Trabajo Fin de Máster:

Optimización del Tráfico Rodado en Ciudades Inteligentes

Alumno:
Daniel H. Stolfi
dhstolfi@lcc.uma.es
Universidad de Málaga

Director:
Enrique Alba Torres
eat@lcc.uma.es
Universidad de Málaga

Septiembre de 2012

Abstract

This work presents an optimization of the road traffic in the city of Málaga. Our proposal consists in the use of spots, called red swarm, which will be used to change the routes of the vehicles and which will be placed on some of the existent traffic lights. An evolutionary algorithm is also proposed in order to find a solution that reduce the travel time of the vehicles by using the red swarm. The working map was obtained from the Open Street Map project and was enriched by adding traffic lights, sensors, routes and vehicle flows. Finally, it has been imported into the SUMO traffic simulator to be used like a method for calculating the fitness of solutions. The results achieved provide an improvement over the original model such as lower travel times, middle stop steps and shorter route length.

Resumen

Este trabajo presenta la optimización del tráfico rodado de la ciudad de Málaga. Nuestra propuesta consiste en la utilización de puntos de cambio de rutas, llamados *red swarm*, que se colocarán en algunos de los semáforos existentes. También se propone un algoritmo evolutivo con el fin de encontrar una solución que reduzca los tiempos de desplazamiento de los vehículos utilizando los *red swarm*. La cartografía utilizada ha sido obtenida desde el proyecto *Open Street Map* y tuvo que ser adaptada, añadiendo semáforos, sensores, rutas y trayectos para los vehículos. Por último, se ha importado el escenario construido para ser utilizado por el simulador *SUMO* como método de cálculo del valor de la función de *fitness* de las soluciones. Los resultados obtenidos mejoran a los del escenario real en cuanto a tiempos de viaje, número de esperas y longitudes de los trayectos recorridos por los vehículos.

Palabras Clave: tráfico rodado, simulación, optimización, algoritmo evolutivo, SUMO, Málaga, Red Swarm

1. Introducción

El concepto Ciudad Inteligente (*Smart City*) [1] comprende seis dimensiones principales, con el objeto de definir un modelo sobre el cual evaluar el desarrollo urbano teniendo en cuenta la sostenibilidad del mismo. Estas dimensiones corresponden a: Economía inteligente (*smart economy*), Personas inteligentes (*smart people*), Gobierno inteligente (*smart governance*), Movilidad inteligente (*smart mobility*), Medio ambiente inteligente (*smart environment*) y Vida cotidiana inteligente (*smart living*).

Si bien el presente trabajo se concentra en mejorar la movilidad (Movilidad inteligente), una ciudad que adolece de atascos también se ve seriamente afectada en su economía (Economía inteligente) porque el trasporte, tanto de mercancías como de personas, pierde eficiencia. Además, un tiempo de trayecto más largo aumenta las emisiones de gases de efecto invernadero y contradice los principios deseables para

un Medio ambiente inteligente. Por último, la fluidez del tráfico también afecta a la vida cotidiana de las personas (Vida cotidiana inteligente), porque éstas no sólo se hallarán de mejor humor cuando no han tenido que padecer un atasco, sino que además dispondrán de más tiempo libre, mejorando así su calidad de vida. Por lo tanto el abordar aquí el problema de la Movilidad Inteligente de una forma innovadora, representará un importante impacto, tanto en la sociedad como en la empresa.

La Unión Europea ha fijado entre sus objetivos para la Estrategia 2020 [2] una reducción en las emisiones de gases de efecto invernadero de los países miembros. España, en particular, se ha comprometido a reducir estas emisiones un 10 % en 2020 respecto a los valores de 2005 en los sectores no cubiertos por el régimen de comercio de los derechos de emisión. Mientras la implantación del coche eléctrico se convierte en una realidad, un sistema eficaz de gestión del tráfico rodado que reduzca los tiempos empleados en cada trayecto contribuirá a la reducción de las emisiones de CO_2 que provenientes de los automotores.

Por otro lado, el problema de la optimización del tráfico rodado ha sido abordado ampliamente como tema de investigación. Se pueden encontrar soluciones que van desde la variación de los ciclos de trabajo de los semáforos [3, 4, 5, 6], la predicción del estado del tráfico utilizando redes neuronales [7], la combinación de modelos matemáticos con modelos basados en conocimientos previos [8], hasta algoritmos enfocados en evitar la paradoja de Braess [9, 10].

Más cercanos a la propuesta de este trabajo se encuentran la simulación de agentes de tráfico que recogen datos mediante sensores de inducción, calculan el estado de congestión de una zona de la ciudad y actualizan los datos del GPS de los vehículos [11] o la reducción de la congestión del tráfico en situaciones puntuales habilitando los carriles bus como vías de circulación adicionales para los todos los tipos de vehículo [12]. Ambas propuestas utilizan técnicas de simulación para validar los resultados obtenidos y ofrecer políticas alternativas al diseñador de la red de circulación urbana.

Actualmente se distinguen tres modelos de simulación de tráfico: el modelo macroscópico [13] que se encuentra basado en la dinámica de fluidos, el modelo mesoscópico [14, 15] que representa el movimiento del tráfico por grupos de vehículos y el modelo microscópico [16], en el que el tráfico se compone de partículas individuales que se mueven según una reglas predefinidas. Luego, entre los simuladores que siguen el modelo microscópico se encuentran los que discretizan el tiempo y el espacio de la simulación y los que trabajan con tiempo y espacio continuo. Estos últimos suelen ser más precisos pero por el contrario requieren más tiempo de cómputo durante la simulación. Otros aspectos importantes de los simuladores de tráfico son la forma en que se define el modelo, pudiendo ser mediante edición manual, interfaz gráfica o importación desde otros formatos. Se distinguen además por la forma en que se visualiza la simulación y cómo se presentan los resultados.

En este trabajo se propone una solución innovadora para evitar la congestión del tráfico rodado en las horas pico, aprovechando parte de la infraestructura existente en muchas ciudades del España y del mundo, como son la cobertura de red WiFi junto con la red de semáforos de tráfico. Analizando la distribución del flujo de tráfico que se produce diariamente, proponemos un algoritmo que precalcule rutas alternativas para los vehículos, evitando la sobre utilización de las vías más rápidas como las avenidas, utilizando en su lugar el resto del trazado urbano existente.

Luego, cuando un vehículo provisto de una OBU[1] se aproxime a un semáforo dotado con red WiFi establecerá un enlace con el mismo comunicándole el destino final de su itinerario. El dispositivo de red en el semáforo junto a su unidad de cómputo, llamados de ahora en adelante *Red Swarm*, comunicará al vehículo una sugerencia de cambio de ruta por una alternativa según la información previamente almacenada obtenida de una optimización a medida del trazado urbano. Si el conductor decide seguir las instrucciones recibidas, se beneficiará de una ruta alternativa hacia su destino o hacia otro *Red Swarm* en el cual el proceso se repetirá hasta alcanzar el destino final de su trayecto. En la Figura 1 se presenta un esquema de funcionamiento en dónde un vehículo es dirigido por una ruta alternativa evitando un atasco mediante la comunicación entre la *OBU* propia y los puntos *Red Swarm*.

Como alternativa a la comunicación V2I[2] se propone el uso de paneles electrónicos, también existentes en muchas ciudades, como medio de comunicación (visual en este caso) de los itinerarios disponibles a los vehículos que circulan por el trazado urbano.

Para llevar a cabo el cálculo previo de las rutas entre *Red Swarms* así como para disponer de un escenario para comparar y valorar la solución propuesta se dispondrá del trazado urbano real de la ciudad de Málaga importado desde el proyecto *Open Street Map* [17], y sobre el cual se generarán una

[1] *On Board Unit*: Dispositivo de usos múltiples que se instala en los vehículo y posee capacidad de comunicarse vía WiFi
[2] *Vehicle to Infraestructure* - Comunicación entre Vehículo e Infraestructura

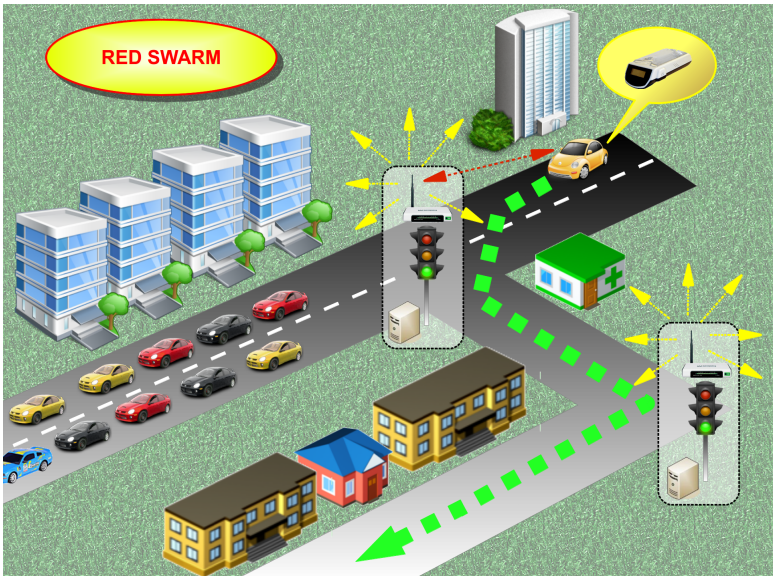


Figura 1: Esquema funcionamiento de los semáforos con *redswarm* integrados en la ciudad

serie de itinerarios para los vehículos adecuándolo para realizar simulaciones utilizando el programa de simulación de tráfico *SUMO (Simulation of Urban MObility)* [18].

SUMO, en su versión 0.15.0, es un simulador que implementa el modelo microscópico de tiempo discreto y espacio continuo junto con el modelo de seguimiento entre coches definido por Krauss en [19]. Se pueden simular distintos tipos de vehículos, calles con múltiples carriles y semáforos con giros a la izquierda y con distintos planes de temporización. Los modelos pueden tener más de 10000 calles y cientos de miles de vehículos. Los principales motivos para la elección se basan en que las características enunciadas se adaptan bien a las necesidades de este trabajo, en que es un software de código abierto, y en especial porque dispone de una interfaz de comunicación con programas externos vía *TraCI (Traffic Control Interface)* [20] además de permitir la importación de los escenarios desde varias fuentes, entre ellas *Open Street Map*.

TraCI consiste en un conjunto de *scripts* escritos en lenguaje *Python* que se interconectan con *SUMO* a través de un puerto TCP y proporcionan una interfaz común para obtener información sobre la simulación así como para realizar modificaciones en la misma. Los cambios de rutas que forman parte de la optimización que persigue este trabajo se realizarán mediante un programa *Python* que hará uso de la biblioteca de funciones provistas por *TraCI*. Además para mejorar la velocidad de ejecución (*Python* es un lenguaje interpretado) se utilizará la implementación alternativa de este lenguaje llamada *PyPy*, que dispone de un compilador *Just-in-Time* con el fin de optimizar los tiempos de ejecución junto con una disminución del consumo de memoria principal.

La optimización será llevada a cabo por un algoritmo evolutivo especialmente diseñado para este problema y para el cual se ensayarán diversos operadores de cruce y mutación para hallar el que mejor comportamiento presente. Cada evaluación de la función de *fitness* para un individuo de la población requerirá una simulación del escenario sujeto a la configuración impuesta por dicho individuo. Luego los resultados se obtendrán a partir del procesamiento de los ficheros de salida que son generados durante la simulación.

En la Figura 2 se presenta un esquema con el flujo de trabajo a realizar que parte de la importación de la información cartografía desde el proyecto *Open Street Map* a la obtención de la configuración para los *Red Swarm*.

El resto de este trabajo se encuentra organizado en las siguientes secciones: A continuación, en la Sección 2 se describe el mapa de la zona de la ciudad sobre la que se va a realizar la optimización y el proceso llevado a cabo para su adecuación e importación. Posteriormente, en la Sección 3 se plantea una solución para la optimización utilizando un algoritmo evolutivo definiendo los distintos componentes y parámetros del mismo. La experimentación realizada junto con los resultados obtenidos se exponen en la Sección 4. Y por último en la Sección 5 se presentan las conclusiones obtenidas así como las posibles

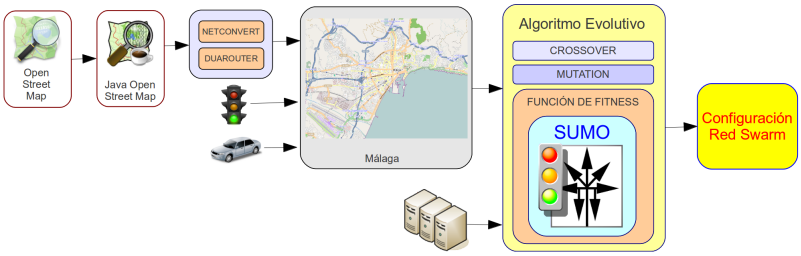
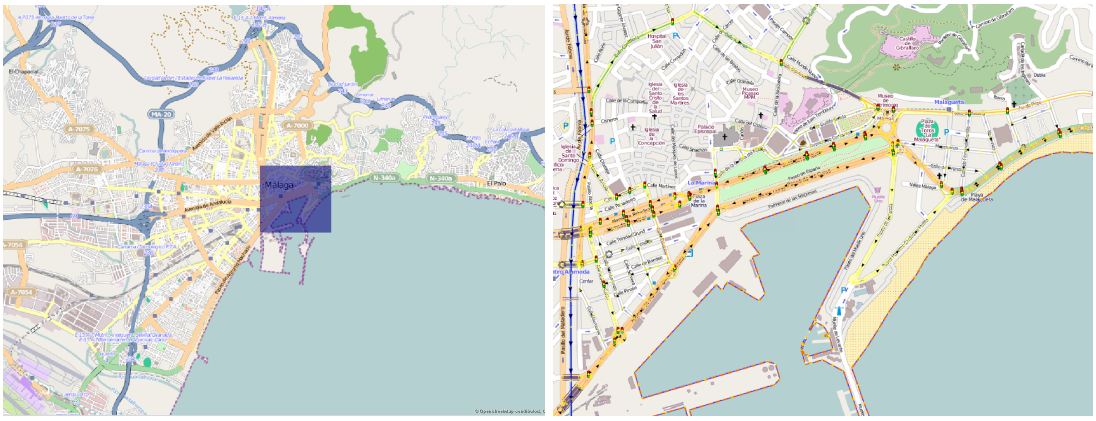


Figura 2: Flujo de trabajo

ampliaciones y trabajos futuros.

2. Escenario de Trabajo

El escenario de trabajo seleccionado consiste en una zona de la ciudad de Málaga con alta concentración de tráfico, la cual se encuentra delimitada por la calle Gutemberg al Este, el río al Guadalmedina al Oeste, el mar Mediterráneo al Sur y la calle Carretería al Norte (Figura 3).



(a) Zona de Málaga (b) Detalle Zona

Figura 3: Zona seleccionada en OpenStreetMap

La ciudad comprende a ocho entradas y ocho salidas que se corresponden con calles por las cuales entran y salen vehículos de la simulación. Además se han dispuesto 28 sensores distribuidos entre las vías de entrada a los diez *Red Swarm* dispuestos en los semáforos, los cuales representan dentro del escenario simulado los puntos en los cuales los vehículos establecen el enlace de radio. En la Figura 4 se incluye un esquema del escenario correspondiente a la ciudad. En el mismo se representan los puntos de entradas y salidas con círculos etiquetados con una letra. Las flechas indican si se trata de una entrada, salida o de una calle de ambos sentidos. Luego, dentro del escenario se incluyen los 28 sensores representados por una pentágono los cuales se encontrarán interconectados de acuerdo al trazado urbano propio de la ciudad. Obsérvese que la entrada/salida denominada **E** representa al tráfico que tiene como origen o destino una ubicación interna de la ciudad al encontrase situada en el interior de la misma. Por lo tanto los vehículos que sigan itinerarios que tengan como destino el salida **E** ingresan a la ciudad desde el exterior de la misma y culminan su viaje en su interior. Por otro lado, los vehículos que sigan itinerarios que tengan como inicio la entrada **E**, representarán a vehículos que abandonan la ciudad durante el tiempo de simulación. Por último, el resto de rutas, al no incluir la entrada/salida **E**, representarán a trayectos que atraviesan la ciudad por completo.

Para la construcción de la zona de simulación se ha partido de los mapas disponibles en el proyecto *Open Street Map* como fuente cartográfica de la ciudad de Málaga importándolos mediante el programa *NetConvert* que acompaña al simulador *SUMO*. Previamente a la importación, ha sido necesario

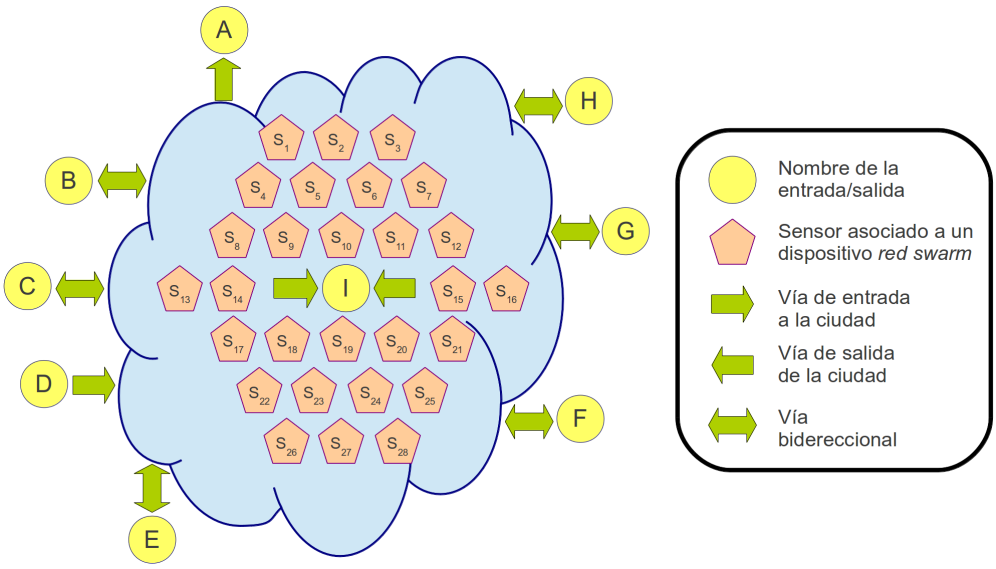


Figura 4: Esquema del escenario de simulación

acondicionar los mapas para la simulación mediante el editor de mapas JOSM (*Java OpenStreetMap*), eliminando información irrelevante (puntos de interés, edificios, etc.), añadiendo semáforos, actualizando rutas, cambiando el tipo de vías, definiendo el número de carriles y velocidades máximas, suprimiendo los giros en U incorrectos y descartando calles no transitables por el tráfico rodado o que carecían de importancia para la simulación. Además, algunos semáforos tuvieron que ser ajustados en cuanto a los giros a la izquierda, la duración y el orden de sus fases. Para el caso de los semáforos de peatones con pulsador manual, se definió un ciclo de operación de 30 segundos en rojo para los vehículos, por cada diez minutos en verde.

Utilizando el programa *Duarouter*, también proporcionado con *SUMO*, se obtuvieron 64 itinerarios entre las ocho entradas y las ocho salidas de la ciudad. Este programa emplea internamente una implementación del algoritmo de Dijkstra [21] no escalable. Como función de coste, se utilizó el tiempo de viaje, bajo la suposición de que los conductores siempre escogen el camino más rápido hacia su destino. Los itinerarios obtenidos tienen la finalidad de que *SUMO* realice la simulación de forma autónoma, seleccionando para cada vehículo que ingresa al sistema uno de los ocho posibles destinos de forma aleatoria según una distribución uniforme. Esta simulación, que la llamaremos **Simulación SUMO**, será la que utilizaremos más adelante como referencia para comparar con los resultados obtenidos en la experimentación.

Habiendo definido las calles, semáforos e itinerarios, el siguiente paso consistió en añadir a la simulación de la ciudad los vehículos que se moverán por la misma. Con el fin de intentar proporcionar variedad en el tráfico rodado se han definido los cuatro tipos de vehículos que se presentan en la Tabla 1 junto con sus características asociadas mediante las cuales influirán en la simulación. Obsérvese que las velocidades indicadas corresponden a las máximas que cada vehículo puede desarrollar aunque luego éstas se verán restringidas por el límite propio de cada tipo de vía.

Tabla 1: Tipos de Vehículos

Tipo	Probabilidad	Vel. Máxima (Km/h)	Aceleración (m/s^2)	Desaceleración (m/s^2)	Longitud (m)
turismo	0,50	160	0,9	5,0	3,8
monovolumen	0,25	100	0,8	4,5	4,2
furgoneta	0,15	50	0,7	4,0	4,3
camión	0,10	40	0,6	3,5	4,5

Los viajes que estos vehículos realizarán se han definido en base a los itinerarios existentes, como se ha mencionado anteriormente, mientras que el tipo de vehículo que es asignado al viaje también

será escogido de forma aleatoria por *SUMO*, pero en este caso según la probabilidad predefinida para cada uno (Tabla 1).

Por otro lado, con el objeto de realizar la optimización de los viajes, se seleccionaron de acuerdo a su ubicación diez semáforos los cuales se comportarán como puntos *Red Swarm*. Éstos propondrán un cambio en la ruta que recorre un vehículo para alcanzar su destino escogiéndola entre las disponibles según la calle por la que se aproxima y el destino final de su itinerario. Para ello se añadieron a la simulación los 28 sensores que dispararán el algoritmo de cambio de ruta, el cual le comunicará al vehículo la propuesta del nuevo itinerario. Para este trabajo se ha supuesto que el 100 % de los vehículos aceptan las indicaciones de cambio de ruta recibidas.

Junto con los itinerarios que seguirán los vehículos en la Simulación SUMO, se calcularon todas las rutas disponibles entre cada sensor asociado a los *Red Swarms* utilizando también el programa *Duarouter*. Para generar todas las rutas posibles se emplearon las distintas funciones de coste para el algoritmo de Dijkstra que *Duarouter* dispone. Estas magnitudes son: tiempo de viaje (*traveltime*), cantidad de ruido (*noise*), emisiones de CO (*CO*), CO_2 (*CO2*), Partículas (*PMx*), Hidrocarburos (*HC*) y Óxidos de Nitrógeno (*NOx*) y por último, el consumo de combustible (*fuel*). El número total de rutas únicas calculadas entre los sensores y entre los sensores y las salidas de la ciudad ha sido de 236.

Obsérvese que la configuración de la simulación se realiza en mayor parte de forma automática por *scripts* sin intervención humana. Esto fue diseñado de este modo para facilitar las futuras extensiones a otras ciudades como se propone más adelante en los trabajos futuros. En la Tabla 2 se presenta un resumen de las características de la zona de simulación que llamaremos de ahora en más, **Málaga**.

Tabla 2: Características de Málaga

Número de *Red Swarms*	10
Número de sensores	28
Número de entradas	8
Número de salidas	8
Itinerarios diferentes	64 (8×8)
Tipos de vehículos	4
Número de vehículos	800 (8×100)

3. Solución Propuesta

La optimización de Málaga con el objeto de minimizar los tiempos de viaje de los vehículos que la recorren se ha abordado mediante un algoritmo evolutivo provisto de un conjunto de operadores de recombinación y mutación a evaluar experimentalmente y así determinar cual se comporta mejor con este tipo de problema.

El pseudocódigo correspondiente al algoritmo evolutivo que se utilizará se presenta en el Algoritmo 1.

Algoritmo 1 Algoritmo Evolutivo

```
generate(population)
while not (stopCondition()) do
  parents ← selection()
  offsprings ← crossover(parents)
  offsprings ← mutation(offspring)
  if evaluation(offsprings) then
    old ← worst(population)
    replace(old, offsprings)
  end if
end while
return best(population)
```

La condición de parada vendrá dada por una cota superior al número de iteraciones totales que se determinará experimentalmente o por un número máximo de iteraciones para las cuales la mejor solución encontrada por el algoritmo no ha variado. Mientras esta condición de parada no se cumpla, en

el algoritmo se seleccionan dos padres de forma totalmente aleatoria entre los individuos pertenecientes a la población. Mediante la función de recombinación se obtienen dos nuevos individuos que sufrirán una posible mutación. Luego se seleccionan desde la población actual los dos individuos para los cuales el valor devuelto por la función de *fitness* son los mayores de la población. Estos individuos, considerados los peores de la generación, serán reemplazados por los nuevos siempre y cuando éstos últimos posean una evaluación mejor o igual que los antiguos, pasando así a formar parte de la población en la siguiente generación. Una vez finalizada la ejecución del bucle principal al hacerse verdadera la condición de detención, el algoritmo devolverá el mejor individuo de la población hallado hasta el momento.

3.1. Sistema de Representación

El sistema de representación escogido para cada individuo contiene la configuración de cada *Red Swarm* de Málaga desglosado en los 28 sensores. Luego, cada sensor contiene a su vez los ocho destinos finales posibles a los que se puede dirigir el vehículo. Por último, cada destino contendrá a su vez todas las rutas disponibles desde el sensor actual hacia el resto de los sensores o salidas del sistema, pudiendo existir una, varias o ninguna ruta, dependiendo del sensor, del destino y del trazado urbano.

En la Figura 5 se puede visualizar el sistema de representación empleado para cada individuo de la población. En el mismo se dispone de los 28 sensores (S_1 – S_{28}), cada uno con los ocho destinos (D_1 – D_8) posibles, y en cada uno de los destinos, K_M rutas posibles.

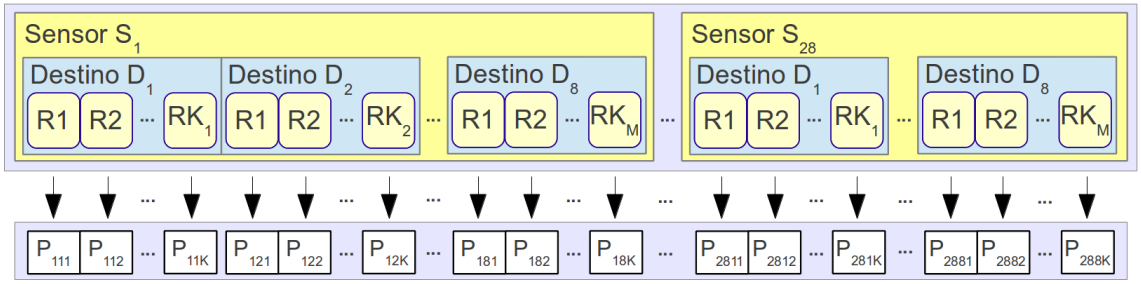


Figura 5: Representación del estado de un individuo

Luego las rutas tendrán asociadas una probabilidad P, siendo estos valores los que diferenciarán un individuo de otro y sobre los que operará el algoritmo evolutivo. La restricción a la que se verá sometida la configuración de las rutas de cada destino dentro de un sensor es que la suma de las probabilidades de sus rutas (P_{S,D,R_i}) sea siempre igual a 1,0 como se expone en la Fórmula 1. Obsérvese en la parte inferior de la Figura 5 una representación a más bajo nivel del individuo en dónde se encuentra el vector de números enteros que pueden variar entre 0 y 100 representando el rango de probabilidades entre 0,0 y 1,0, sujeto a las restricciones antes mencionadas. Se han utilizado números enteros en vez de coma flotante por cuestiones de velocidad y eficiencia a la hora de trabajar con estos números en el computador.

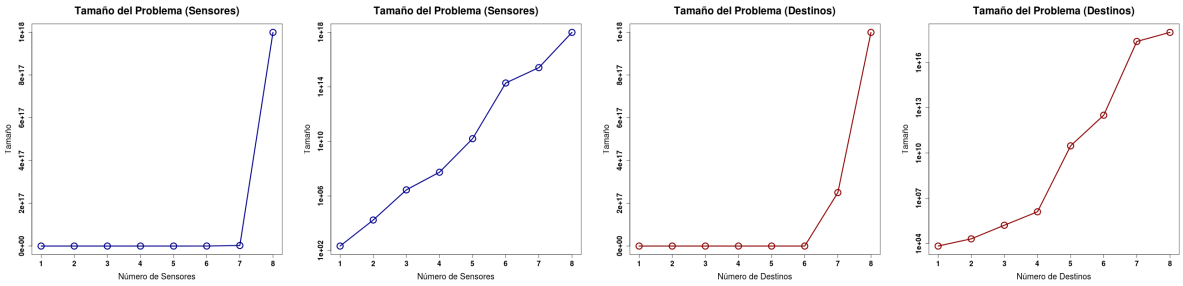
$$P_{S_N D_M} = P_{N,M,R_1} + P_{N,M,R_2} + \ldots + P_{N,M,R_{K_M}} = \sum_{i=1}^{K_M} P_{N,M,R_i} = 1,0 \tag{1}$$

Con esta representación y suponiendo que se selecciona sólo una ruta (probabilidad igual a 1,0 para la seleccionada y 0,0 para el resto) por cada destino del sensor, se tiene que la complejidad del problema para el sistema de representación escogido es de $1,34 \times 10^{128}$ configuraciones distintas, denotando la explosión combinatoria que se produce a partir del número relativamente pequeño de entradas, salidas y sensores, tal como se ejemplifica en la Fórmula 2, en dónde R_{ij} es el número total de rutas entre el sensor i y el destino j, mientras que N representa al número total de sensores y M al de destinos.

$$C = \prod_{i=1}^{N} \prod_{j=1}^{M} R_{ij} = 1,34 \times 10^{128} \qquad N = 28, M = 8 \tag{2}$$

Para comprobar que el problema obedece a un crecimiento exponencial se incluyen las gráficas que se obtienen al incrementar el número de sensores asociados a los *Red Swarms* manteniendo constante

el número de entradas y salidas de la ciudad (figuras 6(a) y 6(b)) y las correspondientes al incremento del número de destinos (salidas de la ciudad) manteniendo constante el número de sensores y entradas (figuras 6(c) y 6(d)).



(a) Sensores (b) Sensores (log) (c) Destinos (d) Destinos (log)

Figura 6: Dimensión del problema respecto al número de sensores y destinos

3.2. Función de *Fitness*

La evaluación de los individuos de la población viene dada por la función de *fitness* la cual requiere un especial cuidado al definirla porque puede influir notablemente en la evolución y convergencia del algoritmo. La forma de evaluar cada configuración consistirá en realizar la simulación del escenario aplicando los cambios de ruta a los vehículos según los valores de probabilidad propios del individuo. Luego de un tiempo de simulación preestablecido, se obtendrán a partir de los ficheros de salida de *SUMO* los resultados de la simulación, calculando en base a ellos el valor de *fitness* del individuo.

En la Fórmula 3 se presenta la función de *fitness* propuesta, la cual consiste en la suma de cuatro términos. Cada término cuenta con un parámetro α_i con el cual se definirá la importancia o peso que se le otorga al mismo.

$$F = \alpha_1(N - n_{viajes}) + \alpha_2 \frac{\sum t_{viajes}}{N} + \alpha_3 \frac{\sum t_{retraso}}{N} \tag{3}$$

El primer término representa a la cantidad de viajes que se completan durante el tiempo de estudio, siendo N el número total de vehículos y n_{viajes} el número de vehículos que han completado su itinerario. El segundo término hace referencia al tiempo medio que han empleado los vehículos en completar su viaje, mientras que el tercero, representa el retraso medio que han sufrido los vehículos en el momento de ser emitidos por el simulador. Esta situación se produce cuando la calle por la cual debe ingresar el vehículo al escenario se encuentra obstruida por vehículos en un atasco y se contabiliza como un tiempo de retraso extra en el viaje del vehículo. Obsérvese que los tiempos presentes en la fórmula son obtenidos a partir de los ficheros de salida generados por *SUMO* y se encuentran medidos en segundos.

3.3. Operadores de Recombinación

A continuación se establecieron dos operadores de recombinación diferentes con el objetivo de combinar las buenas soluciones parciales obteniendo las siguientes generaciones durante la ejecución del algoritmo. Los mismos serán evaluados durante la experimentación para seleccionar el más apropiado para el problema.

3.3.1. Cruce por Sensores

Como primera opción para el operador de recombinación se optó por el cruce de dos puntos. Con el mismo es posible seleccionar la configuración de las probabilidades pertenecientes a un subconjunto de los sensores de un individuo y combinarlas con las configuraciones de los sensores del otro y viceversa, dando lugar a dos descendientes. Los individuos objeto del cruce se escogen de forma aleatoria, siguiendo una distribución uniforme, de igual modo que los puntos de cruce para cada ejecución del operador. De esta forma se garantiza, con alta probabilidad, que los resultados obtenidos serán diferentes a pesar de

que para el cruce se hayan seleccionado los mismos individuos en más de una ocasión. En la Figura 7 se presenta un ejemplo del cruce por sensores de dos individuos en donde se han escogido como puntos de cruce desde el sensor S_4 hasta el sensor S_6.

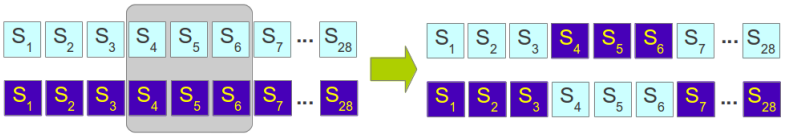


Figura 7: Cruce de individuos en base a la configuración de sus sensores

3.3.2. Cruce por Destinos

Como alternativa al operador de cruce por sensores, se evaluará el operador de cruce por destinos. En este caso, se seleccionan aleatoriamente un subconjunto de destinos los cuales se intercambian entre ambos individuos de la población dando lugar a dos nuevos descendientes. De esta manera tras el cruce se conservan la configuración de las rutas completas que los vehículos seguirán desde un origen hacia un destino a través de los *Red Swarm* a diferencia del caso anterior en el cual las rutas variaban notablemente porque se modificaba la configuración de todo el sensor. En la Figura 8 se visualiza un ejemplo de cruce por destinos en el que se han seleccionado los destinos D_2 y D_3 como elementos de cruce. De esta forma el primer individuo conserva las rutas para los destinos D_1 y desde D_4 a D_8, mientras que reemplaza las de los destinos D_2 y D_3 con las provenientes del otro. Lo mismo ocurre con el segundo individuo generándose así dos nuevos potenciales descendientes para la siguiente generación.

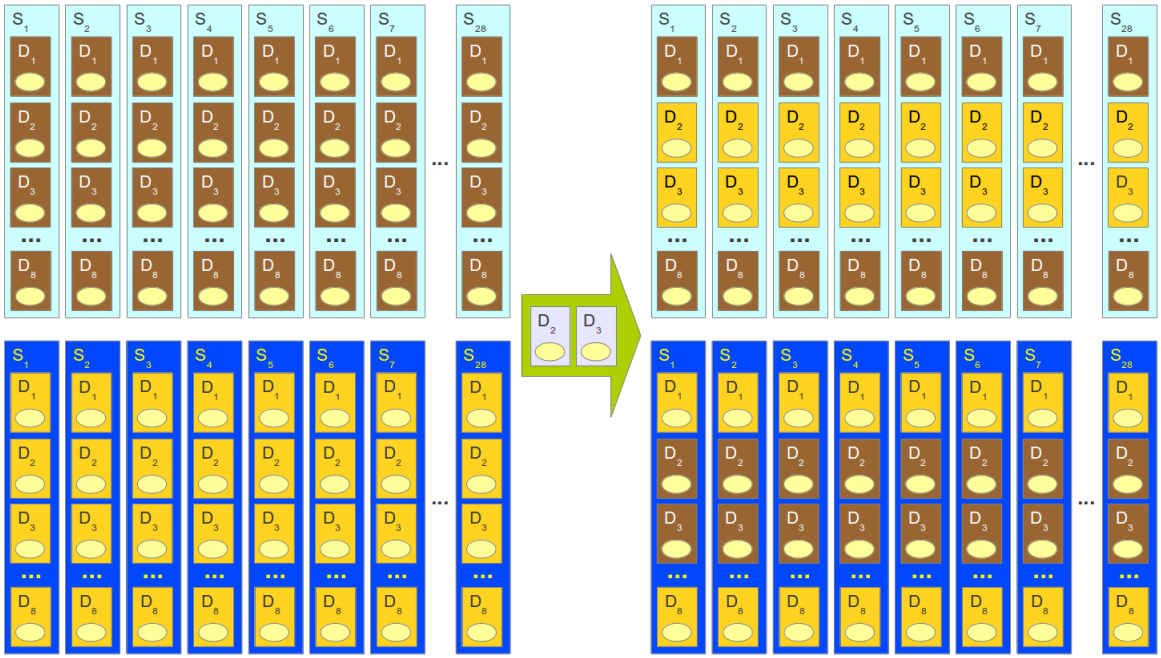
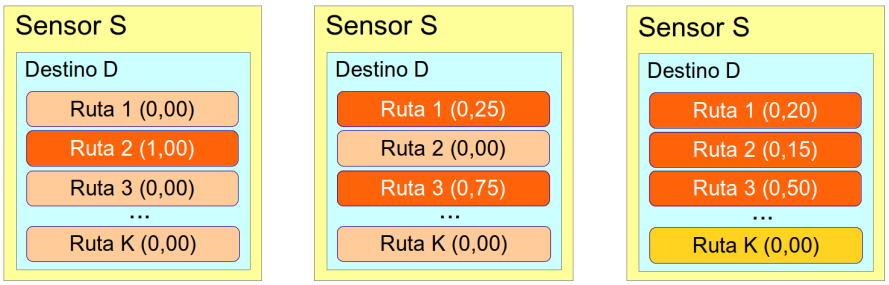


Figura 8: Cruce de individuos en base a la configuración de sus destinos

3.4. Operadores de Mutación

La mutación en un algoritmo evolutivo evita un pronta convergencia a un mínimo local añadiendo diversidad genética a la población. Este operador se aplica según la probabilidad de mutación, definida como un parámetro del algoritmo y actúa directamente modificando la probabilidad de que las rutas asociadas a cada sensor de los *Red Swarm* sea asignada a un vehículo que es detectado por el mismo cuando se dirige rumbo hacia su destino final.

En la Figura 9 se representan los posibles modos de cambio de probabilidad que serán evaluados: **Mutación Binaria** (Figura 9(a)), en la cual siempre se selecciona una sola ruta asignándole una probabilidad de 1,0; **Mutación 0,25** (Figura 9(b), en dónde la probabilidad se asigna en cuatro porciones fijas de valor 0,25. Obsérvese la posibilidad de repetición, es decir que si se asigna un valor de 0,25 tres veces a la misma ruta, se obtienen los 0,75 asignados a la Ruta 3 en el ejemplo de la figura; y por último, **Mutación Decimal** (Figura 9(c)), en donde cada ruta puede tomar cualquier valor como probabilidad de ser escogida en el rango 0,0 – 1,0 y sujeta a la condición descrita en la Fórmula 1 con anterioridad.



(a) Mutación Binaria (b) Mutación 0,25 (c) Mutación Decimal

Figura 9: Tipos de Mutación

Los diferentes operadores de mutación que han sido propuestos para ser evaluados se describen a continuación. Cada uno de ellos determina cuales rutas del individuo serán modificadas mientras que la magnitud del cambio (Binaria, 0,25 o Decimal) vendrá dada por el tipo de mutación seleccionado.

3.4.1. Mutación de Todos los Destinos en un Sensor (MTDUS)

Este operador selecciona aleatoriamente una configuración completa de un sensor del individuo y modifica las probabilidades de las rutas de cada uno de los destinos empleando uno de los tipos de mutación propuestos. De este modo si se emplea Mutación Binaria se seleccionará solo una de las rutas para el sensor y destino, asignándole una probabilidad de 1,0. Si se emplea Mutación 0,25, se repartirá de forma aleatoria entre las rutas cuatro veces el valor de 0,25, etc. En el Algoritmo 2 se presenta el pseudocódigo de este operador y en la Figura 10(a), un ejemplo en el que se selecciona la configuración del sensor S_4 para ser modificada. Luego se recorren cada uno de los destinos, desde D_1 hasta D_8, modificando las probabilidades de todas las rutas disponibles para los vehículos que se dirigen hacia estos y que pasan por sensor S_4. Al realizar tantos cambios simultáneamente se espera que este operador favorezca una rápida convergencia del algoritmo.

Algoritmo 2 Algoritmo MTDUS

```
sensors[] ← getSensors(individual)
sensor ← getRND(sensors[])
destinations[] ← getDestinations(sensor)
for all d in destinations[] do
  routes[] ← getRoutes(d)
  for all r in routes[] do
    changeProbabilities(r, type) {type = Binaria, 0,25 or Decimal}
  end for
end for
return individual
```

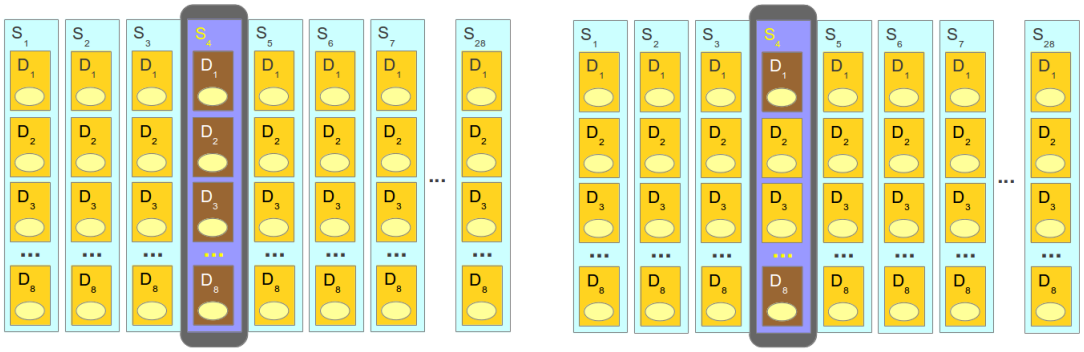
3.4.2. Mutación de Varios Destinos en un Sensor (MVDUS)

Este operador selecciona aleatoriamente una configuración de un sensor del individuo de igual modo que el operador anterior, pero en este caso la modificación de las probabilidad de las rutas se realiza sólo para un subconjunto de destinos los cuales se seleccionan también de forma aleatoria. En este caso la

convergencia del algoritmo no debería ser tan rápida como en el caso anterior, permitiendo así explorar más soluciones adyacentes a la actual. En el Algoritmo 3 se visualiza el pseudocódigo de este operador y en la Figura 10(b), los elementos afectados en el individuo cuando este operador selecciona el sensor S_4 y las probabilidades de las rutas que se modificarán, corresponden a las de los vehículos que se dirigen a los destinos D_1 y D_8.

Algoritmo 3 Algoritmo MVDUS

```
sensors[] ← getSensors(individual)
sensor ← getRND(sensors[])
destinations[] ← getDestinations(sensor)
for all d in destinations[] do
  if mutateDestination(RND) then
    routes[] ← getRoutes(d)
    for all r in routes[] do
      changeProbabilities(r, type) {type = Binaria, 0,25 or Decimal}
    end for
  end if
end for
return individual
```



(a) MTDUS: Todos los Destinos de Un Sensor

(b) MVDUS: Varios Destinos en Un Sensor

Figura 10: Operadores de Mutación (I)

3.4.3. Mutación de un Destino en Todos los Sensores (MUDTS)

Es este caso el operador afecta siempre a todos los sensores, modificando solamente las probabilidades de las rutas de los vehículos que se dirigen a un mismo destino, el cual se escoge de forma aleatoria. En el Algoritmo 4 se presenta el pseudocódigo de este operador y en la Figura 11(a), un ejemplo de la aplicación del mismo a un individuo. En este caso se ha seleccionado el destino D_2 para ser modificado en todos los sensores. Obsérvese que el destino se selecciona aleatoriamente una sola vez y las rutas que cambian en las configuraciones de todos los sensores son las disponibles para ese mismo destino.

3.4.4. Mutación de Varios Destinos en Todos los Sensores (MVDTS)

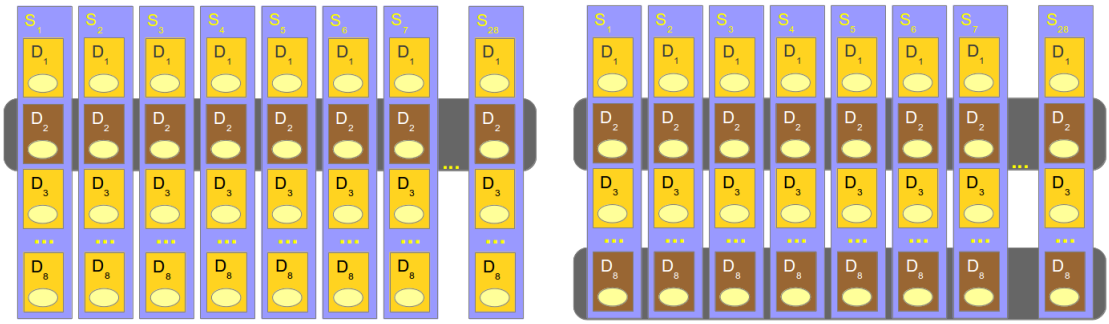
De igual forma que el operador anterior, se modifican las probabilidades de las rutas en todos los sensores, pero en este caso se selecciona aleatoriamente un subconjunto de destinos. En el Algoritmo 5 se presenta el pseudocódigo de este operador y en la Figura 11(b), un ejemplo de la aplicación de este operador a un individuo en el cual los destinos seleccionados aleatoriamente son D_2 y D_8.

Algoritmo 4 Algoritmo MUDTS

```
sensors[] ← getSensors(individual)
selected ← getRNDDestination(individual) {one destination randomly selected}
for all s in sensors[] do
  destinations[] ← getDestinations(s)
  if d = selected then
    routes[] ← getRoutes(d)
    for all r in routes[] do
      changeProbabilities(r, type) {type = Binaria, 0,25 or Decimal}
    end for
  end if
end for
return individual
```

Algoritmo 5 Algoritmo MVDTS

```
sensors[] ← getSensors(individual)
selected[] ← getRNDDestinations(individual) {subset randomly selected}
for all s in sensors[] do
  destinations[] ← getDestinations(s)
  for all d in destinations[] do
    if d in selected[] then
      routes[] ← getRoutes(d)
      for all r in routes[] do
        changeProbabilities(r, type) {type = Binaria, 0,25 or Decimal}
      end for
    end if
  end for
end for
return individual
```



(a) MUDTS: Un Destino en Todos los Sensores

(b) MVDTS: Varios Destinos en Todos los Sensores

Figura 11: Operadores de Mutación (II)

3.4.5. Mutación de un Destino en un Sensor (MUDUS)

Con este último operador propuesto se busca explorar las soluciones muy cercanas a la actual de modo que el algoritmo no se aleje demasiado de la mejor solución encontrada hasta el momento. Esto tiene como inconveniente que la convergencia puede llegar a ser muy lenta. En la Figura 12 se presenta un ejemplo de este operador en el cual se han seleccionado aleatoriamente el sensor S_4 y el destino D_2 para ser objeto del cambio de probabilidades en sus rutas. Como consecuencia los vehículos que pasen por el sensor S_4 y de dirijan a D_2 tomarán un camino diferente en este nuevo individuo recién generado que en los del resto de la población.

Algoritmo 6 Algoritmo MUDUS

```
sensors[] ← getSensors(individual)
sensor ← getRND(sensors[])
destinations[] ← getDestinations(sensor)
destination ← getRND(destinations[])
routes[] ← getRoutes(destination)
for all r in routes[] do
  changeProbabilities(r, type) {type = Binaria, 0,25 or Decimal}
end for
return individual
```

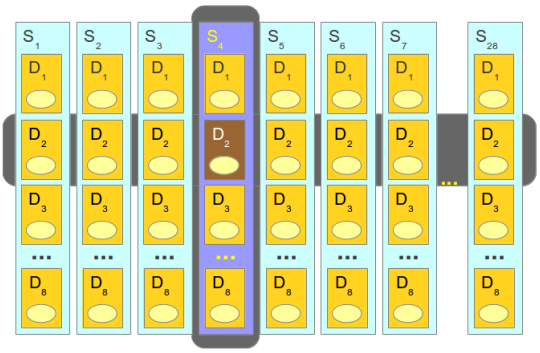


Figura 12: MUDUS: Mutación de Un Destino en Un Sensor

3.5. Cambio Dinámico de la Ruta de los Vehículos

Para que las distintas configuraciones puedan ser simuladas y evaluadas es necesario comunicar con *SUMO* con el objetivo de modificar las rutas de los vehículos. Para ello se ha utilizado la extensión *TraCI* que consiste en una serie de módulos *Python* distribuidos junto con *SUMO* con el fin de facilitar el acceso a todas las variables de la simulación, incluido el estado de las carreteras, de los semáforos y de los vehículos.

A continuación se presenta en el Algoritmo 7 el pseudocódigo correspondiente a la rutina desarrollada para la búsqueda y cambio de rutas de los vehículos durante la simulación. Este código se ejecutará cada vez que un vehículo entre en una calle que conduce a uno de los semáforos asociado a un punto *Red Swarm*, mediante la activación del sensor incluido en la misma, emulando así el enlace radial que se realizaría en una implementación real.

Algoritmo 7 Búsqueda y cambio de rutas de los vehículos

```
currentRoad ← getRoad(vehicle)
if isDestination(currentRoad) then
  route ← currentRoad
else
  availableRoutes ← getDestinationRoutesForSensor(currentRoad)
  if availableRoutes = [] then
    availableRoutes ← getSensorRoutesForSensor(currentRoad)
    route ← getRouteByProbability(availableRoutes)
  end if
end if
setNewRoute(route, vehicle)
```

En primer lugar se obtiene la calle en la que se encuentra actualmente el vehículo y si se trata de su destino se mantiene la ruta actual ya que el vehículo está abandonando la ciudad. Por el contrario, si el vehículo no se encuentra en la calle final de su itinerario se obtienen todas las rutas hacia el destino final del mismo desde el sensor en el que se encuentra (en la calle actual) y se almacenan en *availableRoutes*.

Si no se ha encontrado ninguna, lo que significa que desde esta posición no se puede alcanzar el destino final del vehículo, se obtienen en su lugar las rutas hacia los otros sensores. Luego, se selecciona una de estas rutas según la probabilidad asociada a cada una de ellas. Finalmente, se asigna al vehículo la nueva ruta obtenida, modificando su itinerario.

4. Experimentación y Resultados Obtenidos

Para considerar todas las situaciones posibles, se han definido tres escenarios de trabajo ($Scen1$, $Scen2$ y $Scen3$) que obedecen a una secuencia aleatoria diferente utilizada por *SUMO* para la generación y emisión de vehículos en cuanto a tipos y tiempos de entrada a Málaga como escenario para la simulación. Luego para evaluar cada escenario, se ha realizado una simulación sobre cada una de ellos obteniendo el comportamiento que presentan cuando no se interfiere externamente sobre los itinerarios seguidos por los vehículos. En estas condiciones, las rutas corresponderán a las calculadas en la construcción de Málaga utilizando el camino más rápido. Como se mencionó anteriormente a estas simulaciones en las cuales no existen puntos *Red Swarm* en los semáforos las llamaremos **Simulación SUMO**.

El número de viajes completados, el tiempo de simulación y la evaluación que la función de *fitness* devuelve para cada escenario de la Simulación SUMO se presentan a continuación en la Tabla 3. Mientras que el resto de los valores obtenidos se pueden consultar en la Tabla 14 incluida en el Apéndice A y donde se exponen los totales, valores medios, desviación estándar y mediana para los valores de tiempos de entrada (tiempo de retraso que sufre un vehículo al entrar en la simulación), número de esperas (número de veces que un vehículo sufre una detención), tiempo de viaje (tiempo que emplea el vehículo en recorrer el itinerario) y distancia recorrida (longitud del itinerario).

Todas las ejecuciones de este trabajo han sido realizadas en un AMD Phenom^{TM} II X4 955 en un sistema GNU/Linux (Kernel 3.2.0-30) con 4GB de RAM.

Tabla 3: Características de los tres escenarios de Simulación SUMO

Métrica	Scen1	Scen2	Scen3
Viajes completados	800	800	800
Tiempo de simulación (s)	1283,0	1253,0	1385,0
Valor de Fitness	532,9	530,8	549,0

Además, con los datos generados por las ejecuciones de los tres escenarios se ha obtenido la gráfica de la Figura 13, en la que se visualiza la densidad de tráfico en la ciudad durante la simulación. En la misma se observa el comportamiento de los 800 vehículos los cuales ingresan a la ciudad durante los primeros 100 segundos en dónde se produce un pico en el número de vehículos ya que la ciudad no es capaz de conducirlos a destino al mismo ritmo que van ingresando. Luego las curvas van disminuyendo de forma exponencial, siendo la cota temporal máxima para los tres escenarios inferior a los 1400 segundos (1283.0, 1253.0 y 1385.0 para $Scen1$, $Scen2$ y $Scen3$ respectivamente).

Dado que la función de *fitness* penaliza fuertemente una solución en la que permanecen vehículos dentro del la ciudad al finalizar la simulación, se optó por fijar el tiempo máximo de simulación en 1800 segundos (30 minutos). Esta decisión se basa en que el objetivo principal es minimizar el tiempo medio de viaje por lo que algunos pocos vehículos podrían exceder los 1400 segundos de viaje (cota superior de la Simulación SUMO) si esto favorece a una disminución en los tiempos para el resto de modo que la media total en las simulaciones utilizando Red Swarm sea menor que la de las Simulaciones SUMO. Esto favorecerá a que el algoritmo continúe su convergencia incluyendo esta solución dentro de la población aproximándose al objetivo propuesto. Por otro lado el tiempo máximo de simulación no se puede prolongar en demasía porque como se debe realizar una simulación completa cada vez que se calcula el *fitness* de un individuo, un aumento de este tiempo repercutirá negativamente en el tiempo empleado por el algoritmo en obtener una generación de individuos y por consiguiente, en hallar una solución.

El tamaño de población con el que se trabajará será pequeño debido al elevado tiempo que toma calcular la función de *fitness* y dado que sólo se va a realizar una recombinación por generación (2 descendientes) la probabilidad del mismo será 1,0. De este modo el algoritmo evolutivo a utilizar será un **10+2**. A continuación a los descendientes se les aplicará el operador de mutación con una probabilidad

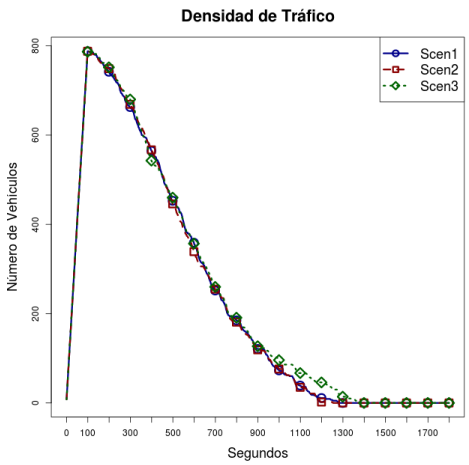


Figura 13: Densidad de tráfico para la Simulación SUMO (3 escenarios)

de 0,75 la cual se ha determinado de forma empírica teniendo en cuenta los tiempos de evolución y convergencia.

Por otro lado, el número total de generaciones dependerá de la convergencia del algoritmo y de la estabilidad de la población. Empíricamente se ha determinado que los valores que mejor se ajustan a la condición de parada son un número máximo de 5000 generaciones o que la mejor solución encontrada se mantenga invariante durante al menos 1000 generaciones.

Finalmente se definieron los valores para los parámetros α_i de la función de *fitness* penalizando con la mitad del tiempo de simulación a cada vehículo que no abandona la ciudad antes de que haya terminado la simulación (α_1). Además, debido a que α_2 y α_3 multiplican el número de medio de segundos que los vehículos permanecen en la ciudad (circulando o retrasados en las entradas), se les ha asignado a ambos parámetros el valor uno, de modo que estos tiempos guarden una relación con el valor de devuelto por la función de *fitness* tal que, un segundo equivalga a una unidad en dicha función.

Un resumen con los parámetros de configuración del algoritmo evolutivo se presenta a continuación en la Tabla 4.

Tabla 4: Parámetros Algoritmo Evolutivo

Tiempo de Simulación (s)	1800
Tamaño población	10
Número de descendientes	2
Probabilidad de cruce	1,0
Probabilidad de mutación	0,75
Máximo número de generaciones	5000
α_1	900
α_2	1
α_3	1

Antes de comenzar la optimización de las rutas para los puntos *Red Swarm* se llevaron a cabo tres simulaciones (una sobre cada escenario) utilizando como configuración, una especialmente realizada de forma tal que todas las rutas tengan la misma probabilidad de ser asignadas. De este modo los itinerarios que los vehículos sigan dentro de la simulación, vendrán dados por la selección aleatoria de los mismos. El objetivo de esta etapa previa a la optimización es obtener resultados previos sobre el comportamiento de los tres escenarios para ser comparados con las soluciones obtenidas en el proceso de optimización. A esta simulación se la ha denominado **Simulación Random** y los resultados obtenidos se comparan con la Simulación SUMO en la Tabla 5.

Obsérvese que en todos las escenarios de la Simulación Random quedan vehículos sin completar su itinerario luego de los 1800 segundos de simulación. Este es el principal motivo del valor elevado de *fitness* de estas soluciones comparados con los de la Simulación SUMO. El resto de las métricas de la simulación no se tienen en cuenta al no tener sentido en la comparativa porque la Simulación Random carece de la

Tabla 5: Comparación entre Simulación SUMO y Simulación Random

Métrica	Simulación SUMO			Simulación Random		
	Scen1	Scen2	Scen3	Scen1	Scen2	Scen3
Viajes completados	**800**	**800**	**800**	489	444	472
Tiempo de simulación (s)	**1283,0**	**1253,0**	**1385,0**	1800,0	1800,0	1800,0
Valor de Fitness	**532,9**	**530,8**	**549,0**	280444,1	320927,6	295766,5

totalidad de valores al quedar vehículos dentro de la ciudad pendientes de finalizar sus trayectos.

Con los parámetros del algoritmo evolutivo ya definidos se procedió a evaluar los operadores de recombinación y mutación para así poder seleccionar los que mejores resultados producen para este problema. Las simulaciones realizadas se presentan en la Tabla 6 en dónde se puede observar que se ha seleccionado $Scen1$ para realizar las simulaciones de la totalidad de los operadores dada la suposición de que éstos se comportan de manera similar en todos los escenarios al tratarse del mismo tipo problema. Posteriormente se comprobaron que los operadores que mejores resultados ofrecieron en $Scen1$, también lo hacían con $Scen2$ y $Scen3$, como se desprende también de la Tabla 6.

Tabla 6: Ejecuciones para la evaluación de operadores

Escenario	Cruce	Mutación	Generaciones	Fitness
Scen1	Sensor	MTDUS Binaria	3000	**578,7**
		MTDUS 0,25	5000	129345,3
		MUDTS Binaria	3000	622,0
		MVDTS Binaria	3000	632,0
	Destino	MTDUS Binaria	3000	**601,6**
		MTDUS 0,25	3000	105977,9
		MUDTS Binaria	3000	597,5
Scen2	Sensor	MTDUS Binaria	3000	613,0
		MUDTS Binaria	3000	**605,9**
	Destino	MTDUS Binaria	3000	642,6
		MUDTS Binaria	3000	**590,1**
Scen3	Sensor	MTDUS Binaria	3000	**610,0**
		MUDTS Binaria	3000	652,7
	Destino	MTDUS Binaria	3000	**599,9**
		MUDTS Binaria	3000	609,0

De los resultados obtenidos se dedujo que ambas funciones de cruce se comportan de forma similar en cuanto a la convergencia del algoritmo. Sin embargo, los operadores de mutación, a excepción de la mutación binaria de Todos los Destinos de un Sensor (MTDUS), necesitan demasiadas generaciones para converger. Incluso la Mutación 0,25 no fue capaz encontrar una solución en la que los 800 vehículos abandonen la ciudad dentro de los 1800 segundos de simulación luego de 5000 generaciones, razón por la cual no se ha incluido la evaluación de la Mutación Decimal por ser una extensión de ésta que adolece de un comportamiento mas lento aún respecto a la velocidad de convergencia.

Dado que se observó que lo que más esfuerzo demandaba al algoritmo era sobrepasar el escalón que se produce en la función de *fitness* cuando se obtiene una solución en la cual los 800 vehículos terminan su itinerario, se utilizará una función mixta de mutación. El algoritmo comenzará utilizando la Mutación MTDUS Binaria que es la que converge en menos generaciones provocando una variación de *fitness* mas rápida que el resto. Una vez superado el escalón, se utilizará otro operador, de convergencia más lenta, para evitar caer en los mínimos locales y explorar de forma más minuciosa el subespacio de búsqueda en el que se encuentra la población en el momento del cambio.

Con motivo de seleccionar experimentalmente cual de los operadores de mutación se comporta mejor una vez que se ha alcanzado el escalón en la función de *fitness*, se realizaron los experimentos cuyos resultados se presentan en la Tabla 7. Obsérvese que la columna Escalón hace referencia a la generación en la cual se conmuta de operador de mutación y que es este caso se han tomado como punto de partida los escalones de las mejores ejecuciones del experimento anterior (generaciones 350 y 575). Además para estas ejecuciones se ha impuesto un número máximo de 3000 generaciones ya que no estamos realizando una optimización si no un test de comportamiento de operadores y porque además se pudo comprobar que la mayoría de las ejecuciones se estabilizaban antes de alcanzar este valor.

De los resultados del experimento se observa que la combinación que mejor resultados ofrece es el

Tabla 7: Ejecuciones con mutación mixta (*Scen*1)

Cruce	Segunda Mutación	Generaciones	Escalón	Fitness
Sensor	MTDUS Binaria	3000	350	578,7
	MVDUS Binaria	2350	350	550,8
	MUDUS Binaria	2350	350	**536,9**
	MTDUS 0,25	2350	350	634,7
	MUDTS Binaria	2350	350	594,5
	MVDTS Binaria	2350	350	650.3
	MUDTS 0,25	2350	350	639.9
Destino	MTDUS Binaria	3000	575	601,6
	MVDUS Binaria	2575	575	590,1
	MUDUS Binaria	2575	575	**589,2**
	MTDUS 0,25	2575	575	689,5
	MUDTS Binaria	2575	575	655,9
	MVDTS Binaria	2575	575	729,1
	MUDTS 0,25	2575	575	722,5

Cruce por Sensores, utilizando como operador de mutación hasta que se alcanza el escalón a la mutación **MTDUS Binaria**. Luego el algoritmo conmutará a la mutación **MUDUS Binaria** utilizándola hasta finalizar su ejecución.

El siguiente paso, disponiendo ya del algoritmo evolutivo plenamente afinado, ha consistido en comenzar el proceso de optimización de Málaga mediante múltiples ejecuciones del algoritmo en búsqueda de obtener el mejor resultado (menor valor de *fitness*) para cada uno de los tres escenarios por separado. Queda fuera del ámbito de este trabajo la realización de un análisis estadístico del algoritmo, dándose más prioridad a la optimización propuesta.

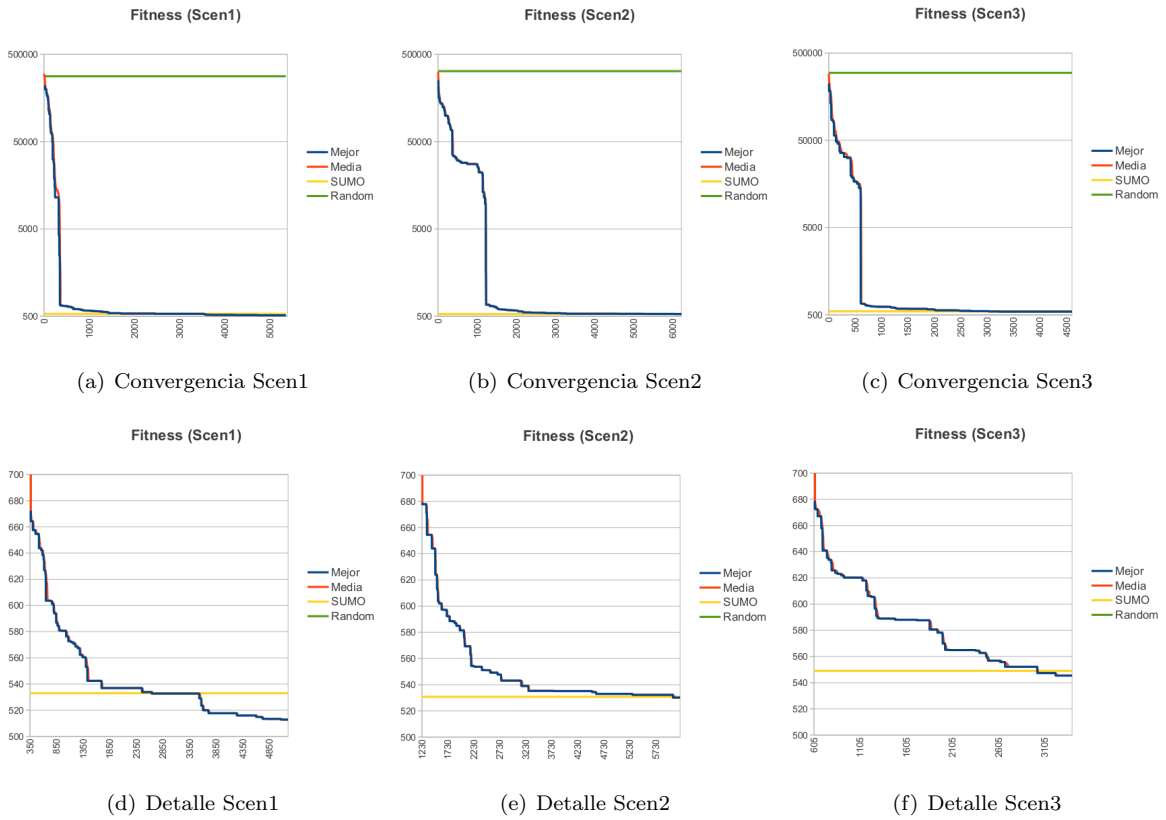
En la Tabla 8 se presentan los resultados obtenidos, ordenados según la valuación obtenida desde la función de *fitness*, desde el peor valor hasta el mejor para cada uno de los escenarios. Obsérvese que las simulaciones realizadas en esta etapa se las ha denominado **Simulación Red Swarm**.

Tabla 8: Ejecuciones utilizando Simulación Red Swarm

Escenario	Fitness Original	Generaciones	Escalón	Fitness
Scen1	532,9	5338	1380	605,0
		4757	757	549.9
		5350	350	**512,7**
Scen2	530,8	2000	–	47522,4
		3000	–	15060,5
		3950	950	611,3
		5475	2475	570,6
		5019	1019	560,5
		5454	454	559,3
		4448	448	550,1
		5424	1424	546,5
		6228	1228	**530,1**
Scen3	549,0	5024	1024	579,2
		4653	653	561,7
		4605	605	**545,4**

En la Figura 14 se visualizan las gráficas que representan la convergencia del algoritmo evolutivo para los tres escenarios del problema. Obsérvese en las figuras 14(a), 14(b) y 14(c) que, como era de esperar, la ejecuciones comienzan con un valor de *fitness* similar a la Simulación Random para ir disminuyendo a medida que las generaciones se suceden. Cada escenario presenta un escalón pronunciado en la función de *fitness* correspondiente a la generación en la cual los 800 vehículos terminan su itinerario dentro de los 1800 segundos simulados. En las gráficas de las figuras 14(d), 14(e) y 14(f), se visualiza en detalle las generaciones próximas al escalón en el cual la Simulación Red Swarm mejora a la Simulación SUMO.

Con las tres soluciones obtenidas se procedió a confeccionar la Tabla 9 en dónde se presentan las diferencias entre las métricas de la Simulación SUMO y la Simulación Red Swarm, mientras que en el Apéndice A, en la Tabla 15, se incluyen el resto de los datos obtenidos. Por último en la Figura 15 se exponen de forma gráfica las mejoras conseguidas encontrándose en la región de la izquierda (magnitudes negativas) los valores en los que *Red Swarm* se comporta mejor que la Simulación SUMO.



(a) Convergencia Scen1 (b) Convergencia Scen2 (c) Convergencia Scen3

(d) Detalle Scen1 (e) Detalle Scen2 (f) Detalle Scen3

Figura 14: Convergencia del algoritmo evolutivo para los tres escenarios

Tabla 9: Comparación entre Simulación SUMO y Simulación Red Swarm

Métrica	Simulación SUMO			Simulación Red Swarm		
	Scen1	Scen2	Scen3	Scen1	Scen2	Scen3
Viajes completados	800	800	800	800	800	800
Tiempo de simulación (s)	1283,0	1253,0	1385,0	**1154,0**	**1201,0**	**1522,0**
Valor de Fitness	532,9	530,8	549,0	**512,7**	**530,2**	**545,4**

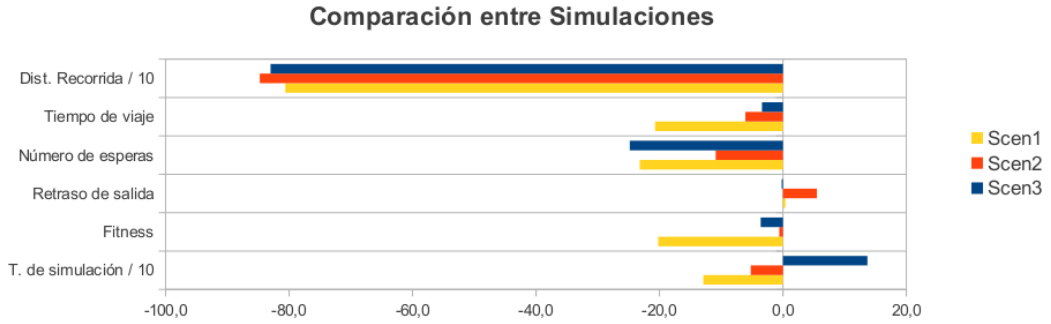
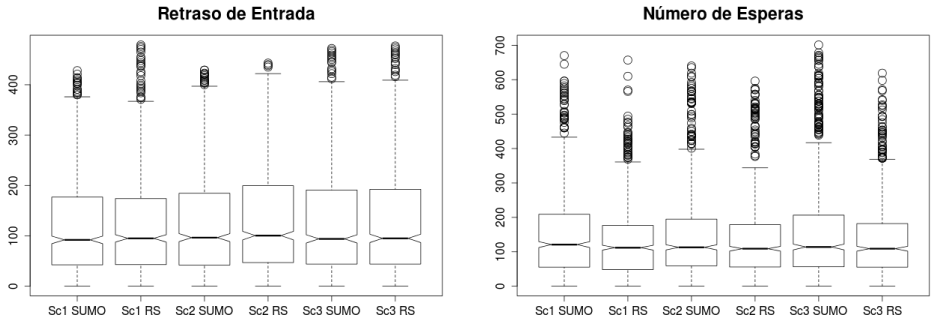


Figura 15: Comparación entre la Simulación SUMO y la Simulación Red Swarm

Obsérvese que las soluciones presentan mejoras enfocadas en el tiempo de viaje, en el número de esperas y en especial en la distancia recorrida. Estas dos últimas métricas no se encontraban incluidas en la función de *fitness* porque no son el principal objetivo de la optimización, sin embargo se han visto también favorecidas en la solución obtenida. La primera, gracias al rápido fluir del tráfico y la segunda por los caminos alternativos generados por los puntos *Red Swarm*. Por otro lado *Scen*1 y *Scen*2 adolecen de un retraso de entrada levemente superior que en el modelo original y *Scen*3 necesita de 137 segundos

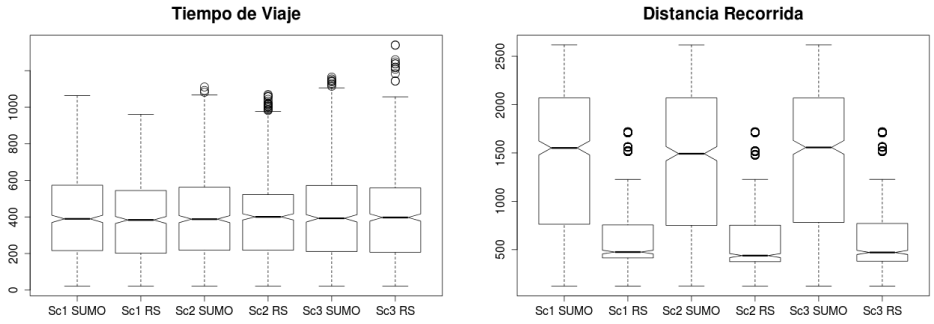
adicionales para finalizar la simulación, sin que esto perjudique la evaluación de los mismos.

En la Figura 16 se presentan los datos de la comparación entre las simulaciones en forma gráficas de cajas en dónde se pueden comparar la mediana, los cuartiles y los valores anómalos para las cuatro métricas consideradas en los tres escenarios del problema.



(a) Retraso de entrada

(b) Número de esperas



(c) Tiempo de viaje

(d) Distancia recorrida

Figura 16: Comparación entre la Simulación SUMO y la Simulación Red Swarm

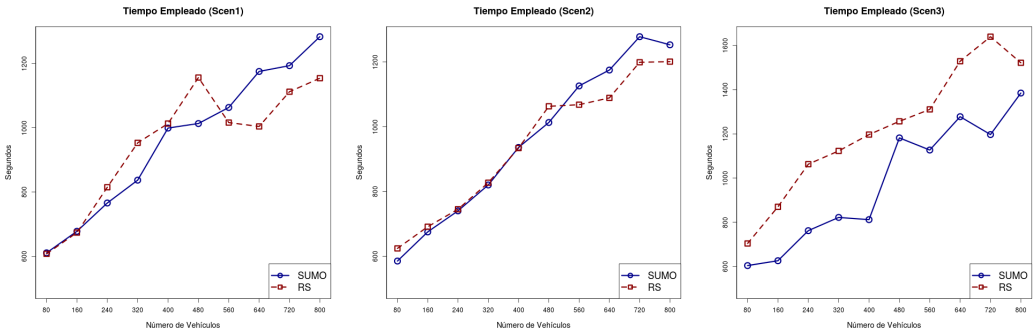
El siguiente experimento a realizar consiste en comparar la evolución del tiempo máximo que necesitan los vehículos para abandonar la ciudad para los tres escenarios en la Simulación SUMO y en la Simulación *Res Swarm*. Para ello se han obtenido los valores partiendo desde un escenario con diez vehículos por entrada (80 vehículos) hasta llegar a los 800 utilizados durante la optimización (100 por entrada) en saltos de 10 en 10 (80 en 80). Los resultados obtenidos que se presentan en la Tabla 10 y en las gráficas de la Figura 17 demuestran que para $Scen1$ y $Scen2$ a bajas concentraciones de vehículos, la Simulación SUMO se comporta levemente mejor que el *Red Swarm*. Esto era lo esperado, ya que el *SUMO* utiliza las rutas generadas por el algoritmo de Dijkstra el cual calcula la ruta más rápida en cada intersección de la ciudad, mientras que la Simulación Red Swarm sólo toma decisiones en cada punto *Red Swarm*.

Sin embargo, a medida que la densidad de tráfico va aumentando, se puede observar como en la Simulación Red Swarm los vehículos finalizan su itinerario en un tiempo inferior a la Simulación SUMO, denotando una mejora frente a las situaciones de congestión que se producen dentro de las calles de Málaga. Para $Scen3$, no se cumple lo observado en los otros dos escenarios, aunque los tiempos medios son mejores (menores) que en el modelo sin optimizar. La explicación se puede deducir observando la Figura 16(c) en dónde se aprecia una serie de valores anómalos para los tiempos de viaje de este escenario en la Simulación Red Swarm (caja en el extremo derecho de la gráfica). Esto se puede interpretar como que unos pocos vehículos han sufrido un tiempo de viaje muy elevado, provocando que la simulación se extienda un tiempo mayor que en el modelo sin optimizar.

Habiendo mejorado los valores de la simulación para los tres escenarios, se procedió a comprobar si las éstas eran también válidas para los escenarios para los cuales no fueron calculadas. Para ello se realizó un **Test de Robustez** que consistió en se ejecutar cada solución obtenida en la Simulación Red Swarm tres veces, cada una de ellas en un escenario distinto.

Tabla 10: Comparación entre tiempos de Simulación SUMO y Red Swarm (segundos)

Vehículos	Scen1		Scen2		Scen3	
	Sim. SUMO	Red Swarm	Sim. SUMO	Red Swarm	Sim. SUMO	Red Swarm
80	**611**	608	**585**	624	**604**	704
160	**678**	674	**675**	691	**626**	870
240	**766**	815	**740**	745	**762**	1063
320	**837**	953	**820**	827	**822**	1123
400	**999**	1013	936	**934**	**812**	1197
480	**1013**	1156	**1013**	1063	**1182**	1257
560	1063	**1016**	1126	**1068**	**1127**	1311
640	1175	**1004**	1175	**1089**	**1278**	1529
720	1193	**1112**	1278	**1199**	**1197**	1640
800	1283	**1154**	1253	**1201**	**1385**	1522



(a) Scen1 (b) Scen2 (c) Scen3

Figura 17: Tiempo de simulación vs Número de vehículos

Los resultados para la función de *fitness* obtenidos se presentan en la Tabla 11 y de forma gráfica en la Figura 18. Obsérvese que la **Solución 1** de la Simulación Red Swarm es la que mejor se comporta, no sólo con la *Scen*1 a partir del cual fue obtenida, sino que también para *Scen*2 y *Scen*3. Otro observación a destacar es que la Solución 3 no es capaz de gestionar el tráfico para *Scen*2 no consiguiendo que todos vehículos finalicen sus itinerarios antes de los 1800 segundos de simulación.

Tabla 11: Fitness obtenidos para el Test de Robustez

Escenario	Simulación Red Swarm			Sim. SUMO
	Solución 1	Solución 2	Solución 3	
Scen1	**512,7**	552,2	585,1	532,9
Scen2	**523,5**	530,2	5103,1	530,8
Scen3	**533,3**	546,4	545,4	549,0

Habiendo determinado cual de las tres soluciones obtenidas es la mejor de forma absoluta, se repitieron los ensayos para *Scen*2 y *Scen*3 utilizando en esta ocasión la **Solución *Red Swarm*** correspondiente a *Scen*1. A continuación se presentan en la Tabla 12 el número de vehículos, el tiempo de simulación y el valor de la función de *fitness* para la simulación de la Solución *Red Swarm* comparada con la Simulación SUMO. En el Apéndice A se incluyen el resto de los valores del experimento dispuestos en la Tabla 16.

Tabla 12: Comparación entre Simulación SUMO y Solución Red Swarm

Métrica	Simulación SUMO			Solución Red Swarm		
	Scen1	Scen2	Scen3	Scen1	Scen2	Scen3
Viajes completados	800	800	800	800	800	800
Tiempo de simulación (s)	1283,0	1253,0	1385,0	**1154,0**	**1164,0**	**1206,0**
Valor de Fitness	532,9	530,8	549,0	**512,7**	**523,5**	**533,3**

En las gráficas de la Figura 19 se presentan nuevamente la comparación de las métricas obtenidas

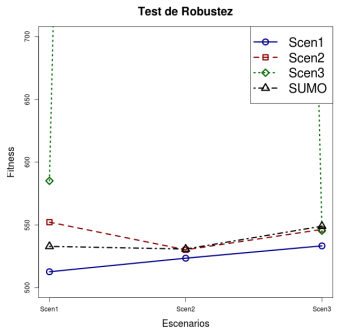
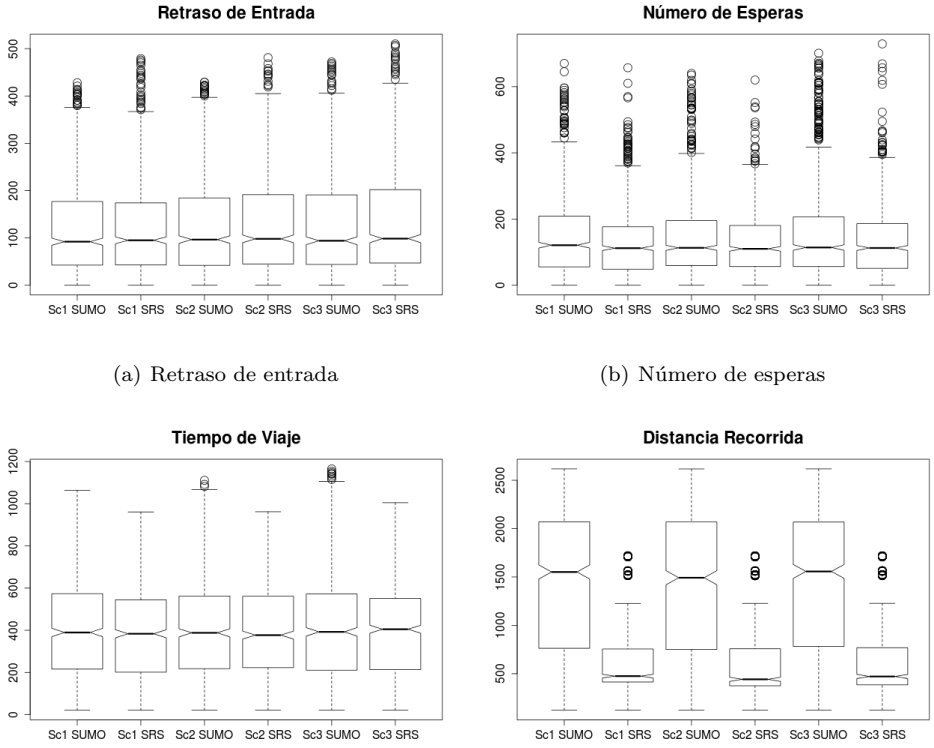


Figura 18: Valores de *fitness* obtenidos en el Test de Robustez

como resultado mediante el uso de diagrama de cajas, en dónde se destaca la ausencia de valores anómalos para los Tiempo de Viaje de las simulaciones con la Solución Red Swarm (Figura 19(c)).



(a) Retraso de entrada

(b) Número de esperas

(c) Tiempo de viaje

(d) Distancia recorrida

Figura 19: Comparación entre la Simulación SUMO y la Solución Red Swarm

En la Figura 20 se presentan las mejoras que se han obtenido en la simulación utilizando la Solución Red Swarm. Obsérvese que se mejoran los tiempos medios de viaje, los tiempos de simulación (los vehículos abandonan la ciudad antes), el número de esperas y sobre todo la distancia recorrida, la cual ha disminuido notablemente respecto a la Simulación SUMO. Por el contrario el tiempo de retraso de entrada nuevamente se presenta algo mayor que en la Simulación SUMO, influido especialmente por los valores anómalos, aislados más allá del tercer cuartil y que no tienen el suficiente peso como para repercutir en los tiempos medios de viaje. Por lo tanto la evaluación general de la simulación utilizando como configuración la Solución Red Swarm es superior a la original, en la cual no se dispone de los puntos de cambio de ruta *Red Swarm*, y que hemos denominado Simulación SUMO.

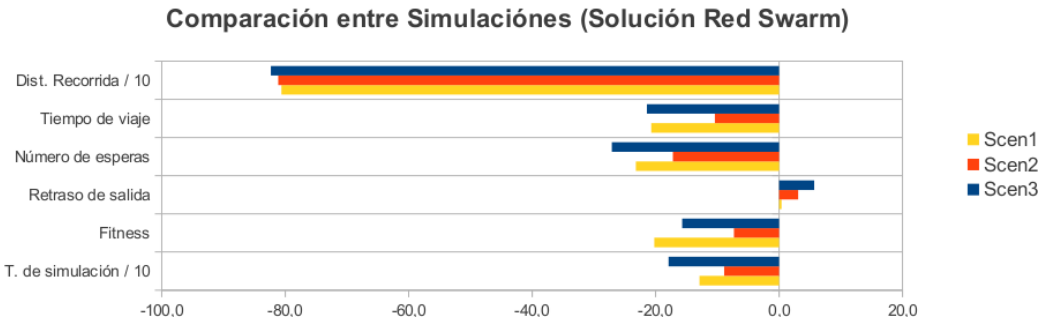


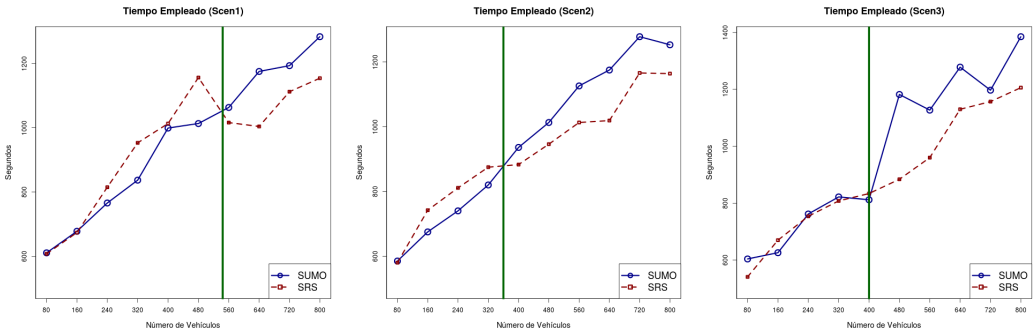
Figura 20: Comparación entre la Simulación SUMO y la Solución Red Swarm

Finalmente, se repitió el estudio de los tiempos de simulación respecto al número de vehículos en la ciudad, pero en este caso utilizando la Solución Red Swarm para simular los tres escenarios. Como se ha volcado en la Tabla 13, ahora el comportamiento de los tres escenarios es consistente en cuando al número de vehículos de modo que cuando no hay congestión, la ciudad gestiona el tráfico de forma más eficiente que cuando se utilizan los semáforos con puntos *Red Swarm* incorporados, pero en el momento en que se sobrepasa un umbral, el cual toma el valor de 480, 320 y 400 vehículos para los escenarios $Scen1$, $Scen2$ y $Scen3$ respectivamente, se comienzan a hacer visibles las mejoras de la simulación utilizando la Solución Red Swarm.

Tabla 13: Comparación entre tiempos de Simulación SUMO y Solución Red Swarm (segundos)

Vehículos	Scen1		Scen2		Scen3	
	Sim. SUMO	Sol. RS	Sim. SUMO	Sol. RS	Sim. SUMO	Sol. RS
80	611	**608**	585	**580**	604	**541**
160	678	**674**	**675**	742	**626**	670
240	**766**	815	**740**	811	762	**754**
320	**837**	953	**820**	875	822	**808**
400	**999**	1013	936	**883**	**812**	834
480	**1013**	1156	1013	**946**	1182	**884**
560	1063	**1016**	1126	**1013**	1127	**960**
640	1175	**1004**	1175	**1019**	1278	**1130**
720	1193	**1112**	1278	**1166**	1197	**1157**
800	1283	**1154**	1253	**1164**	1385	**1206**

A continuación en la Figura 21 se incluyen las gráficas que indican en los puntos de cruce el umbral a partir del cual la Solución Red Swarm comienza a ser efectiva a la hora de lidiar con la congestión del tráfico de la ciudad de Málaga.



(a) Scen1 (b) Scen2 (c) Scen3

Figura 21: Tiempo de simulación vs Número de vehículos

5. Conclusiones y Trabajo Futuro

En este trabajo se han importado los datos cartográficos de la ciudad de Málaga desde *Open Street Map*, se han añadido semáforos y puntos *Red Swarm*, se han calculado 64 itinerarios entre las ocho entradas y ocho salidas existentes, para finalmente definir tres escenarios correspondientes a otras tantas distribuciones de tráfico rodado con el objeto de ser simulados y optimizados atendiendo al tiempo medio de viaje de los vehículos que circulan por los mismos.

Para ello se definió un algoritmo evolutivo partiendo de dos posibles operadores de cruce, cinco posibles operadores de mutación y tres métodos para el cambio de los valores, seleccionando los que mejor comportamiento presentaban para este tipo de problema. Luego se realizaron ejecuciones del algoritmo en busca de una configuración para los puntos *Red Swarm* que mejore las métricas de la simulación original.

La solución obtenida, denominada **Solución Red Swarm**, mejora los tres escenarios del problema en cuanto al tiempo total que necesitan los vehículos para abandonar la ciudad, el tiempo medio de desplazamiento de los mismos, la distancia media recorrida y el número medio de detenciones a lo largo de los trayectos que cada vehículo sigue durante su itinerario.

Los resultados que se presentan tanto en forma de tablas numéricas como en gráficas demuestran que si bien para bajas concentraciones de vehículos el algoritmo de Dijkstra presenta mejores resultados, al seleccionar la ruta más rápida previamente calculada, tal como era previsible. Pero cuando el modelo comienza a recibir un número mayor de vehículos, se presentan situaciones de congestión en las que los vehículos no avanzan rumbo a su destino. En esta situación, la selección de itinerarios alternativos que no se encuentran incluidos dentro del camino más rápido, provoca un aumento en la fluidez del tráfico rodado mejorando las métricas antes mencionadas.

Adicionalmente, unos tiempos de viajes menores así como una menor distancia recorrida, tienen como consecuencia la disminución de las emisiones de gases de efecto invernadero, así como una mejora en la vida cotidiana de las personas que pasarán menos tiempo en atascos, y por último un ahorro en el consumo de combustible que repercutirá directamente en el bolsillo de los habitantes de la ciudad inteligente que proponemos.

Como trabajo futuro se propone abordar la optimización de ciudades de tamaño medio, la obtención de información real del flujo de tráfico por horas para implementarlo como escenario de simulación y el desarrollo de un algoritmo que determine cuales son las mejores intersecciones para la localización de los puntos *red swarm*.

Si bien los resultados obtenidos en este primer enfoque son satisfactorios, en cuanto mejoran a los de los escenarios originales, sería interesante explorar otras alternativas en cuanto a metaheurísticas tales como PSO, ACO u otros Algoritmos Evolutivos.

Por último, para independizar los resultados obtenidos de la implementación del simulador sobre el cual se basan los resultados, sería deseable el empleo de otro simulador microscópico diferente de *SUMO* como *TRANSIMS (TRansportation ANalysis and SIMulation System)* o *VISSIM*.

Referencias

[1] R. Giffinger, C. Fertner, H. Kramar, R. Kalasek, N. Pichler-Milanovic, and E. Meijers, "Smart cities – Ranking of European medium-sized cities." http://www.smart-cities.eu/download/smart_cities_final_report.pdf. (Consultado el 1 de septiembre de 2012).

[2] "Europe 2020 - Europe's growth strategy - European Commission." http://ec.europa.eu/europe2020/. (Consultado el 1 de septiembre de 2012).

[3] H. K. Lo, E. Chang, and Y. C. Chan, "Dynamic network traffic control," *Transportation Research Part A: Policy and Practice*, vol. 35, no. 8, pp. 721–744, 2001.

[4] D. Krajzewicz, E. Brockfeld, J. Mikat, J. Ringel, C. Rössel, W. Tuchscheerer, P. Wagner, and R. Wösler, "Simulation of modern traffic lights control systems using the open source traffic simulation SUMO," in *Proceedings of the 3rd industrial simulation conference*, vol. 2205, 2005.

[5] J. Sánchez, M. Galán, and E. Rubio, "Applying a traffic lights evolutionary optimization technique to a real case:"Las Ramblas" area in santa cruz de tenerife," *Evolutionary Computation, IEEE Transactions on*, vol. 12, no. 1, pp. 25–40, 2008.

[6] J. Garcia-Nieto, E. Alba, and A. Olivera, "Enhancing the urban road traffic with Swarm Intelligence: A case study of Córdoba city downtown," in *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, pp. 368–373, IEEE, 2011.

[7] P. Ming-bao and Z. Xin-ping, "Traffic Flow Prediction of Chaos Time Series by Using Subtractive Clustering for Fuzzy Neural Network Modeling," in *Intelligent Information Technology Application, 2008. IITA'08. Second International Symposium on*, vol. 1, pp. 23–27, IEEE, 2008.

[8] J. McCrea and S. Moutari, "A hybrid macroscopic-based model for traffic flow in road networks," *European Journal of Operational Research*, vol. 207, no. 2, pp. 676–684, 2010.

[9] D. Braess, "Über ein Paradoxon aus der Verkehrsplanung," *Mathematical Methods of Operations Research*, vol. 12, no. 1, pp. 258–268, 1968.

[10] A. Bazzan and F. Klügl, "Case studies on the Braess paradox: simulating route recommendation and learning in abstract and microscopic models," *Transportation Research Part C: Emerging Technologies*, vol. 13, no. 4, pp. 299–319, 2005.

[11] A. Gomez, G. Diaz, and K. Boussetta, "How Virtual Police Agents can help in the traffic guidance?," in *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*, pp. 360–364, IEEE, 2012.

[12] L. Bieker and D. Krajzewicz, "Evaluation of opening bus lanes for private traffic triggered via V2X communication," in *Integrated and Sustainable Transportation System (FISTS), 2011 IEEE Forum on*, pp. 48–53, IEEE, 2011.

[13] M. Lighthill and G. Whitham, "On kinematic waves. II. A theory of traffic flow on long crowded roads," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 229, no. 1178, pp. 317–345, 1955.

[14] M. Ben-Akiva, M. Bierlaire, H. Koutsopoulos, and R. Mishalani, "Real time simulation of traffic demand-supply interactions within DynaMIT," *Applied optimization*, vol. 63, pp. 19–34, 2002.

[15] H. Haj-Salem, N. Elloumi, S. Mammar, M. Papageorgiou, J. Chrisoulakis, and F. Middelham, "METACOR: A macroscopic modelling tool for urban corridor," in *Towards An Intelligent Transport System. Proceedings of The First World Congress on Applications of Transport Telematics and Intelligent Vehicle-Highway Systems, November 30-3rd December 1994, Paris. Volume 3*, 1994.

[16] D. Gazis, R. Herman, and R. Rothery, "Nonlinear follow-the-leader models of traffic flow," *Operations Research*, pp. 545–567, 1961.

[17] "OpenStreetMap." http://www.openstreetmap.org. (Consultado el 1 de septiembre de 2012).

[18] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO - Simulation of Urban MObility: An Overview," in *SIMUL 2011, The Third International Conference on Advances in System Simulation*, (Barcelona, Spain), pp. 63–68, October 2011.

[19] S. Krauss, P. Wagner, and C. Gawron, "Metastable states in a microscopic model of traffic flow," *Physical Review E*, vol. 55, no. 5, p. 5597, 1997.

[20] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J. Hubaux, "TraCI: an interface for coupling road traffic and network simulators," in *Proceedings of the 11th communications and networking simulation symposium*, pp. 155–163, ACM, 2008.

[21] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

A. Tablas con los Resultados de las Simulaciones Realizadas

Tabla 14: Métricas de los vehículos en los tres escenarios de Simulación SUMO

Escenario	Métrica	Total	Media	Desviación	Mediana
Scen1	Retraso de entrada (s)	98394,0	123,0	103,1	92,0
	Número de esperas	121431,0	151,8	130,3	121,0
	Tiempo de viaje (s)	327902,0	409,9	237,8	389,5
	Distancia recorrida (m)	1163894,5	1454,9	749,0	1551,6
Scen2	Retraso de entrada (s)	99748,0	124,7	102,5	96,5
	Número de esperas	117039,0	146,3	127,4	113,0
	Tiempo de viaje (s)	324883,0	406,1	234,1	388,0
	Distancia recorrida (m)	1152487,7	1440,6	760,3	1492,4
Scen3	Retraso de entrada (s)	101572,0	127,0	107,3	94,0
	Número de esperas	128972,0	161,2	153,3	114,0
	Tiempo de viaje (s)	337627,0	422,0	260,3	392,5
	Distancia recorrida (m)	1177399,7	1471,8	735,8	1557,2

Tabla 15: Comparación entre métricas de Simulación SUMO y Simulación Red Swarm

Scen	Métrica	Simulación SUMO				Simulación Red Swarm			
		Total	Media	Desv.	Mediana	Total	Media	Desv.	Mediana
1	Retraso de entrada (s)	98394,0	**123,0**	103,1	92,0	98745,0	123,4	104,1	95,0
	Número de esperas	121431,0	151,8	130,3	121,0	102876,0	**128,6**	106,4	112,0
	Tiempo de viaje (s)	327902,0	409,9	237,8	389,5	311390,0	**389,2**	214,2	383,5
	Dist. recorrida (m)	1163894,5	1454,9	749,0	1551,6	519009,7	**648,8**	421,6	475,0
2	Retraso de entrada (s)	99748,0	**124,7**	102,5	96,5	104158,0	130,2	104,4	100,5
	Número de esperas	117039,0	146,3	127,4	113,0	108284,0	**135,4**	116,4	109,0
	Tiempo de viaje (s)	324883,0	406,1	234,1	388,0	319960,0	**400,0**	219,2	400,5
	Dist. recorrida (m)	1152487,7	1440,6	760,3	1492,4	474632,5	**593,3**	390,5	438,4
3	Retraso de entrada (s)	101572,0	127,0	107,3	94,0	101436,0	**126,8**	105,3	95,0
	Número de esperas	128972,0	161,2	153,3	114,0	109109,0	**136,4**	114,5	109,0
	Tiempo de viaje (s)	337627,0	422,0	260,3	392,5	334872,0	**418,6**	259,5	397,0
	Dist. recorrida (m)	1177399,7	1471,8	735,8	1557,2	513395,0	**641,7**	432,6	470,8

Tabla 16: Comparación entre métricas de Simulación SUMO y Solución Red Swarm

Scen	Métrica	Simulación SUMO				Solución Red Swarm			
		Total	Media	Desv.	Mediana	Total	Media	Desv.	Mediana
1	Retraso de entrada (s)	98394,0	**123,0**	103,1	92,0	98745,0	123,4	104,1	95,0
	Número de esperas	121431,0	151,8	130,3	121,0	102876,0	**128,6**	106,4	112,0
	Tiempo de viaje (s)	327902,0	409,9	237,8	389,5	311390,0	**389,2**	214,2	383,5
	Dist. recorrida (m)	1163894,5	1454,9	749,0	1551,6	519009,7	**648,8**	421,6	475,0
2	Retraso de entrada (s)	99748,0	**124,7**	102,5	96,5	102266,0	127,8	104.8	98,0
	Número de esperas	117039,0	146,3	127,4	113,0	103288,0	**129,1**	98,1	110,0
	Tiempo de viaje (s)	324883,0	406,1	234,1	388,0	316557,0	**395,7**	213,8	377,0
	Dist. recorrida (m)	1152487,7	1440,6	760,3	1492,4	503425,9	**629,3**	432,2	442,0
3	Retraso de entrada (s)	101572,0	**127,0**	107,3	94,0	106144,0	132,7	111,2	98,5
	Número de esperas	128972,0	161,2	153,3	114,0	107312,0	**134,1**	110,9	112,5
	Tiempo de viaje (s)	337627,0	422,0	260,3	392,5	320509,0	**400,6**	213,7	405,0
	Dist. recorrida (m)	1177399,7	1471,8	735,8	1557,2	518856,8	**648.6**	432.6	471.3

B. Instantáneas de la Simulación

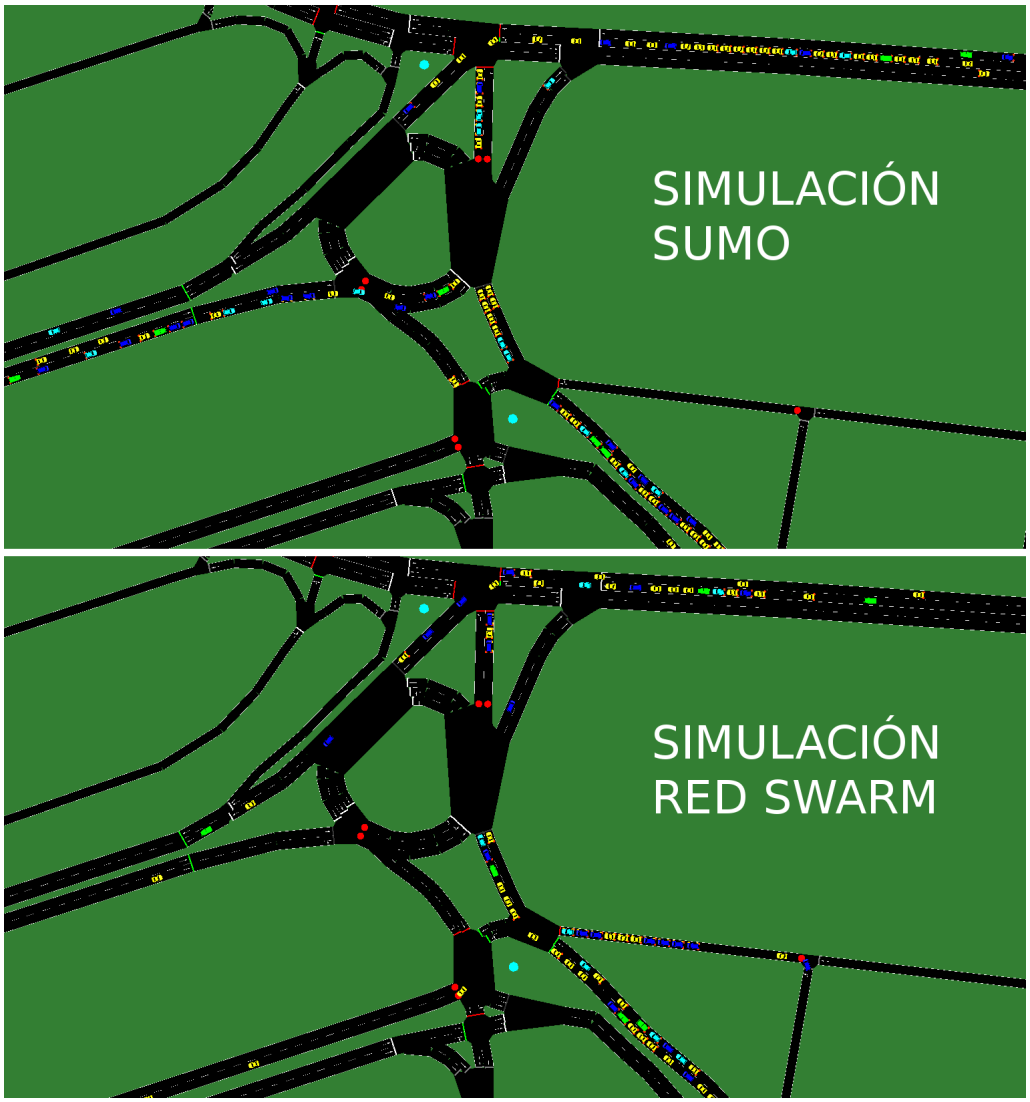


Figura 22: Captura del entorno gráfico del simulador *SUMO*, para el instante $t = 475s$, en dónde se comparan el estado del tráfico entre la **Simulación SUMO** y la **Simulación Red Swarm**. Obsérvese el menor número de vehículos en espera en los semáforos en el escenario de Simulación Red Swarm

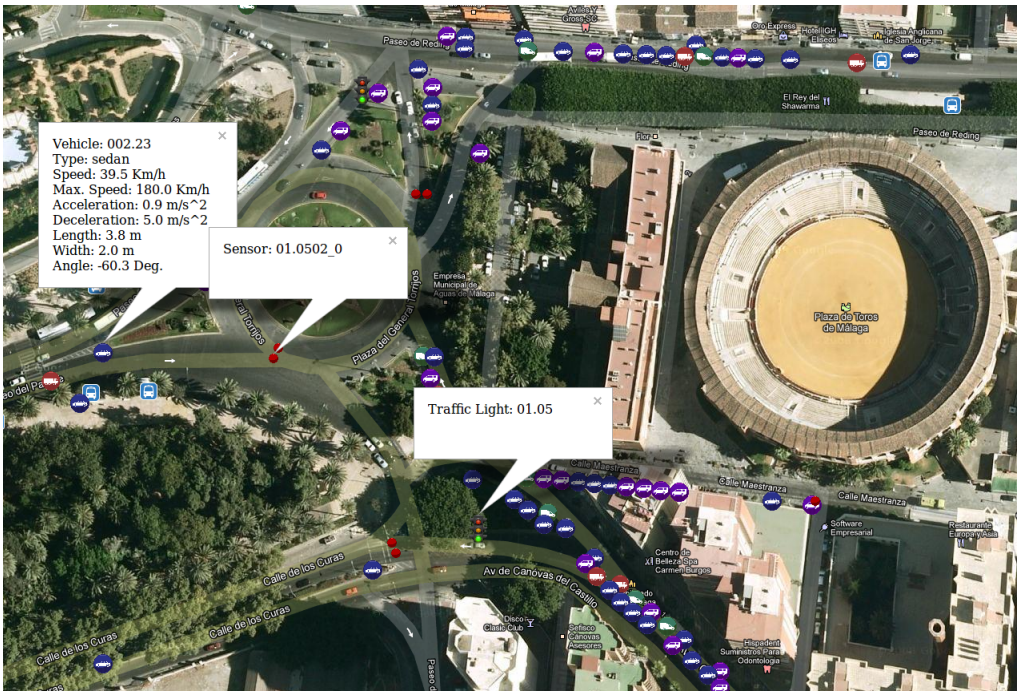


Figura 23: Exportación de una instantánea de la simulación a *Google Maps* en la que se visualiza el instante $t = 475s$ correspondiente a la **Simulación Red Swarm**

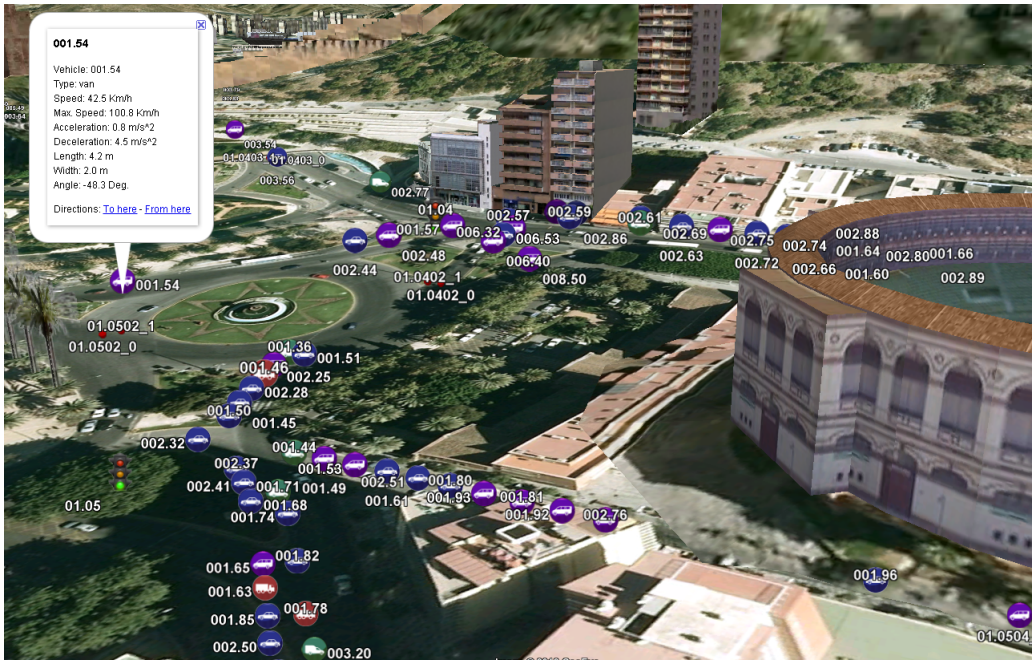


Figura 24: Exportación de una instantánea de la simulación a *Google Earth* en la que se visualiza el instante $t = 475s$ correspondiente a la **Simulación Red Swarm**