos modificar y leer los datos de estos archivos con gran facilidad.

Las principales herramientas incluidas en el paquete de instalación de SUMO utilizadas durante la realización de este proyecto son:

- ***Netedit*:** es un editor de redes gráfico que se ha utilizado para solventar errores en las redes utilizadas en el proyecto y para eliminar vías innecesarias.
- ***RandomTrips*:** es una herramienta escrita en *Python* que genera archivos con rutas para vehículos dentro de una red determinada.
- ***NetConvert*:** permite la conversión de archivos de redes de carreteras en distintos formatos a uno que pueda ser utilizado por SUMO.

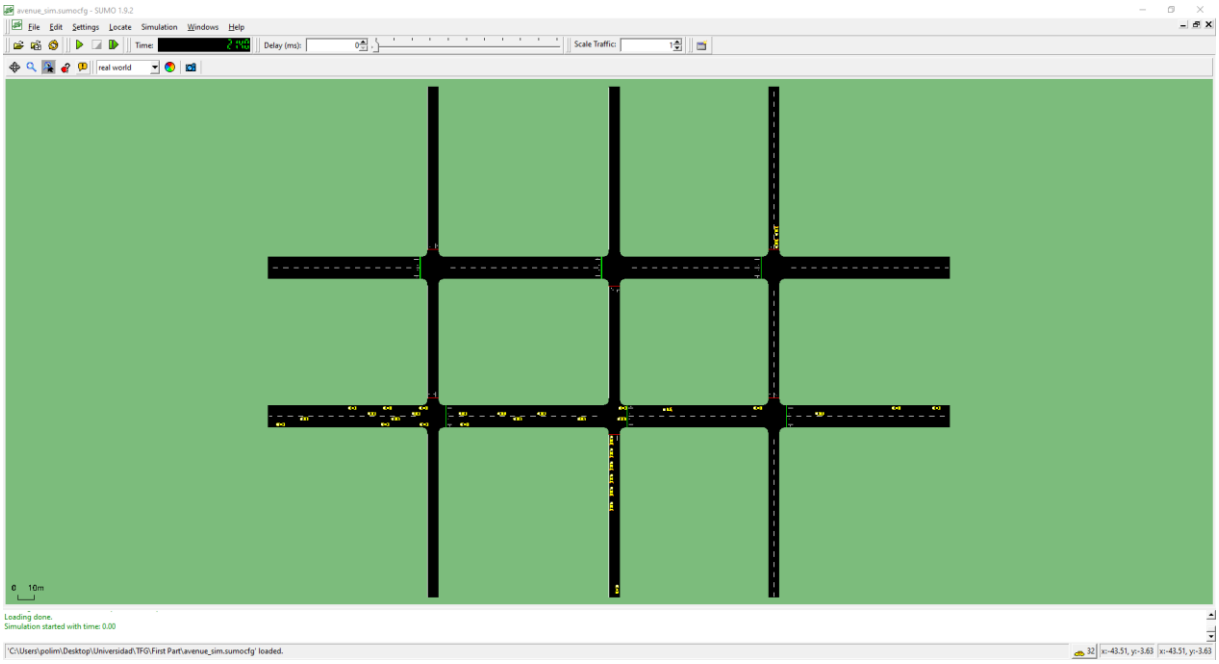


Figura 3 Interfaz gráfica de una simulación en sumo-gui

3. Fundamentos de los algoritmos genéticos

Un algoritmo genético (a partir de ahora AG) es una técnica de optimización inspirada en los principios de la evolución y la genética. Se utiliza para encontrar soluciones óptimas, o lo más cercanas posible a las óptimas, en problemas donde el espacio de búsqueda y el número de parámetros, a tener en cuenta, es muy grande. Todo ello de una forma más rápida y eficiente que si se utilizase un método más convencional [21]. Pertenecen al conjunto de los algoritmos evolutivos dentro de la familia de las técnicas metaheurísticas.

3.1 Terminología básica

En este apartado introduciremos algunos de los términos básicos sobre los AG, que serán necesarios conocer para entender el resto de la explicación y la memoria.

- **Población:** es un subconjunto de todas las soluciones posibles a un problema. Estas soluciones se encuentran codificadas siguiendo un genotipo concreto definido para el problema.
- **Cromosoma:** es cada una de las posibles soluciones codificadas al problema. Una población está formada por un conjunto de estos cromosomas.
- **Gen:** es la posición de un elemento de un cromosoma.
- **Alelo:** es el valor que toma un gen de un cromosoma en concreto.

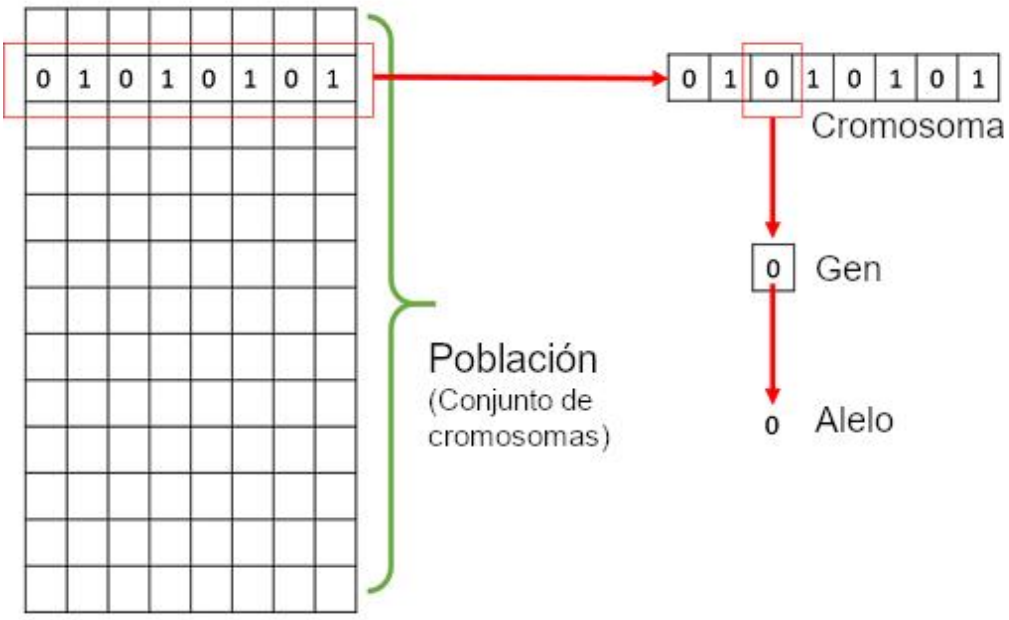


Figura 4 Diagrama representación de soluciones AG [21]

- **Genotipo:** es la forma en la que representamos las soluciones de un problema concreto, y que nos permite fácilmente su manipulación usando un ordenador. Este es el formato en el que se encontrará la población del problema dentro del AG y al decodificarla obtendremos su fenotipo. Por ejemplo, si quisiéramos representar la posición en la que se encuentra un objeto en un espacio

tridimensional, su genotipo podría ser un vector de tres números reales, donde cada gen representa la posición del objeto en uno de los distintos ejes que forman ese espacio. Esta representación sería fácil de utilizar para un ordenador.

- **Fenotipo:** es la forma de representar a la población como soluciones reales al problema. En el ejemplo anterior esto sería la posición en sí dentro del espacio tridimensional y que obtenemos tras descodificar un cromosoma que sigue ese genotipo concreto.
- **Operador genético:** se encargan de alterar la composición genética de la población y de generar sus descendientes.
- **Generación:** es cada una de las iteraciones que realizará nuestro algoritmo genético. Entre generaciones la población variará mediante el uso de operadores genéticos.
- **Función de fitness:** se encarga de evaluar las distintas soluciones al problema. Proporciona como resultado un valor que indica cómo de idónea es una solución respecto a un problema determinado.
- **Fitness:** es un valor que representa la idoneidad de una solución a un problema.

3.2 Representación genotípica de la población

Es una de las decisiones más importantes que hay que tener en cuenta a la hora de implementar un AG. Los cromosomas deberían poder contener la información de la solución que están representando. Para ello, debemos elegir o crear un genotipo concreto que codifique correctamente las soluciones a nuestro problema particular. El tipo de codificación utilizada varía en función del problema que se busque resolver y determina el tipo de operadores genéticos que se podrán utilizar. Si la representación elegida no es la más adecuada, el AG podría llegar a tener problemas de rendimiento.

Algunos de los tipos de representación más comúnmente utilizados son los siguientes:

- **Representación binaria:** es el tipo de representación más sencillo y usado de todos. El genotipo se representa como una cadena de bits.
- **Representación real:** en este tipo se utilizan cadenas de números. Los números pueden ser enteros o reales. Usado en problemas que requieran del uso de variables de tipo continuo.
- **Representación de orden:** los individuos se representan como permutaciones que representan un orden de elementos.

3.3 Estructura de un AG

En la Figura 5 podemos ver la estructura básica de un AG. Primero se genera la población inicial y se realiza su evaluación. Después, a partir de los resultados obtenidos en el proceso de evaluación, seleccionamos a los padres de entre la población que serán utilizados para crear a la descendencia que acabará formando parte de la población de la siguiente generación. La descendencia se genera mediante el uso de los operadores genéticos de cruce y mutación. Estos pasos se repetirán una y otra vez

hasta que se cumpla con alguno de los criterios de finalización del AG. A continuación, explicaremos en detalle cada uno de los pasos.

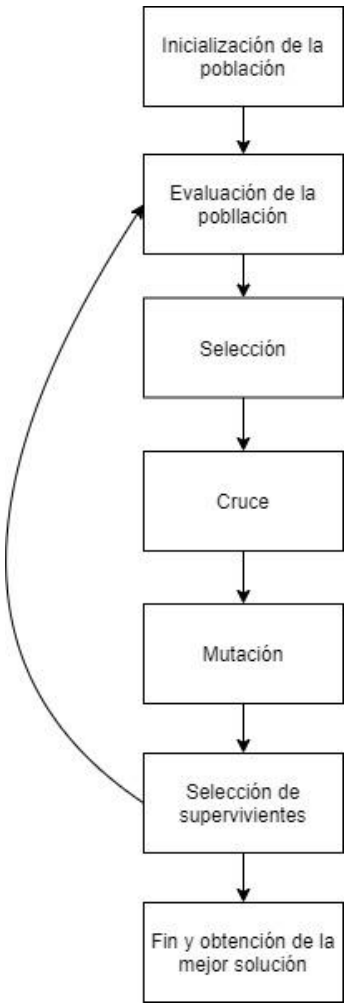


Figura 5 Estructura de un algoritmo genético

3.3.1 Inicialización de la población

Para poder empezar a ejecutar un AG necesitamos una población inicial de la cual podamos obtener la próxima generación. Existen dos métodos para inicializar la población:

- **Inicialización aleatoria:** poblar la población inicial con soluciones aleatorias pertenecientes al espacio de búsqueda.
- **Inicialización heurística:** poblar usando una heurística conocida del problema.

No se debería utilizar exclusivamente la inicialización heurística a la hora de crear una población inicial, ya que tener soluciones muy similares reduce la diversidad. La inicialización heurística solo tiene efectos en el valor del fitness inicial, la diversidad de las soluciones es la que finalmente nos lleva a las mejores soluciones. Por lo tanto,

cuando se utiliza la inicialización heurística esta solo suele generar unas pocas soluciones y las demás se obtienen de forma aleatoria

3.3.2 Evaluación de la población

Una vez se ha generado la población de una generación, debemos evaluar a sus individuos. Para ello, se necesita de una función de fitness que se encargue de valorar la idoneidad de cada una de las soluciones candidatas al problema. Una vez finalizado el proceso de evaluación, cada solución tendrá un valor de fitness asignado que representará la idoneidad de su solución. Dependiendo de la implementación realizada, un valor del fitness más alto puede llegar a representar mayor o menor idoneidad.

Este paso suele tener el mayor coste computacional dentro de un AG, por lo tanto, a la hora de elegir una función de fitness, no solo hay que tener en cuenta que evalúe de forma correcta siguiendo los objetivos de su diseñador, sino también que sea computacionalmente eficiente. La velocidad de ejecución del proceso de evaluación es muy importante, ya que debe repetirse muchas veces, por cada individuo de la población y en cada generación.

3.3.3 Selección

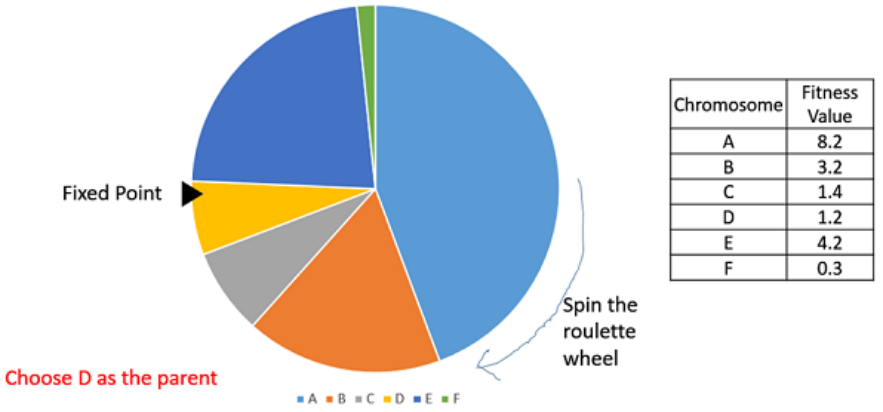
Una vez evaluados todos los individuos de la población, debemos realizar un proceso de selección teniendo en cuenta el valor de fitness que ha obtenido cada uno de ellos en la evaluación. En este proceso se elegirán a los individuos padres que se reproducirán para formar la descendencia de la siguiente generación.

Se ha de tener cuidado a la hora de elegir una metodología de selección, y evitar dar demasiada preferencia de selección a una única solución que acabaría dominando toda la población, produciendo así una gran pérdida de la diversidad. A este fenómeno se le conoce como convergencia prematura, y como ya hemos dicho anteriormente, es de suma importancia mantener la diversidad poblacional para alcanzar buenas soluciones. Los métodos más utilizados de selección son, la selección por ruleta, la selección por rango y la selección por torneo.

3.3.3.1 Selección por ruleta

En este método de selección cada individuo tiene una probabilidad de convertirse en padre proporcional a su valor de fitness. Por lo tanto, los individuos más idóneos tienen más posibilidades de propagar sus características a la siguiente generación.

Consideramos una rueda circular dividida en el mismo número de trozos como individuos hay en la población, donde cada trozo tiene un tamaño proporcional al fitness de su solución. Elegimos un punto fijo de la circunferencia de la rueda y la hacemos girar. El cromosoma asociado a la región que caiga en este punto es elegido como uno de los padres. Podemos ver un ejemplo de esto en la Figura 6. Este proceso se repite tantas veces como padres necesitemos. Normalmente se utiliza esta técnica de selección cuando trabajamos con valores de fitness positivos y estamos buscando incrementarlo.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Figura 6 Ejemplo gráfico selección por ruleta [21]

3.3.3.2 Selección por rango

Este método funciona para valores negativos de fitness y se utiliza principalmente cuando los valores de fitness de los individuos de la población son muy cercanos entre ellos. Por ejemplo, si utilizásemos selección por ruleta con valores de fitness de los individuos próximos, todos tendrían una posibilidad muy parecida de ser seleccionados, por lo que ya no estaríamos dando más importancia al individuo con mejor fitness. En este método también se utiliza una ruleta, en la que el tamaño de las regiones dependerá del número de individuos y no de su fitness. A cada individuo se le asigna un rango, cuanto mejor sea el valor del fitness de un individuo, mayor será su rango asignándole una región de la ruleta en función de este rango. De esta forma, los individuos con mejor fitness tendrán una región mayor y habrá más probabilidades de que sean seleccionados.

También se utiliza para mantener la diversidad de la población. Si una de las soluciones tiene un valor de fitness muy grande en comparación con las demás, sus probabilidades de ser elegida disminuyen, mientras que las de las demás aumentan. De esta forma se evita que una sola solución pueble enteramente la siguiente generación. Podemos ver un ejemplo de esta situación en la Figura 7. En la ruleta de la izquierda una de las soluciones acapara gran parte de la misma, en la de la derecha, se le asigna una sección en base al número de soluciones y su rango.

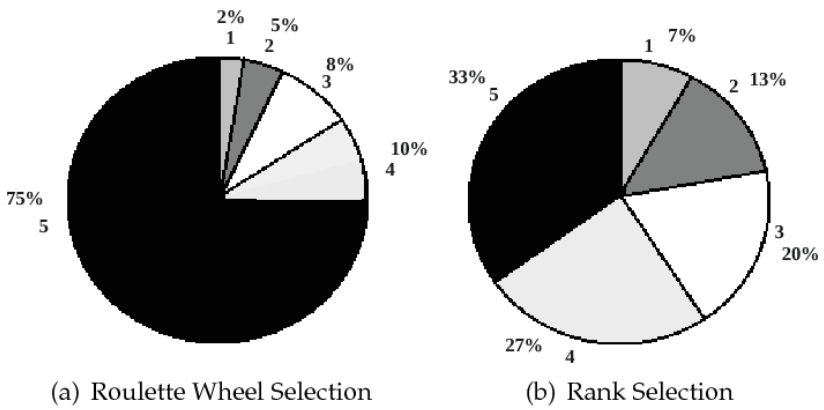


Figura 7 Selección por ruleta vs Selección por rango [22]

3.3.3.3 Selección por torneo

En este tipo de selección se realizan una serie de “torneos”. En ellos escogemos un número determinado de distintos individuos de la población al azar. El individuo con el mejor valor de fitness de todos los seleccionados es el ganador de este y, por lo tanto, acaba siendo padre. Este método de selección es muy flexible, se puede utilizar para valores de fitness positivos, negativos y tanto si buscamos aumentar o disminuir su valor. Se muestra un ejemplo de este proceso en la Figura 8.

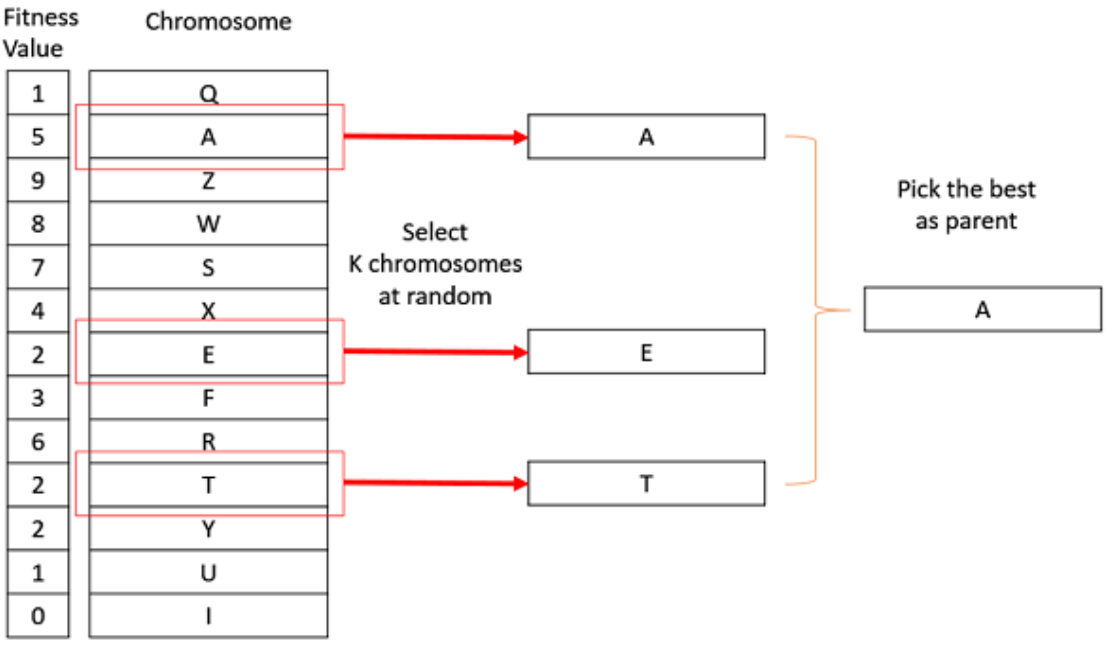


Figura 8 Ejemplo selección por torneo [21]

3.3.4 Cruce

Tras seleccionar a los que serán los cromosomas padre de la generación actual, se procede a realizar un proceso de cruce entre ellos que simula la recombinación genética sexual. Las operaciones de cruce producen un intercambio del material genético entre dos cromosomas y forman la descendencia. El objetivo es que la operación de cruce junte los mejores atributos de cada uno de los padres para que así la descendencia generada obtenga un valor de fitness mejor. A esta operación se le asigna una probabilidad (normalmente alta) que controla la proporción de las parejas que serán elegidas para reproducirse. El tipo de operación de cruce a realizar dependerá del problema y la representación elegida.

Algunos de los operadores de cruce más utilizados son:

- **Cruce de un punto:** este método genera como resultado dos nuevos cromosomas. Primero se elige un punto de cruce aleatorio en los cromosomas de los padres para crear un nuevo cromosoma que está formado, por una parte, por los genes que encontramos en uno de los cromosomas padre desde el inicio

hasta el punto de cruce, y, por otra parte, de los genes encontrados desde el punto de cruce al final del otro padre. El segundo cromosoma generado será la combinación contraria de las partes. Podemos observar un ejemplo en la Figura 9, donde la línea roja representa el punto de cruce seleccionado.

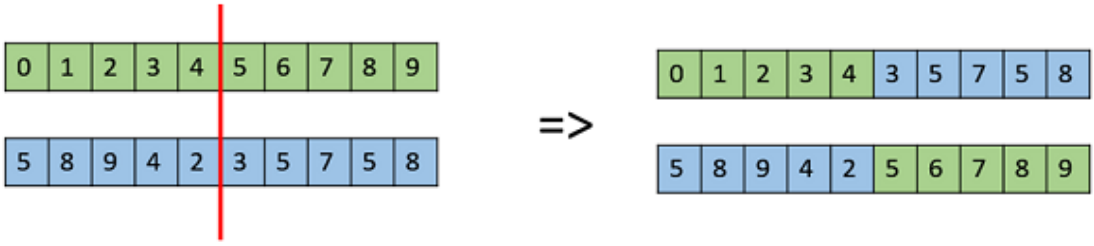


Figura 9 Ejemplo cruce de un punto [21]

- **Cruce multipunto:** es el mismo concepto que el método anterior (cruce de un punto), pero los puntos de cruce elegidos forman segmentos en los cromosomas padre que son intercalados de forma alterna para formar a los descendientes. Podemos observar un ejemplo en la Figura 10, en la que se realiza un cruce de dos puntos.

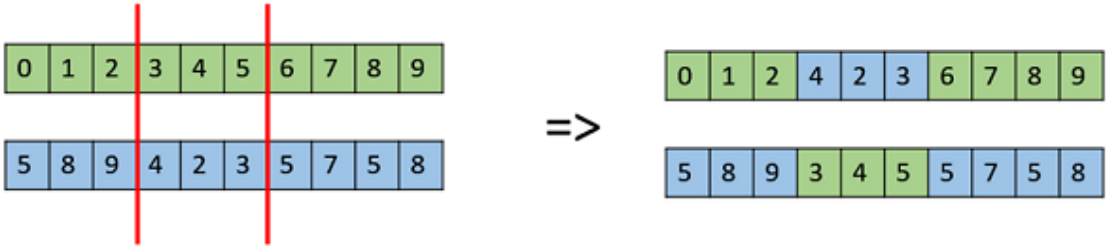


Figura 10 Ejemplo cruce multipunto (Dos puntos) [21]

3.3.5 Mutación

Tras el cruce tiene lugar el proceso de mutación, que se encarga de añadir y de mantener la diversidad de la población. Para ello, se realizan modificaciones sobre algunos genes de los cromosomas de forma aleatoria. La probabilidad de que ocurran este tipo de modificaciones es muy baja, si fuera alta, reduciríamos la eficacia del AG, ya que estaríamos realizando una simple búsqueda aleatoria.

Al igual que los operadores de cruce, la elección del operador de mutación utilizado dependerá de la representación de los cromosomas y del problema. Por ejemplo, si las soluciones estuvieran representadas de forma binaria, la mutación consistiría en invertir el valor de un gen específico del cromosoma, como se pude observar en la Figura 11. En caso de que fuera numérica, podríamos cambiar el valor de un gen por otro aleatorio (dentro de los valores permitidos), intercambiar valores de lugar o aplicarle una operación matemática aleatoria.

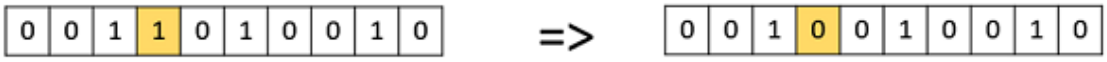


Figura 11 Ejemplo mutación en una representación binaria [21]

3.3.6 Selección de supervivientes

En esta fase se elige qué individuos pueden permanecer en la siguiente generación y cuáles deben expulsarse. Es importante tener una buena política de selección que se encargue, tanto de mantener a los individuos mejor adaptados, como de mantener la diversidad. Tras este paso habremos obtenido la población que formará la siguiente generación y se repetirán todos los proceso anteriores, en caso de que no se cumpla con alguno de los criterios de finalización.

Algunos AG utilizan una aproximación elitista en la que a los individuos mejor adaptados de cada generación se les garantiza tener un lugar en la siguiente. Además, no se les realizará ningún proceso de mutación que podría empeorar la solución. Sin embargo, sí que podrán ser elegidos padres.

3.3.7 Terminación

Normalmente no conocemos cuál es la mejor solución a un problema de optimización, así que es necesario establecer un criterio de terminación que indique al AG cuándo debe parar. Inicialmente los AG progresan de forma rápida, obteniendo mejores soluciones cada poco tiempo, sin embargo, cuanto más nos acercamos a la solución óptima, el progreso se ralentiza y las mejoras en las soluciones se vuelven más pequeñas. No merece el esfuerzo gastar una gran cantidad de recursos en una pequeña mejora. Algunos de los criterios más utilizados son:

- Detener la ejecución cuando no ha habido una mejora lo suficientemente grande en un determinado periodo de tiempo o de generaciones.
- Llegar a un número máximo de generaciones.
- En caso de que sí conozcamos el resultado final, acercarse mucho a este o llegar a obtenerlo.

4. Implementación de la solución

En este capítulo empezaremos viendo cómo están codificados los semáforos dentro de un archivo de red de carreteras compatible con SUMO y cómo funcionan. Después hablaremos de cómo se han obtenido los datos de entrada necesarios para ejecutar el algoritmo genético, estos son, los mapas de carreteras, las rutas de los vehículos y los archivos de configuración inicial. Finalmente se explicará cómo se ha llevado a cabo la implementación del algoritmo genético, cómo se ha representado a la población y los operadores genéticos utilizados.

4.1 Semáforos en SUMO

Un archivo de red de SUMO contiene la información para representar el mapa por el que circulan los vehículos de la simulación. Esta red es un grafo dirigido formado por las carreteras (aristas) y las intersecciones (nodos). Cada una de las intersecciones que encontramos en una red, puede estar controlada o no por un semáforo. En caso de que lo esté, el conjunto semafórico o luz de tráfico, tendrá asociado un programa de funcionamiento. Estos programas indican el comportamiento que deben seguir en cada momento, durante la simulación, cada uno de los distintos semáforos pertenecientes al conjunto semafórico de la intersección. Todos estos programas se encuentran guardados como elementos XML dentro del archivo que contiene toda la información de la red. Tienen el nombre de *tlLogic* y podemos observar su estructura en la Figura 12.

La estructura de un programa está formada por un elemento padre (*tlLogic*) y un conjunto de subelementos hijos (phase), que representan cada una de las fases del ciclo semafórico asociado al controlador. Cada uno de estos elementos tiene una serie de atributos. Describiremos los más importantes a continuación.

```
<tlLogic id="TL1" type="static" programID="0" offset="0">
    <phase duration="42" state="GGGGrrrrr"/>
    <phase duration="3"  state="yyyyrrrrr"/>
    <phase duration="42" state="rrrrGGGGG"/>
    <phase duration="3"  state="rrrryyyyy"/>
</tlLogic>
```

Figura 12 Estructura XML tlLogic

Atributos de tlLogic

- **id:** es el identificador de la luz de tráfico a la que está asociado el programa.
- **type:** es el tipo de la luz de tráfico. En este trabajo solo vamos a utilizar el tipo “static”, que hace que el semáforo se comporte como uno de tiempo fijo. Como ya hemos explicado antes, este es el comportamiento que tendría un semáforo estándar en la realidad que no se adapta a las circunstancias del tráfico, sino en el que se repite el mismo programa una y otra vez y en el que las fases tienen un tiempo de duración preconfigurado.
- **programID:** es el identificador del programa dentro de los distintos programas que pueden estar asignados a una misma luz de tráfico.

Atributos de phase

- **duration:** indica la duración de la fase en segundos.
- **state:** configuración de los estados de las luces de tráfico para esta fase. Usando como ejemplo la cadena de la primera fase para el controlador de la Figura 12 “GGGGrrrrr”, cada uno de los caracteres controla el estado o color de una de las señales asociadas a la luz de tráfico. Hay que tener en cuenta, que un mismo carril puede contener varias señales (por ejemplo, una para vehículos que giran a la izquierda y otra para los que siguen recto). Esto quiere decir que las señales no controlan carriles, sino “enlaces”. El valor “G” indica que los vehículos pueden circular por la intersección y, el valor “r”, que deben parar. En otras fases de la Figura 12 podemos ver el uso del valor “y” en su cadena state. Este valor indica que se va a cambiar a rojo y que los vehículos que aún no hayan entrado en la intersección deben parar. Podemos ver cómo se traduciría el state de esta primera fase (“GGGGrrrrr”) dentro de la simulación en la Figura 13, donde cuatro de los “enlaces” controlados por los semáforos están en verde (“G”) y los otros cinco en rojo (“r”).

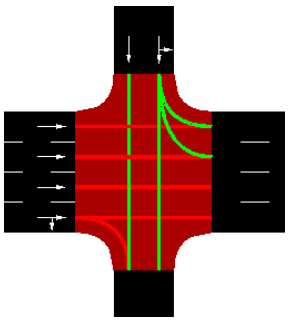


Figura 13 Primera fase TL1

Durante la ejecución de la simulación, se ejecutará cada programa asociado a una luz de tráfico en bucle, pasando por todas sus fases y estando en cada una de ellas el tiempo que tenga asignado en su duración hasta que finalice la simulación.

4.1.1 Algoritmo SCPG de SUMO

Los programas para las luces de tráfico de una red de carreteras son creados de forma automática por el algoritmo SCPG (SUMO Cycle Program Generator), encargado de asignar las duraciones de las fases de las luces de tráfico, siguiendo una serie de normas:

- La duración de los ciclos semafóricos es de noventa segundos, por defecto, por lo tanto, la suma de la duración de todas las fases que forman un ciclo es de noventa segundos.
- La duración de las fases verdes se calcula teniendo en cuenta, el número de vías de entrada y de salida a la intersección, el tipo de cada vía y las velocidades de frenado [23] [15].
- Después de cada fase verde hay una fase amarilla con la duración mínima necesaria para que no haya colisiones en la intersección. En ella se da el tiempo

necesario para que los vehículos que estén dentro de la intersección puedan salir [24].

Como se puede deducir, el algoritmo no tiene en cuenta muchos factores a la hora de asignar los tiempos a las fases en verde, como puede ser, la relación que hay entre los distintos semáforos. Esto suele llevar a una circulación del tráfico muy ineficiente, especialmente cuando la demanda de tráfico es alta. Intentaremos realizar una asignación de tiempos más inteligente y eficiente utilizando un algoritmo genético.

4.2 Obtención de los datos de entrada

Para poder ejecutar una simulación en SUMO, necesitamos tres componentes principales: un archivo con el mapa de la red sobre el que se van a mover los vehículos, un archivo en el que se encuentren las rutas que van a seguir estos por el mapa y un archivo de configuración necesario para iniciar la simulación. En este último indicamos, el mapa, las rutas a utilizar y otros parámetros que pueden afectar al comportamiento de la simulación.

4.2.1 Creación de mapas de redes de carreteras

En este trabajo se quiere demostrar el potencial de optimización que tienen las zonas urbanas en la actualidad, por lo tanto, se van a utilizar mapas urbanos reales. Para ello, vamos a hacer uso de *OpenStreetMap* [25], que es un mapa mundial online creado de forma colaborativa y que nos permite exportar zonas específicas del mapa. Los mapas contienen información sobre la topología de las calles, carreteras y las posiciones, de no todos, pero sí de la mayoría de los semáforos. Podemos ver la interfaz web de esta herramienta en la Figura 14.

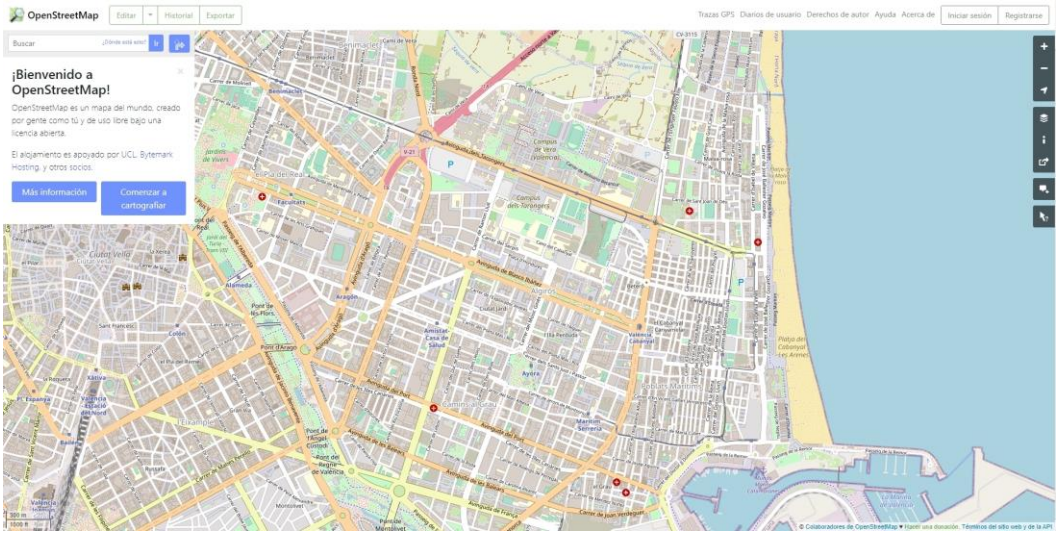


Figura 14 Interfaz web OpenStreetMap

Los mapas exportados se encuentran en formato OSM y pueden ser transformados a redes compatibles con SUMO utilizando la herramienta *NetConvert*, junto con una serie de parámetros. Con estos parámetros le indicaremos a la herramienta que solo queremos crear una red con los segmentos por los que puedan circular los vehículos

particulares, ya que en nuestras simulaciones no tendremos en cuenta a los peatones ni al transporte público. *NetConvert* también se encargará de crear los programas de todas las intersecciones controladas por semáforos de nuestro mapa haciendo uso de SCPG.

Después, se ha utilizado *Netedit* para corregir algunos de los errores generados durante el proceso de conversión y para eliminar aristas innecesarias. Se han escogido dos mapas urbanos distintos para así demostrar que la herramienta funciona para cualquier tipo de red. Estos mapas son, un fragmento de Barcelona, ya que es una de las ciudades más congestionadas de toda España y un fragmento de Valencia, ya que se trata de una ciudad con la que están más familiarizados los posibles lectores de este trabajo.

La zona escogida de Barcelona es la que se encuentra alrededor de la Plaza de España. En esta zona hay varias avenidas importantes, carreteras, calles y una rotonda central. Podemos ver el aspecto final de esta primera red y la posición de las 69 intersecciones controladas por luces de tráfico en la Figura 15.

Figura 15 Red de Barcelona

La zona escogida de Valencia es algo más extensa y mantiene un número similar de luces de tráfico con respecto a la zona de Barcelona. La zona escogida está formada por gran parte de la zona universitaria y de sus barrios adyacentes, y en el centro del mapa se encuentra la avenida Blasco Ibáñez. Podemos ver la red y la posición de las 65 intersecciones controladas por luces de tráfico en la Figura 16.

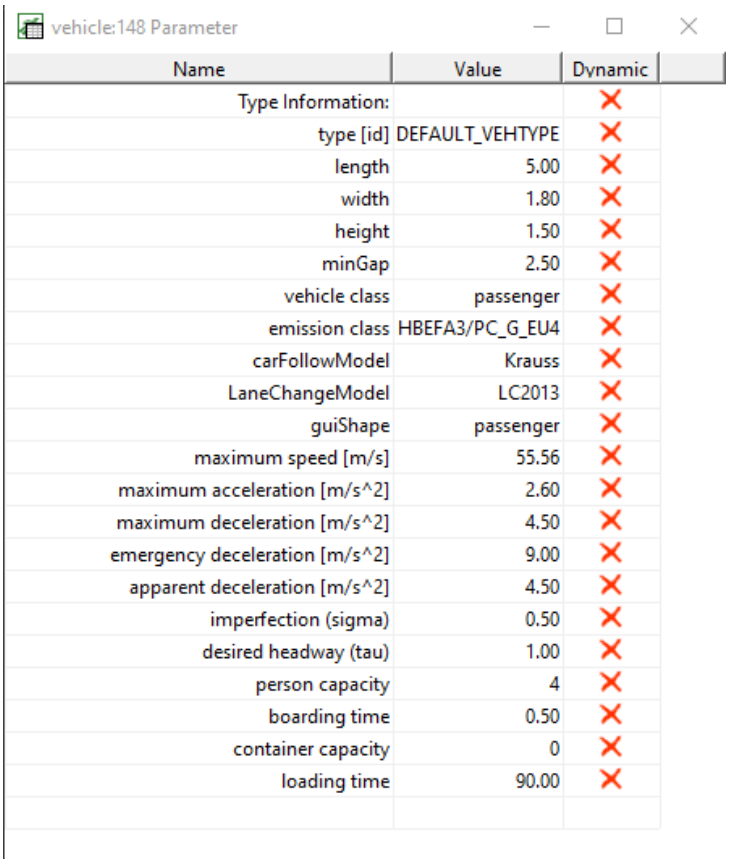
Figura 16 Red de Valencia

4.2.2 Generación de rutas de los vehículos

Ahora que ya hemos elegido y creado las redes que vamos a utilizar, podemos generar rutas para cada una de ellas. Hay varias formas para crear demanda de tráfico realista en SUMO. Se pueden crear a mano, utilizando matrices de origen-destino o utilizando estadísticas poblacionales. El problema es que, para usar la mayoría de estos métodos, necesitamos muchos datos reales a los que no tenemos acceso, bien porque no son públicos, o directamente no existen para zonas tan específicas como las que hemos elegido. Esto hace que el mejor método para crear las rutas sea la herramienta de *RandomTrips*. Esta herramienta nos permite generar rutas de forma rápida y para cualquier red de carreteras. También nos da la opción de ajustar la frecuencia de aparición de los vehículos y así poder simular situaciones de alta o baja demanda de

tráfico. El único problema que tiene usar esta solución es que las rutas creadas son aleatorias y los resultados pueden no asemejarse a la realidad. Sin embargo, si la herramienta de optimización funciona con rutas aleatorias, también lo hará con rutas reales.

Para cada mapa generaremos dos archivos de rutas, uno con baja demanda de tráfico y otro con alta. De esta forma podremos ver cómo se comporta el AG en cada situación. Los archivos de rutas están en formato XML y cada uno de ellos está formado por un conjunto de rutas distintas, donde cada una tiene una serie de atributos que indican en qué instante de tiempo se inicia, su punto de origen, su punto de destino y el tipo de vehículo que la realiza. En las simulaciones solo utilizamos un tipo de vehículo, uno estándar de pasajeros, de gasolina, constituido por las propiedades de la Figura 17.



Name	Value	Dynamic
Type Information:		✗
type [id]	DEFAULT_VEHTYPE	✗
length	5.00	✗
width	1.80	✗
height	1.50	✗
minGap	2.50	✗
vehicle class	passenger	✗
emission class	HBEFA3/PC_G_EU4	✗
carFollowModel	Krauss	✗
LaneChangeModel	LC2013	✗
guiShape	passenger	✗
maximum speed [m/s]	55.56	✗
maximum acceleration [m/s^2]	2.60	✗
maximum deceleration [m/s^2]	4.50	✗
emergency deceleration [m/s^2]	9.00	✗
apparent deceleration [m/s^2]	4.50	✗
imperfection (sigma)	0.50	✗
desired headway (tau)	1.00	✗
person capacity	4	✗
boarding time	0.50	✗
container capacity	0	✗
loading time	90.00	✗

Figura 17 Propiedades vehículo estándar

4.2.3 Archivo de configuración

Los archivos de configuración utilizados para ejecutar las simulaciones son muy sencillos, ya que podemos configurar más parámetros a la hora de ejecutar SUMO desde la línea de comandos. Como, por ejemplo, la duración o número de pasos de la simulación, o la retroalimentación en la consola. Estos archivos tienen la estructura que podemos ver en la Figura 18.

```
<configuration>
    <input>
        <net-file value="barcelona.net.xml"/>
        <route-files value="trips.trips.xml"/>
    </input>
    <processing>
     <time-to-teleport value="-1" />
    </processing>
</configuration>
```

Figura 18 Estructura archivo configuración SUMO

En el apartado de net-file se encuentra el nombre del archivo de la red que queremos utilizar, y en routes-files el nombre del archivo de las rutas generadas a partir de dicha red y para funcionar en ella. Con el parámetro *time-to-teleport* con un valor de -1 evitamos que los vehículos se teletransporten al encontrarse parados durante mucho tiempo, como puede ser por causa de un atasco. De esta forma obtenemos resultados más reales y penalizamos más a combinaciones de programas ineficientes que generen este tipo de situaciones indeseadas.

4.3 Implementación del Algoritmo Genético

Ahora que ya disponemos de todo lo necesario para realizar las simulaciones, necesitamos implementar un AG que se encargue de optimizar los tiempos de duración de las fases de los ciclos semafóricos de una red, y sea capaz de ejecutar las simulaciones que le permitirán evaluar cada una de las soluciones propuestas dentro de la población. La implementación del AG se ha realizado en *Python* 3.9 junto al uso de las librerías de TraCI de SUMO, que permiten la ejecución, manipulación y extracción de datos de las simulaciones mediante código. El AG seguirá la estructura mostrada en la Figura 5.

4.3.1 Representación de la población y codificación

El primer paso para implementar un AG es elegir cómo vamos a realizar la representación de las soluciones. Como el parámetro que queremos optimizar es la duración de las fases que se encuentran dentro de los programas de los semáforos del archivo XML de una red, como el de la Figura 12, vamos a utilizar una codificación de cadenas de números enteros. Cada gen de la cadena representará la duración que tendrá una fase verde de un programa de la red. No queremos modificar las fases amarillas, ya que estas fases ya tienen la duración mínima necesaria para evitar colisiones dentro de la intersección.

Podemos ver un ejemplo de cómo se encontrarían codificadas (izquierda) y decodificadas (derecha) las soluciones en la Figura 19, respecto a dos programas de luces de tráfico de una red.

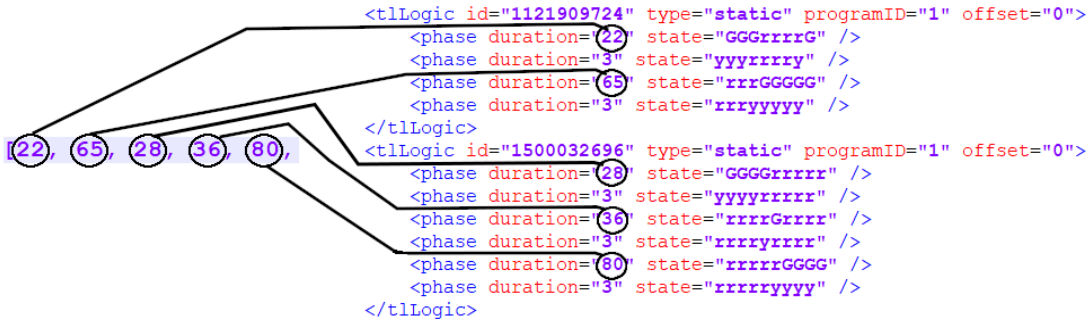


Figura 19 Ejemplo método codificación

Para evitar que el espacio de búsqueda sea muy grande, limitaremos los valores que pueden tomar cada uno de los genes de los cromosomas, a un intervalo de entre 10 y 80 segundos, y así también evitar tiempos de fases muy cortos o excesivamente grandes. La longitud de los cromosomas corresponderá al número de fases verdes que haya en total dentro de un archivo de red XML.

La población inicial del algoritmo se generará de forma completamente aleatoria. Se creará un vector de números enteros pertenecientes al intervalo elegido anteriormente por cada individuo de la población.

4.3.2 Proceso de evaluación

Todos los individuos de cada generación deben pasar por un proceso de evaluación para determinar cómo de apta es la solución que proponen. Los datos necesarios para la evaluación saldrán de los resultados obtenidos tras cada simulación.

Cada individuo propone un conjunto de programas semafóricos distintos para una red, por lo que en cada simulación que ejecutemos, tenemos que sustituir los programas predeterminados por los nuevos. Esto se puede lograr a través de un *additional-file*. En cada generación se creará un *additional-file* por cada individuo de la población, que contendrá los programas de las luces de tráfico, que propone decodificadas (fenotipo), y se ejecutarán las simulaciones de SUMO. En cada paso de simulación recogeremos el valor de una serie de variables, que al finalizar se utilizarán para calcular el fitness del individuo mediante la función de fitness. En SUMO cada paso de simulación equivale a un segundo. Se ha elegido un número de pasos límite de 750 para todas las simulaciones, de esta forma reducimos el tiempo que tarda en ejecutarse cada una de ellas y, por lo tanto, en procesar cada generación. Además, es una cantidad de pasos más que suficiente para observar los resultados que obtenemos tras modificar los ciclos semafóricos.

Los valores recogidos en cada simulación, hasta que esta llegue al número de pasos límite establecidos y utilizados en la función de fitness, son:

- **Vehículos en destino (Vd):** número de vehículos que han conseguido finalizar su ruta completa (han llegado a su destino).
- **Vehículos totales (Vt):** el número total de vehículos que han iniciado su ruta.

- **Consumo de combustible (Fc):** sumatorio de la cantidad de combustible utilizada por cada uno de los vehículos (combustible utilizado total).
- **Tiempo de espera (Wt):** sumatorio del tiempo que ha pasado cada vehículo en parada debido a un atasco o a una luz de tráfico.
- **Mayor tiempo de espera (BWt):** la cantidad de tiempo que ha esperado el vehículo que más tiempo ha esperado de toda la simulación.

Entre los valores anteriores no se han incluido las emisiones de CO2 de los vehículos, pese a que fuese uno de los parámetros a reducir. La razón es que se ha observado que, en SUMO, la cantidad de emisiones de este y otros gases, es proporcional a la cantidad de combustible utilizada en cada instante de tiempo, como se puede observar en la Figura 20 (uso de combustible en la gráfica izquierda y emisiones de CO2 a la derecha). Por lo tanto, reduciendo el uso de combustible, también reducimos las emisiones de CO2 en la misma proporción. Además, cuanto menor sea la cantidad de variables a recoger en cada paso de simulación, el tiempo de ejecución será mucho menor.

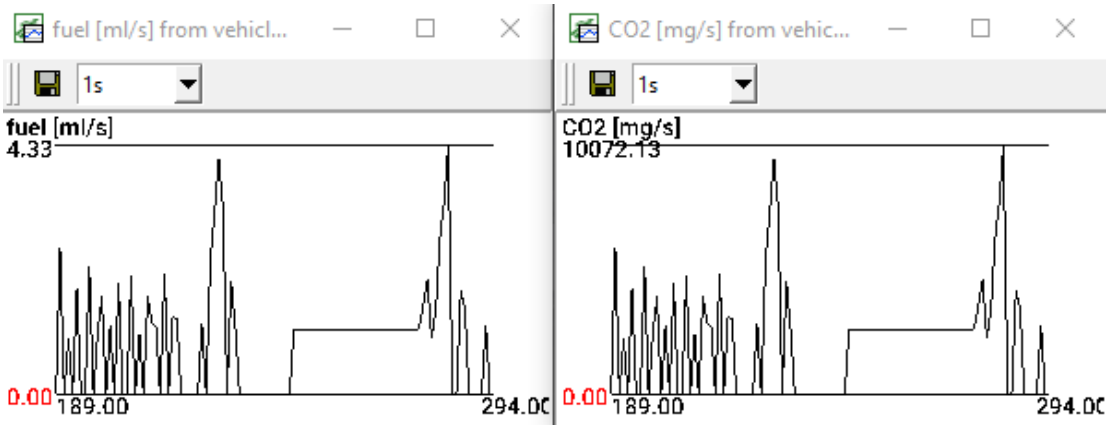


Figura 20 Consumo de combustible vs emisiones de CO2 en SUMO

La función de fitness utilizada es la siguiente:

$$Fitness = \alpha \frac{Vt}{Vd} + \frac{(\beta Fc + \gamma Wt)}{Vt} + \delta BWt$$

Donde α, β, γ y δ son ponderaciones con las que podemos definir la importancia que le damos a cada uno de los parámetros. Si nos fijamos en la fórmula, podemos observar que, cuanto menor sea el valor de los parámetros de tiempo de espera (Wt) y consumo de combustible (Fc) (que son los que estamos buscando minimizar), y mayor sea el número de vehículos que completan su ruta (Vd) (que estamos buscando maximizar), menor será el valor obtenido como resultado de esta fórmula. Por lo tanto, cuanto más pequeño sea el valor del fitness de un individuo de la población, más apta será su solución al problema.

4.3.3 Operadores genéticos utilizados

Una vez evaluados todos los individuos, procederemos a realizar el proceso de selección para elegir a los padres y generar la descendencia que poblará la próxima generación a través de los mecanismos de cruce y mutación.

4.3.3.1 Selección

Se ha decidido utilizar dos métodos de selección para este AG. Uno de ellos es el de selección por torneo, que es el encargado de elegir a los padres. Se ha optado por él, ya que nos permite controlar la presión de selección. Cuanto mayor sea el número de individuos que participan en cada torneo, menores serán las probabilidades de elegir a los individuos más débiles y viceversa. Es importante elegir un buen tamaño de torneo para evitar una pérdida rápida de la diversidad entre generaciones.

El otro método de selección elegido es el de selección elitista. Su función es la de asegurar que los mejores individuos de la generación actual formen parte de la población de la siguiente generación, aunque no hayan sido seleccionados como padres. De esta forma, evitamos perder las mejores soluciones de cada generación en caso de que no hayan sido elegidas para reproducirse.

4.3.3.2 Cruce y mutación

Tras seleccionar a los padres, primero se realiza un proceso de cruce o reproducción. El método seleccionado es el de cruce de dos puntos mostrado en la Figura 10, que permitirá un intercambio de información genética entre los padres para producir nuevos individuos, compatible con la representación que hemos elegido.

Después, estos nuevos individuos pasarán al proceso de mutación. Durante este proceso, cada gen de cada cromosoma tendrá una pequeña probabilidad de modificar su valor por uno distinto aleatorio dentro del intervalo seleccionado (10 - 80), al igual que ocurre con la generación de la población inicial. Tras este paso habremos generado la descendencia que, junto a los individuos seleccionados de forma elitista, formarán la siguiente generación.

4.3.4 Criterio de finalización

Tras ejecutar la evaluación de la población, se comprobará si se cumple con el criterio de finalización. El criterio de finalización utilizado en nuestro AG observa tras cada generación la mejora del fitness respecto a la generación anterior. Si tras varias generaciones esta mejora no existe o es muy pequeña, por ejemplo, solo se ha reducido en unos milisegundos el tiempo medio de espera o solo se ha conseguido que un par de vehículos más finalicen su ruta, el AG daría por terminada la ejecución y devolvería la mejor solución.

5. Pruebas y resultados

Para empezar, en este capítulo realizaremos una serie de pruebas que nos permitirán obtener los mejores valores posibles para algunos de los parámetros que afectan al rendimiento de nuestro AG. Buscamos obtener unos valores que nos permitan encontrar buenas soluciones en una cantidad de tiempo razonable.

Después, realizaremos una evaluación final en la que compararemos los resultados obtenidos de las simulaciones para los dos mapas seleccionados anteriormente, y para las distintas intensidades de demanda de tráfico, entre los programas asignados por defecto del algoritmo SCPG de SUMO y los generados por el AG.

5.1 Análisis de parámetros del AG

No existe una regla que nos permita conocer cuáles son los mejores valores para los distintos parámetros que controlan el comportamiento de un AG, ya que estos dependen de muchos factores, entre ellos, el problema que estamos intentando resolver, los operadores de mutación elegidos, la longitud de los cromosomas, etcétera. Existen algunos métodos automatizados de calibración para los parámetros de un AG basados en metaheurísticas, aunque, también es posible llevar a cabo un análisis de forma manual en el que observemos cómo los cambios en los distintos parámetros afectan al desempeño del AG.

Para este trabajo se ha elegido realizar un análisis de forma manual. Las pruebas se van a realizar sobre el mapa de Barcelona con una demanda de tráfico baja. Por cada valor a probar sobre cada parámetro vamos a ejecutar el AG 5 veces por 100 generaciones, de esta forma obtenemos unos resultados más realistas, ya que estamos trabajando con probabilidades. También se va a disminuir el número de pasos de la simulación para agilizar el proceso de pruebas.

Como medida para evaluar la influencia que tienen los distintos valores de los parámetros sobre el AG, vamos a guardar el valor del fitness de la mejor solución de cada generación y posteriormente calcular una media de su mejora por generación. Por ejemplo, en nuestro AG buscamos disminuir el valor del fitness con cada generación, por lo que, si el valor de esta medida para una ejecución concreta del algoritmo fuese de 0.25%, nos estaría indicando que aproximadamente con el paso de cada generación disminuimos el valor del fitness en un 0.25%.

En la Tabla 1 podemos observar los valores de los parámetros por defecto utilizados en las pruebas. En cada una de estas pruebas modificaremos únicamente el valor que estemos analizando en ese momento, dejando los demás invariables.

Parámetro	Valor
Tamaño población	100
Número de generaciones	100
Tamaño del torneo	5
Probabilidad de mutación	0.05 (5%)
Probabilidad de cruce	0.8 (80%)
Número de pasos de simulación	600

Tabla 1 Valores por defecto parámetros en las pruebas

Los parámetros que vamos a estudiar son, el tamaño de la población, la probabilidad de mutación, el tamaño del torneo y el valor para el criterio de finalización, que son los parámetros que afectan en mayor medida a una serie de factores importantes en el rendimiento del AG.

Los parámetros de tamaño de la población y valor para el criterio de finalización han sido elegidos para ser analizados y ajustados por el impacto que tienen en el tiempo de ejecución del algoritmo. Intentaremos buscar unos valores que proporcionen un buen punto de equilibrio entre el tiempo de ejecución y la calidad de las soluciones.

Por otro lado, los parámetros de probabilidad de mutación y tamaño del torneo se han seleccionado por el impacto que tienen en la exploración y explotación del algoritmo. Si el algoritmo se centra mucho en la exploración, puede no llegar a converger nunca en una solución o tardar demasiado, y si se centra mucho en la parte de explotación, obtendrá soluciones pobres (óptimos locales).

5.1.1 Tamaño de la población

El tamaño de la población es el parámetro que, junto al número de generaciones, afecta en mayor medida al tiempo de ejecución del AG. Debemos elegir un valor que no sea demasiado pequeño, ya que podría afectar a la diversidad de la población, pero tampoco tremendamente grande que requiera de un excesivo coste temporal.

Se han realizado una serie de ejecuciones utilizando un intervalo de 10 a 150 individuos. Podemos observar cómo el tamaño de la población afecta al porcentaje de mejora media por generación del fitness en la Figura 21, mientras que en la Figura 22 se muestra cómo afecta al tiempo en segundos que tarda en ejecutarse cada generación.

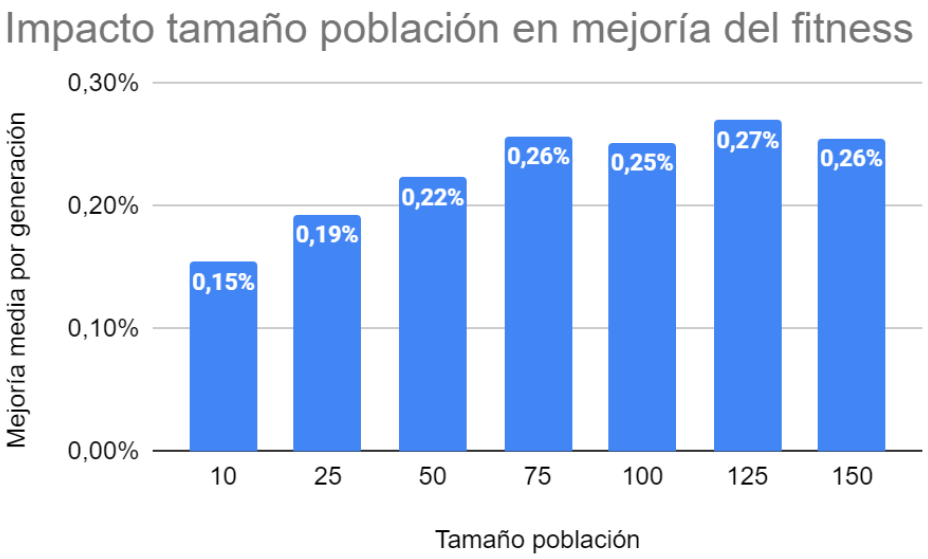


Figura 21 Impacto tamaño población en mejoría del fitness

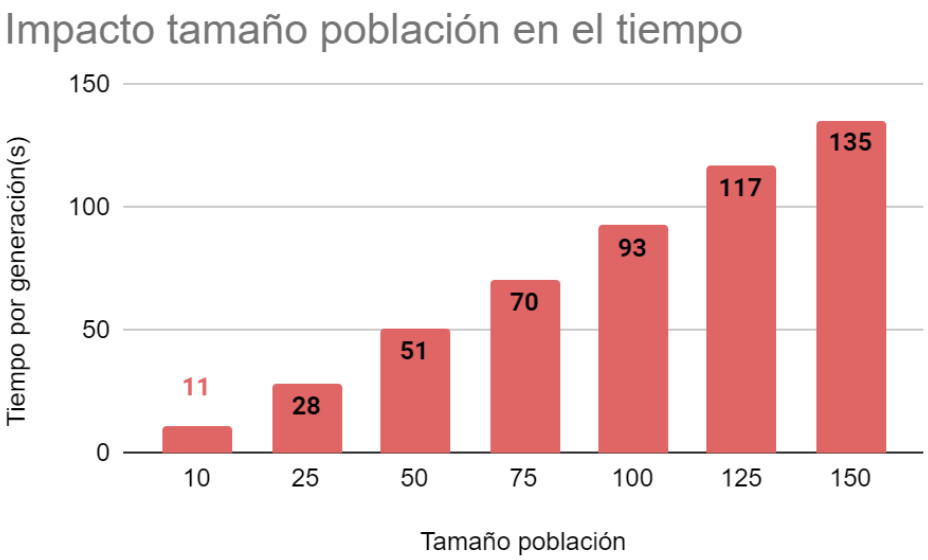


Figura 22 Impacto tamaño población en tiempo de ejecución por generación

En la Figura 21 podemos ver que, para tamaños de población reducidos, la mejoría por generación del fitness es muy pequeña. Esto se debe a la poca variabilidad de la población. También podemos observar que a partir de los 75 individuos y hacia arriba, la mejoría del fitness se mantiene más o menos en la misma proporción. En la Figura 22 podemos ver que efectivamente el tamaño de la población afecta al tiempo de ejecución, en gran medida. Cada individuo aumenta el tiempo en aproximadamente un segundo. Por lo tanto, se ha elegido un tamaño de 75 individuos, ya que es el que tiene un menor tiempo de ejecución por generación de entre los tamaños que han mostrado una mayor mejoría del fitness.

5.1.2 Probabilidad de mutación

Ahora vamos a intentar encontrar un buen valor para la probabilidad de aplicar el operador de mutación sobre un gen. Si esta probabilidad tuviese un valor alto, dificultaría que el algoritmo convergiera en una buena solución. Mientras que uno bajo llevaría a una convergencia prematura en una solución óptima local, al no producirse la suficiente variabilidad.

Se ha seleccionado estudiar un intervalo de 0.0025 a 0.11 o, dicho de otra forma, de 0,25% a 11% de probabilidad de mutación. Podemos ver cómo afecta la probabilidad de mutación a la mejoría del fitness por generación, en la Figura 23.

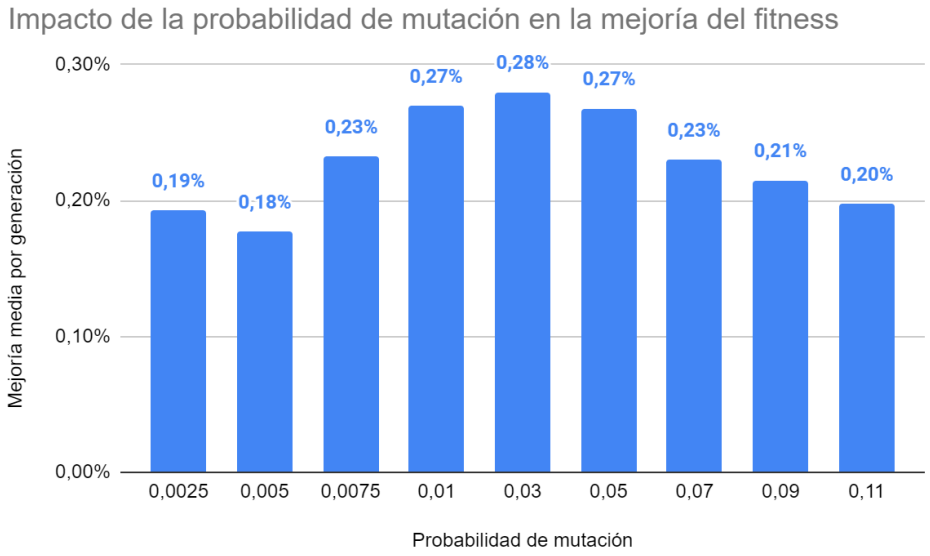


Figura 23 Impacto de la probabilidad de mutación en la mejoría del fitness

En la figura se ve claramente la importancia de este parámetro, y que obtenemos los mejores porcentajes de mejora en el intervalo del 0.01 (1%) al 0.05 (5%) de probabilidad de mutación. Cuanto más nos alejamos de este intervalo, peores son los porcentajes de mejora. Se ha decidido utilizar un valor de 0.03 (3%) perteneciente a este intervalo y, aunque su porcentaje de mejora del fitness solo sea mayor en una centésima respecto a sus dos valores adyacentes, esta diferencia acaba suponiendo un notable impacto tras ejecutar el AG por varias generaciones.

5.1.3 Tamaño del torneo

El tamaño del torneo es un parámetro importante del operador de selección por torneo que hemos elegido para seleccionar a los padres de cada generación. Este parámetro controla la presión de selección, si utilizáramos un valor de tamaño de torneo grande, la presión de selección sería alta, por lo que la diversidad de la población entre generaciones se reduciría considerablemente. Por otro lado, si fuese pequeño, el número de generaciones necesarias para encontrar una solución aumentaría y, por lo tanto, habría un menor porcentaje de mejora del fitness entre generaciones. Debemos buscar un valor que nos permita mantener una buena diversidad poblacional y alcanzar una solución de calidad en un tiempo razonable.

Para analizar este parámetro, se han realizado pruebas variando el valor del tamaño del torneo de manera porcentual. Por ejemplo, si estamos usando un 5% como tamaño del torneo en una población de 100 individuos, en cada torneo se seleccionarán a 5 individuos distintos. Se ha elegido analizar un intervalo que va desde un 2% a un 50%. Podemos observar cómo influyen en la mejora del fitness por generación en la Figura 24.

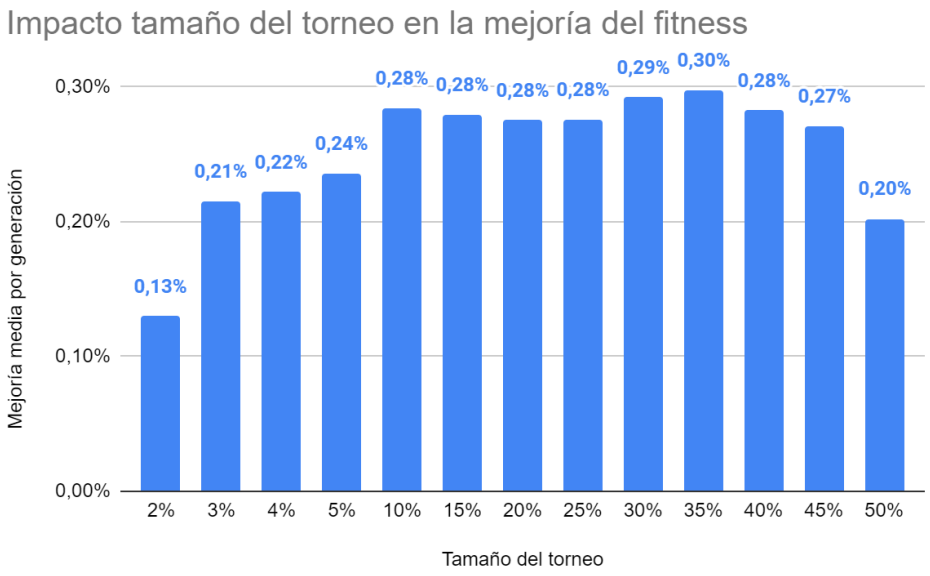


Figura 24 Impacto tamaño del torneo en la mejoría del fitness

En la figura se puede ver como valores menores al 10% del tamaño de la población tienen una mejora de fitness por generación muy pequeña y, que a partir del 40%, también disminuye debido a la reducción de la diversidad de las poblaciones. Se ha decidido utilizar un valor para tamaño del torneo del 10%, que tiene un buen porcentaje de mejora del fitness. Al ser pequeño, también conseguirá mantener la diversidad entre la población permitiéndonos así obtener soluciones de mayor calidad en mayor medida.

5.1.4 Valor para el criterio de finalización

Como ya hemos dicho anteriormente, en el capítulo de la implementación, nuestro criterio de finalización no es el de llegar a un número máximo de generaciones, sino el de terminar cuando el AG no consiga mejoras en el fitness tras un número concreto de generaciones. Con esta prueba vamos a determinar un valor para este número de generaciones. Para obtener resultados más precisos en esta última prueba, vamos a utilizar los valores seleccionados para los parámetros obtenidos en las pruebas anteriores. Podemos ver un resumen de estos valores en la Tabla 2. En esta prueba no vamos a indicar un número máximo de generaciones, de esta forma podemos ver el coste computacional de cada uno de los distintos valores, ya que cuanto mayor sea el valor, más probable será que el número de generaciones y, por lo tanto, el tiempo de ejecución sea también mayor.

Parámetro	Valor
Tamaño población	75
Número de generaciones	N/A
Tamaño del torneo	8 (10%)
Probabilidad de mutación	0.03 (3%)
Probabilidad de cruce	0.8 (80%)
Número de pasos de simulación	600

Tabla 2 Valores utilizados en la prueba del valor para el criterio de finalización

Vamos a utilizar un intervalo de números desde 2 a 50 generaciones en las pruebas que constituyen el número máximo de generaciones sin mejora. Por ejemplo, si lo ejecutamos con un valor de 2, en cuanto 3 generaciones consecutivas no hayan aportado ninguna mejora al valor del fitness, el algoritmo finalizará. En la Figura 25 podemos observar una gráfica en la que se muestra cómo afecta este valor a la mejora total porcentual del fitness, entre el fitness obtenido de la evaluación de los programas por defecto y la mejor solución obtenida por nuestro algoritmo. Después en la Figura 26 se muestra el número de generaciones que necesita nuestro AG para finalizar para cada valor.

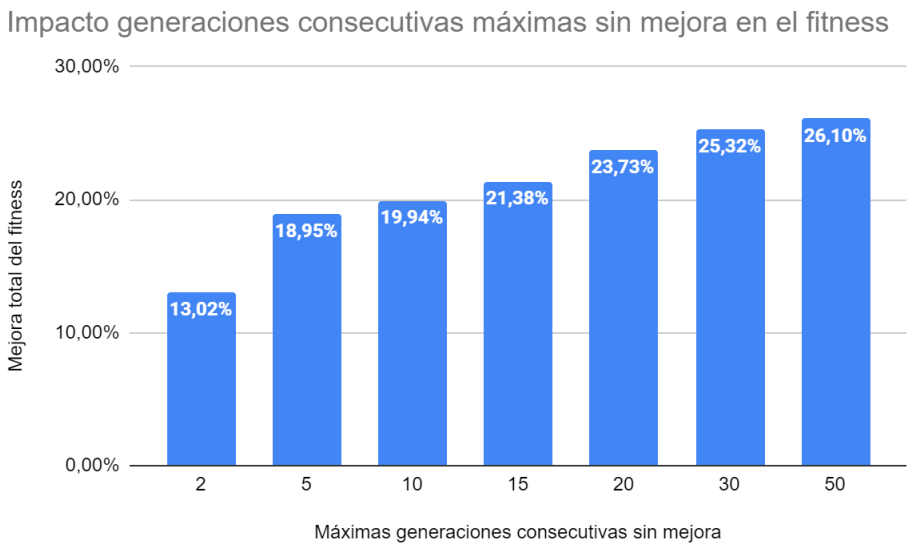


Figura 25 Impacto generaciones consecutivas máximas sin mejora en el fitness

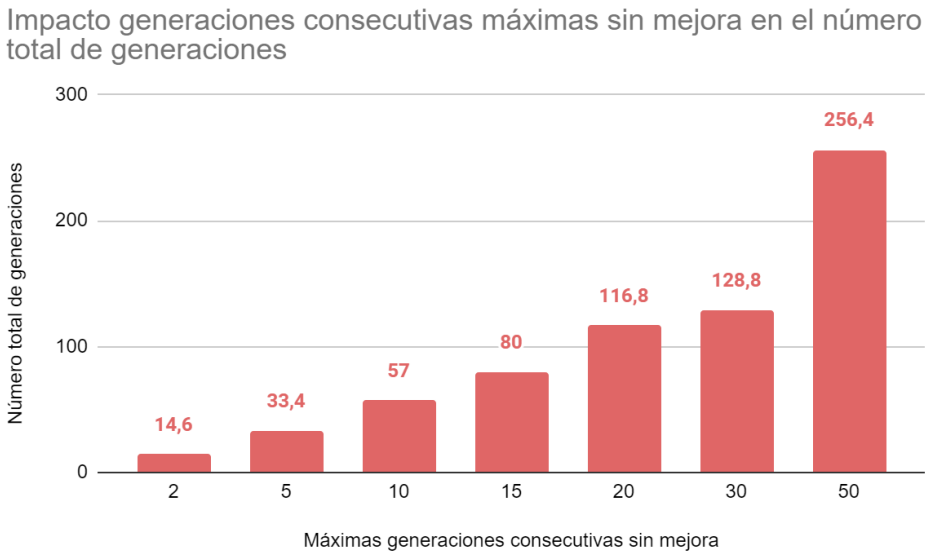


Figura 26 Impacto generaciones consecutivas máximas sin mejora en el número total de generaciones

Como podemos ver en la Figura 25, a partir de un valor de 15 para este parámetro, ya obtenemos mejoras superiores al 20% en el fitness. También se puede observar que cuanto mayor sea el valor de este parámetro, mayor suele ser esta mejora. Sin embargo, en la Figura 26 se puede ver que aumentar este valor también aumenta el número de generaciones totales en gran medida y, por lo tanto, el tiempo de ejecución total, lo que nos lleva a tener que elegir un valor que nos proporcione un buen porcentaje de mejora en un número de generaciones no muy elevado.

Un valor de 30 es el que mejor cumple con estos objetivos, ya que para el valor de 50 el número de generaciones totales se incrementa demasiado para una mejora inferior al 1% respecto al valor del 30. Además, para valores inferiores a 30, la mejora también es bastante más pequeña, especialmente para los inferiores a 20. Como usar el valor de 30 no nos supone tampoco un incremento de generaciones muy grande, será el que utilizaremos.

5.2 Resultados finales

Podemos ver en la Tabla 3 cuales son los valores finales que utilizaremos para ejecutar el AG, obtenidos como resultado de las pruebas anteriores, donde el número de generaciones no tiene un valor, porque el criterio de finalización utilizado no es el de llegar a un número máximo de generaciones, sino, como hemos concluido en las pruebas, no realizar ningún progreso en la mejora del fitness en 30 generaciones consecutivas. También cabe destacar, que se ha incrementado el número de pasos de simulación.

Parámetro	Valor
Tamaño población	75
Número de generaciones	N/A
Tamaño del torneo	8 (10%)
Probabilidad de mutación	0.03 (3%)
Probabilidad de cruce	0.8 (80%)
Número de pasos de simulación	750

Tabla 3 Valores finales parámetros del AG

A continuación, analizaremos y compararemos los resultados de los programas generados por el AG con los del algoritmo SCPG de SUMO. Para cada mapa se han generado dos archivos de rutas distintos, uno con baja demanda de tráfico y otro con alta. La demanda de tráfico en el mapa de Valencia es mayor que en el de Barcelona, ya que estamos utilizando un mapa más grande. Cabe esperar mayor mejora para las situaciones de demanda alta en el que los programas generados por defecto por el algoritmo SCPG de SUMO son muy ineficientes.

Los datos que vamos a comparar son el fitness, el consumo de combustible medio por vehículo, el tiempo de espera medio por vehículo y el número de vehículos que consiguen finalizar su ruta antes de llegar al número de pasos límite de ejecución. Todos los resultados mostrados a continuación se han obtenido como media tras realizar tres ejecuciones del algoritmo para cada configuración de demanda y mapa.

5.2.1 Resultados Barcelona

En la Figura 27, Figura 28, Figura 29 y Figura 30 podemos ver las gráficas comparativas de los resultados para el mapa de Barcelona. Las barras de color rojo representan a los datos de las ejecuciones por defecto con los programas generados por el algoritmo SCPG de SUMO y las azules a los obtenidos del AG. En la parte de debajo de las gráficas hay una etiqueta que nos indica si se trata de una situación de baja demanda de tráfico (BD) o alta (AD). Cabe recordar que buscamos reducir los resultados de todos los apartados, menos el del número de vehículos que finalizan el trayecto, ya que buscamos aumentarlo. Finalmente, en la Figura 31 se encuentra una gráfica resumen en la que se muestra el porcentaje de mejora que hemos obtenido mediante la utilización del AG en cada uno de los apartados.

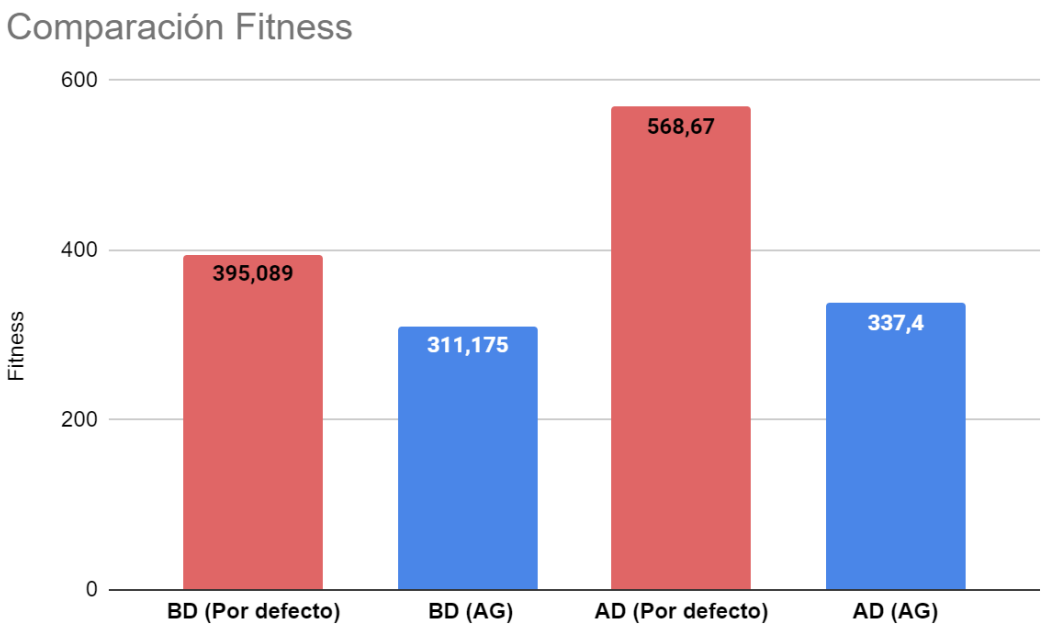


Figura 27 Comparación fitness (Barcelona)

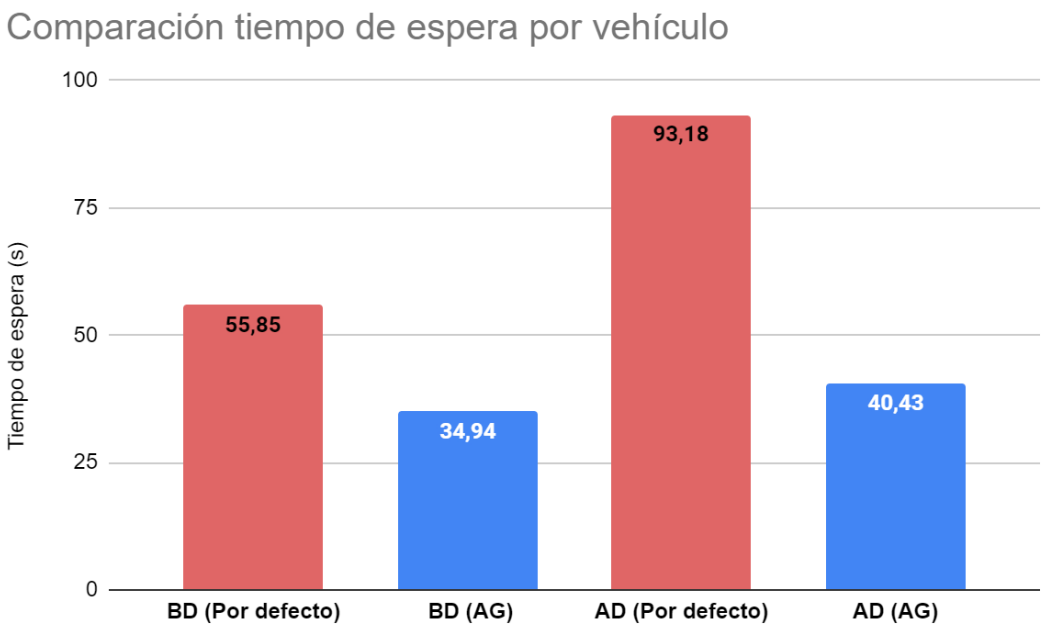


Figura 28 Comparación tiempo de espera medio por vehículo (Barcelona)

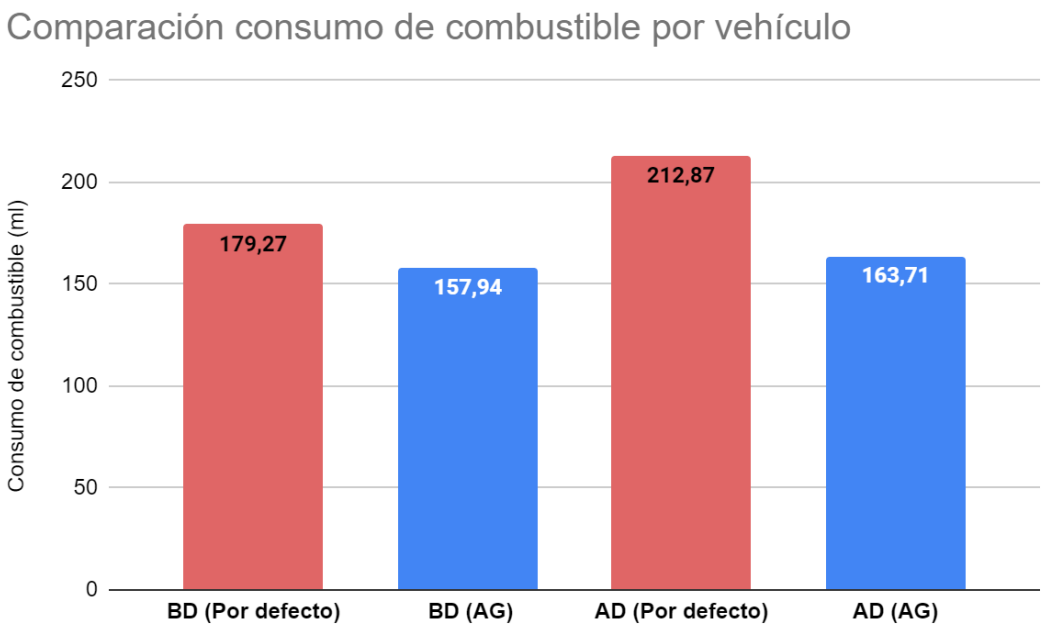


Figura 29 Comparación consumo de combustible medio por vehículo (Barcelona)

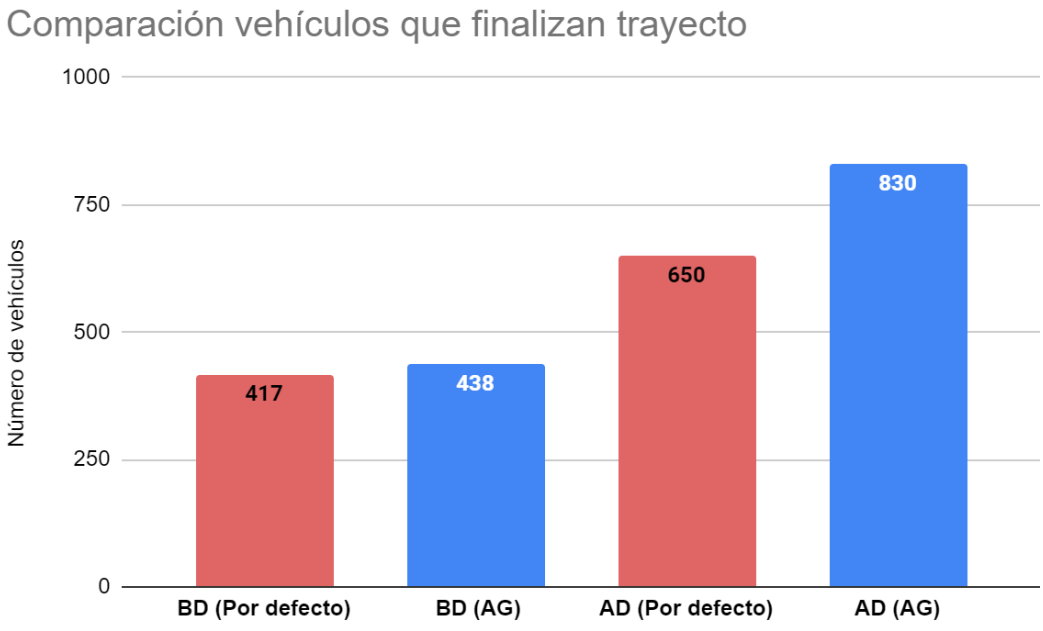


Figura 30 Comparación vehículos que finalizan trayecto (Barcelona)

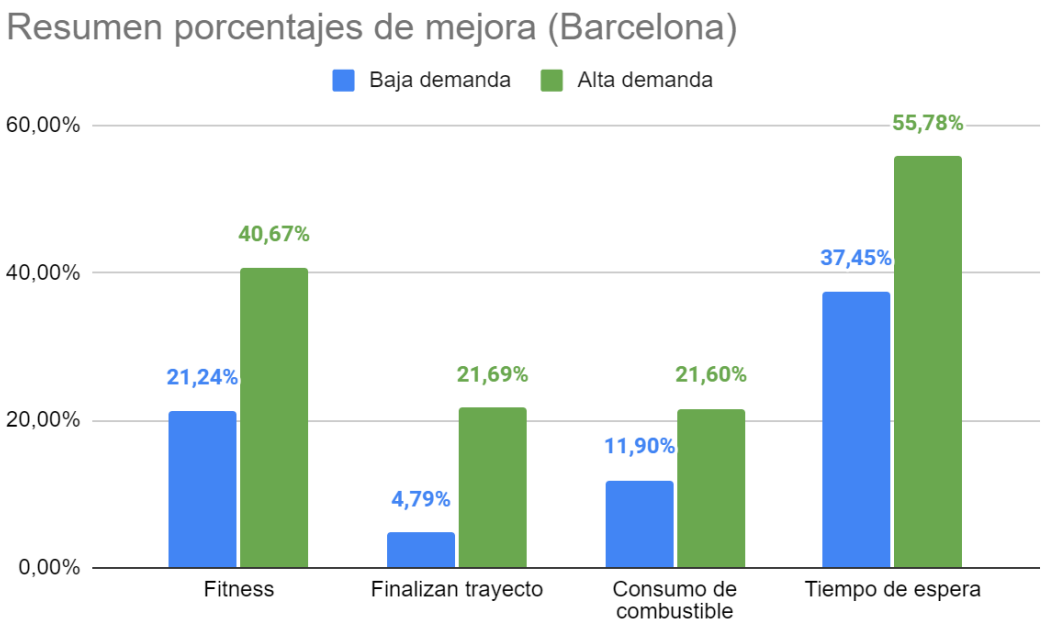


Figura 31 Resumen porcentajes de mejora (Barcelona)

5.2.2 Resultados Valencia

En la Figura 32, Figura 33, Figura 34 y Figura 35 podemos ver las gráficas comparativas de los resultados para el mapa de Valencia. Las barras de color rojo representan a los datos de las ejecuciones por defecto con los programas generados por el algoritmo SCPG de SUMO y las azules a los obtenidos del AG. En la parte de debajo de las gráficas hay una etiqueta que nos indica si se trata de la situación de baja demanda de tráfico (BD) o alta (AD). Finalmente, en la Figura 36 se encuentra una gráfica resumen en la que se muestra el porcentaje de mejora que hemos obtenido mediante la utilización del AG en cada uno de los apartados.

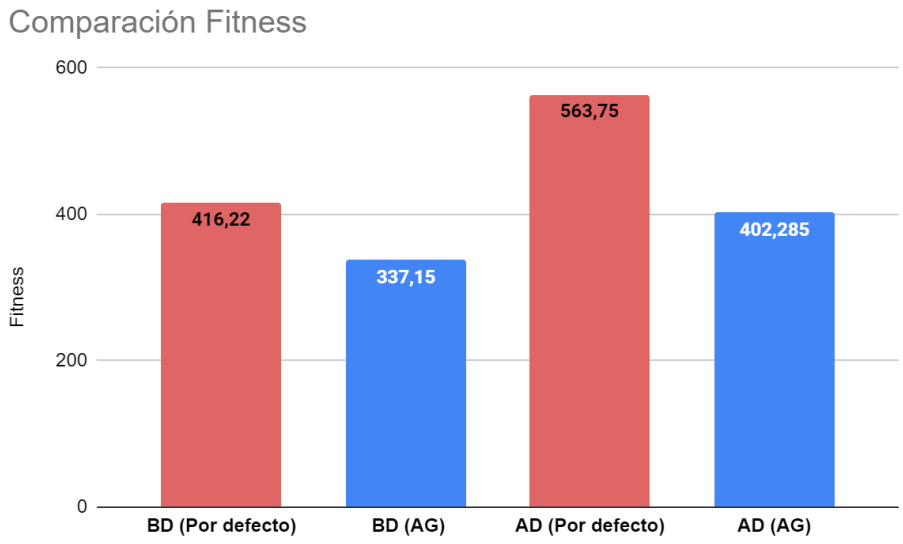


Figura 32 Comparación fitness (Valencia)

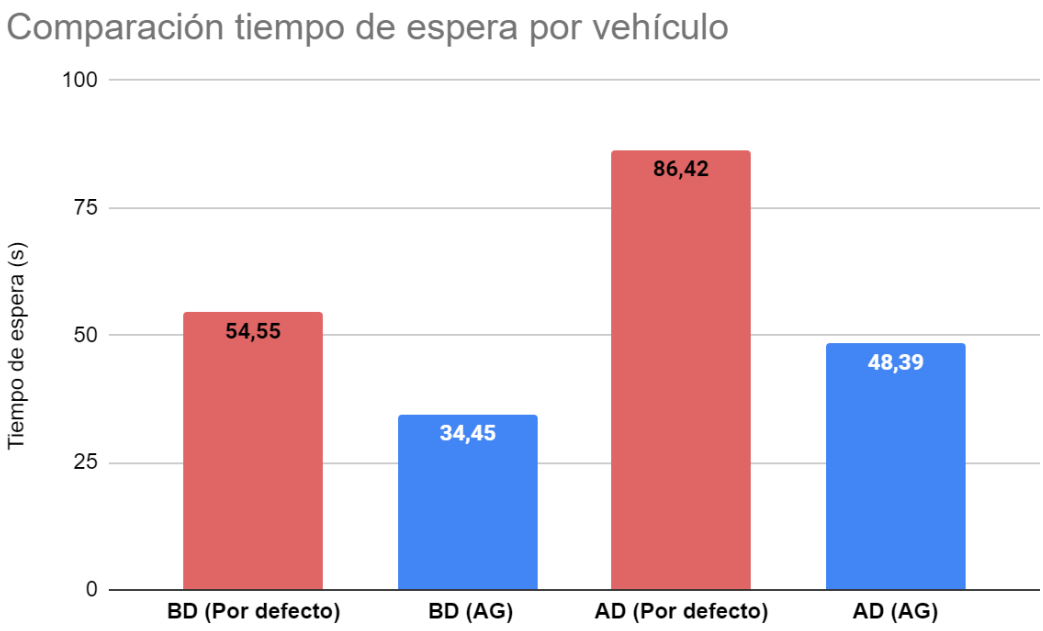


Figura 33 Comparación tiempo de espera medio por vehículo (Valencia)

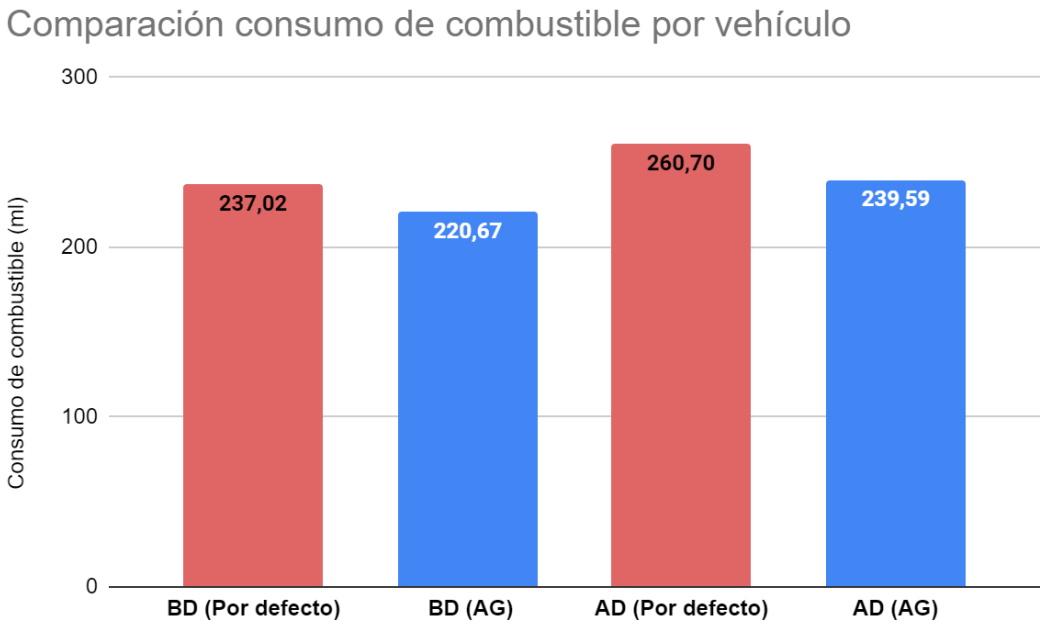


Figura 34 Comparación consumo de combustible medio por vehículo (Valencia)

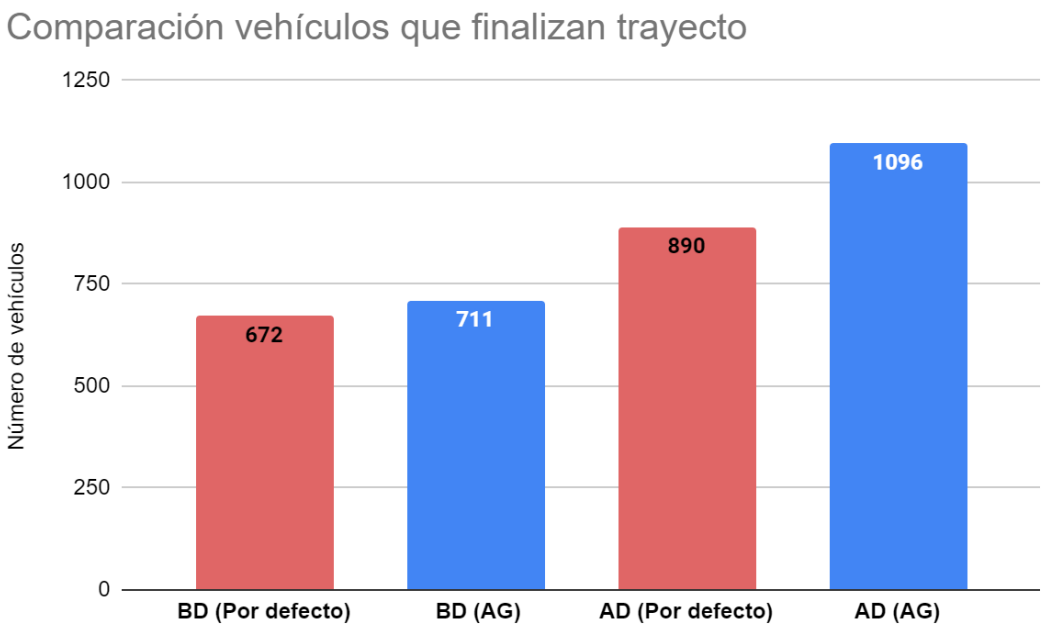


Figura 35 Comparación vehículos que finalizan trayecto (Valencia)

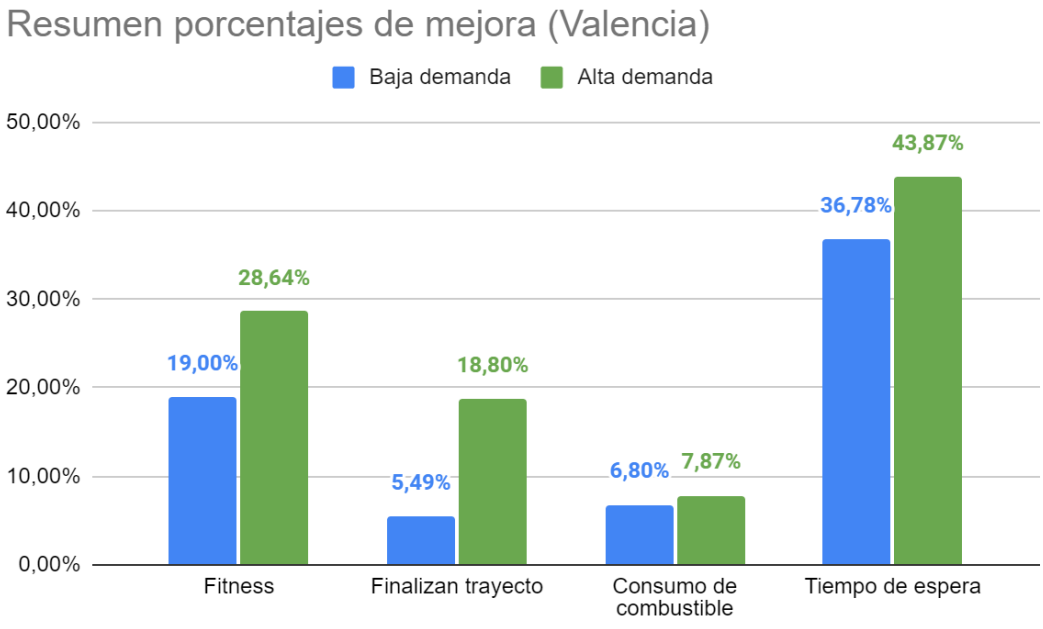


Figura 36 Resumen porcentajes de mejora (Valencia)

6. Conclusiones y trabajos futuros

Para terminar, en este apartado veremos si hemos cumplido con los objetivos propuestos y lo que se ha aprendido durante la realización de este trabajo. También, nombraremos una serie de asignaturas cursadas durante la carrera que han ayudado a la realización del proyecto, y daremos algunas ideas que se podrían añadir al trabajo para conseguir una herramienta más completa.

6.1 Conclusiones

El principal objetivo de este proyecto era el de crear una herramienta de optimización de los tiempos de los ciclos semafóricos, que produzca una reducción de los tiempos de espera de los conductores, del consumo de combustible y de las emisiones. Podemos dar por cumplido este objetivo tras haber observado las distintas gráficas presentadas en el apartado de resultados finales, donde se aprecian grandes porcentajes de mejora en las comparaciones de cada uno de los datos analizados y una mejora general del flujo del tráfico debido al aumento de vehículos que finalizan el trayecto.

Los resultados también demuestran que la implementación del AG y de todos sus parámetros ha sido la adecuada, ya que se consigue optimizar el problema que hemos planteado. Además, también ha quedado demostrado que la herramienta es capaz de funcionar con diferentes mapas de redes de carreteras, rutas y demandas de tráfico.

Con la realización de este trabajo el alumno ha obtenido una gran cantidad de nuevos conocimientos, mayormente relacionados con la inteligencia artificial que es un campo de gran interés y que no había trabajado mucho con la rama cursada. Entre estos conocimientos adquiridos cabe destacar:

- El funcionamiento del aprendizaje profundo, las principales técnicas metaheurísticas y, en especial, sobre los algoritmos genéticos, que son una herramienta con un gran potencial y complejidad.
- El aumento del desempeño con el lenguaje de programación *Python* y con algunas de sus librerías, ya que aparte del AG, también se han creado una serie de herramientas para el análisis de datos.
- La obtención de un gran entendimiento del funcionamiento, utilización y programación de la herramienta de simulación de microtráfico SUMO.

6.2 Relación del trabajo con los estudios cursados

Varias de las asignaturas cursadas durante la carrera han proporcionado los conocimientos básicos que han permitido realizar el trabajo. Algunas de estas son:

- **Mantenimiento y evolución de software:** en esta asignatura se realizó una introducción al funcionamiento y uso de *GitHub,* que ha permitido trabajar en el mismo proyecto desde distintos ordenadores.
- **Computación paralela:** las simulaciones en SUMO son el proceso más costoso computacionalmente de todo el algoritmo, por lo que ha sido necesario paralelizar las simulaciones y utilizar una cola de trabajo.

- **Diseño de software:** se ha intentado seguir en todo momento unas buenas prácticas de código para poder desarrollar un código limpio sobre el que trabajar.
- **Introducción a la informática y a la programación:** en esta asignatura se presentaron las bases sobre la programación que han permitido al alumno aprender a utilizar nuevos lenguajes de programación como *Python*.
- **Sistemas inteligentes:** en esta asignatura se impartió un conocimiento básico sobre cómo funcionan algunas técnicas de inteligencia artificial, que han facilitado el entendimiento de muchos conceptos.

6.3 Trabajos futuros

Aunque hayamos cumplido los objetivos buscados, todavía hay una serie de mejoras que se le pueden hacer a la herramienta para que esta sea más completa y eficiente. También ha quedado pendiente observar el funcionamiento de la herramienta con otros tipos de datos de entrada. Algunos de los cambios y mejoras son:

- Analizar el funcionamiento de la herramienta con mapas mucho más grandes. Esto no ha sido posible, por ahora, debido a las limitaciones de exportación de *OpenStreetMap*.
- Añadir a la simulación la circulación de otros medios de transporte, como puede ser el transporte público u otros tipos de vehículos.
- Añadir la simulación de peatones y de pasos para peatones, para poder reducir los tiempos de espera de estos en los semáforos y el tiempo que tardarían en completar sus viajes.
- Utilizar rutas de vehículos más realistas de entrada para el AG, utilizando otras herramientas del paquete de SUMO como *ActivityGen*.
- Utilizar distintos modelos de emisión en los vehículos.
- Usar TraCI reduce la velocidad de ejecución de las simulaciones. Podría aumentarse el rendimiento si se utilizase el cliente de C++ en lugar del de *Python,* ya que este es un poco más eficiente.

7. Bibliografía

1. **elPeriódico. *Barcelona, la ciudad con más atascos de España.* [En línea] [Citado el: 2 de Agosto de 2021.] https://www.elperiodico.com/es/barcelona/20200129/barcelona-la-ciudad-con-mas-atascos-de-espana-7826454.**

2. ¿Cuántas horas al año pasas atascado en el tráfico? [En línea] [Citado el: 3 de Agosto de 2021.] https://www.lavanguardia.com/motor/rankings/20180330/441822247588/cuantas-horas-ano-atascado-trafico.html.

3. OMS. Air Pollution. [En línea] [Citado el: 2 de Agosto de 2021.] https://www.who.int/health-topics/air-pollution#tab=tab_1.

4. Optimizing Traffic Signal Timing Significantly Reduces the Consumption of Fuel in the City of Portand. [En línea] [Citado el: 1 de Agosto de 2021.] https://www.c40.org/case_studies/optimizing-traffic-signal-timing-significantly-reduces-the-consumption-of-fuel.

5. Changes to signal timing mean big benefits for Phoenix drivers. [En línea] [Citado el: 2 de Agosto de 2021.] https://azdot.gov/adot-news/changes-signal-timing-mean-big-benefits-phoenix-drivers.

6. Contaminación del aire y muertes prematuras. [En línea] 2 de Agosto de 2021. https://www.sumendi.org/es/2020/10/15/contaminacion-del-aire-y-muertes-prematuras/.

7. El ciclo semafórico. ¿Sabes qué es y cómo puede ser útil en la reconstrucción de un accidente de tráfico? [En línea] [Citado el: 4 de 08 de 2021.] https://reconstruccionaccidentestrafico.com/el-ciclo-semaforico-sabes-que-es-y-como-puede-ser-util-en-la-reconstruccion-de-un-accidente-de-trafico/.

8. All 3 Types of Traffic Signals And General Practice of Design. [En línea] [Citado el: 4 de 08 de 2021.] https://engineeringbrother.com/types-of-traffic-signals.html.

9. *Smart Traffic Light System Using Machine Learning.* M. B. Natafgi, M. Osman, A. S. Haidar y L. Hamandi. 2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET).

10. *Q-learning method for controlling traffic signal phase time in a single intersection.* S. Araghi, A. Khosravi, M. Johnstone y D. Creighton. 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), 2013.

11. Abu-Shawish, Israa & Ghunaim, Sara & Azzeh, Mohammad & Nassif, Ali. Metaheuristic Techniques in Optimizing Traffic Control Lights: A Systematic Review. International Journal of Systems Applications, Engineering & Development. 2020.

12. *A survey on metaheuristics for stochastic combinatorial.* Bianchi, Leonora, y otros. 2009.

13. Metaheurística wikipedia. [En línea] [Citado el: 16 de 08 de 2021.] https://es.wikipedia.org/wiki/Metaheur%C3%ADstica.

14. Particle swarm optimization wiki. [En línea] [Citado el: 16 de 08 de 2021.] https://en.wikipedia.org/wiki/Particle_swarm_optimization.

15. *"Optimising traffic lights with metaheuristics: Reduction of car emissions and consumption," 2014 International Joint Conference on Neural Networks (IJCNN).* J. Garcia-Nieto, J. Ferrer y E. Alba. 2014.

16. Thun, Dr. Jörn-Henrik. *What is the Ant Colony Optimization Algorithm?* [YouTube] 2016.

17. [En línea] [Citado el: 16 de 08 de 2020.] https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#Algorithm_and_formulae.

18. *"Traffic signal optimization using Ant Colony Algorithm," The 2012 International Joint Conference on Neural Networks (IJCNN).* Yu, D. Renfrew y X. 2012.

19. SUMO at a Glance. [En línea] [Citado el: 4 de Julio de 2021.] https://sumo.dlr.de/docs/SUMO_at_a_Glance.html.

20. SUMO User Documentation. [En línea] [Citado el: 28 de Junio de 2021.] https://sumo.dlr.de/docs/index.html.

21. Genetic Algorithms tutorial. [En línea] https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_quick_guide.htm.

22. Task Scheduling in Grid Environment Using Simulated Annealing and Genetic Algorithm. [En línea] [Citado el: 21 de Julio de 2021.] https://www.semanticscholar.org/paper/0-Task-Scheduling-in-Grid-Environment-Using-and-Abdulal-Jabas/188018edb5a6d3131e5ec94245fb4c7dda5193b7/figure/9.

23. *The Open Source Traffic Simulation Package SUMO. RoboCup 2006 Infrastructure Simulation Competition .* Daniel KrajzewiczT, Michael Bonert y Peter Wagner. 2006.

24. Traffic lights SUMO. [En línea] [Citado el: 28 de Junio de 2021.] https://sumo.dlr.de/docs/Simulation/Traffic_Lights.html.

25. OpenStreetMap. [En línea] [Citado el: 26 de 06 de 2021.] https://www.openstreetmap.org/.