UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MIARFID Official Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging

TRABAJO FINAL DE MÁSTER 2016-2017

COORDINACIÓN INTELIGENTE EN VEHÍCULOS AUTÓNOMOS

AUTOR: Ángel Miguel Ferrando Ábalos

DIRECTORES:

Vicente Julián Inglada

Carlos Carrascosa Casamayor

Universidad Politécnica de Valencia

CONTENIDO

1- INTRODUCCIÓN

1.1. MOTIVACIÓN

Desde la creación del primer automóvil de la historia en 1769 hasta el día de hoy, el automóvil se ha convertido en el medio de transporte más utilizado del mundo. Más de 1.200.000 vehículos circulan por las extensas redes de carreteras construidas que comunican prácticamente todas las poblaciones mundiales.

Para mantener un orden y evitar los accidentes, se ha creado una normativa que todos los conductores deben cumplir para conseguir un tráfico ordenado y sobre todo seguro. Los coches han evolucionado en eficiencia, seguridad y comodidad, pero en los últimos años se empieza a hacer realidad algo que siempre ha sido característico de la ciencia ficción, los coches autónomos.

Debido a que los vehículos ya no serán controlados por personas, tanto la normativa como las infraestructuras deben adaptarse a este nuevo paradigma de la conducción para así mantener el orden en las carreteras. Es en el punto de las infraestructuras en el que se centra este trabajo, concretamente en mejorar la gestión de la preferencia de paso en los cruces de carreteras, haciendo que la decisión de paso o parada sea tomada por la Inteligencia Artificial. Utilizando la optimización de paso en los cruces, no sólo se vería mejorado el tráfico, sino que también afectaría positivamente a otros factores como:

- Una reducción en la emisión de gases de gases de efecto invernadero, y por lo tanto, una menor contaminación.
- Un ahorro de combustible para los usuarios de los vehículos, suponiendo además un gran ahorro para las empresas de transporte por carretera.
- La reducción de accidentes de tráfico.

Se han realizado diversos trabajos previos en la gestión de intersecciones y sería de gran utilidad crear un sistema que permita realizar comparaciones entre distintos métodos de gestión para poder decidir cuál es el más indicado para cada situación. Con este sistema además, se reducirían los costes de desarrollo de nuevos métodos ya que podrían simularse y optimizarse antes de tratar de construir una estructura real con la que podían no obtenerse los resultados esperados.

1.2. OBJETIVOS

En este trabajo se pretende implementar un simulador que permita probar y comparar distintas técnicas de gestión del paso de vehículos por una intersección. El objetivo principal se divide en los siguientes sub-objetivos:

1- Implementar un simulador que permita visualizar en tiempo real el paso de vehículos a través de una intersección.
2- Implementar un sistema de gestión de cruce utilizando semáforos.
3- Implementar un sistema de gestión de cruce que utilice inteligencia artificial. Para la consecución de este objetivo, se desarrollará una técnica de gestión presentada en el trabajo "*Cooperative driving: an ant colony system for autonomous intersection*

management" por los autores JiaWu, Abdeljalil Abbas-Turki y Abdellah El Moudni en el año 2011.

4- Añadir controles al simulador que permita modificar los parámetros del sistema con facilidad.
5- Implementar un sistema que genere gráficas a partir de los resultados del simulador. Esto permitirá comparar los valores obtenidos con las distintas técnicas de gestión del cruce.
6- Ampliar el simulador para que permita repetir experimentos previos a partir del registro de datos correspondiente.
7- Ampliación del simulador para que permita realizar simulaciones complejas con más de un cruce.

Para la implementación de los distintos componentes, se combinarán las siguientes tecnologías y metodologías:

- Programación dinámica: Algoritmo de colonia de hormigas para la programación del algoritmo de gestión del tráfico.
- Python: Lenguaje de programación sobre el que se desarrollará el sistema.
- Intelligent-Driver Model (IDM): Modelo de simulación de tráfico
- Matplotlib: Librería de generación y visualización de gráficas para Python.
- Pygame: Librería multimedia para Python

1.3. ESQUEMA DE LA OBRA

Para alcanzar los objetivos presentados, se va a seguir el siguiente esquema:

Inicialmente, en el capítulo 2, se hará un recorrido a lo largo de la historia del automóvil y su evolución tecnológica. Se describirán los diversos problemas que existen en la actualidad relacionados con el tráfico.

En el capítulo 3, se comentarán dos trabajos previos que se han realizado y que han servido de base para la elaboración de este trabajo. Se presentará la idea básica que pretendían transmitir, los modelos de gestión de tráfico diseñados y los resultados obtenidos.

A continuación, en el capítulo 4, se presentará el diseño e implementación del simulador de tráfico. Se detallarán todos los componentes que lo forman así como todas sus características.

El capítulo 5, se centrará en los dos modelos de gestión de tráfico implementados para probar el simulador. Se hará una breve descripción del método por semáforos ya que su funcionamiento es muy básico. En el caso del método por inteligencia artificial, se explicará con detalle el modelo de tráfico utilizado y su implementación.

En el capítulo 6, se mostrarán los resultados obtenidos con el simulador usando los dos métodos de gestión de tráfico implementados. Se utilizarán distintas gráficas y se compararán para comprobar que resultados se obtienen si se modifican los parámetros del sistema y que método de gestión es el más óptimo.

Para finalizar, en el capítulo 7, se presentarán las conclusiones finales del trabajo y se plantearán distintas mejoras que podrían ser implementadas en trabajos futuros.

2- ESTADO DEL ARTE

2.1. INTRODUCCIÓN HISTÓRICA

Fue Nicolas-Joseph Cugnot (1725-1804), un escritor e inventor francés, el creador del primer automóvil de la historia. Comenzó a circular con él por las calles de París en 1769. El vehículo consistía en el montaje de un motor de vapor sobre la rueda delantera de un triciclo, haciendo de ésta la rueda motriz y tractora del invento.

Figura 1- Primer automóvil de vapor. Versión de 1771

Es en 1886 cuando Karl Friedrich Benz crea el primer automóvil por motor de combustión interna, usando gasolina como combustible, en la ciudad de Mannheim (Alemania). El modelo fue denominado Benz Patent-Motorwagen y es uno de los vehículos base que comenzaron a dar forma a los coches actuales.

Figura 2- Réplica del Benz Patent-Motorwagen

La historia del automóvil se divide en distintas etapas:

2.1.1. ETAPA VETERANA

En 1889 comienza la producción masiva de automóviles en Francia y Estados Unidos. Las primeras compañías en hacerlo fueron las francesas *Panhard Et Levassor* (1889) y *Peugeot* (1891). En Estados Unidos, Henry Ford comenzó a construir automóviles en una cadena de montaje en 1908. Este sistema permitió alcanzar tasas de producción hasta entonces impensables.

2.1.2. ETAPA DE LATÓN O EDUARDIANA

Así nombrada por el uso del latón en las carrocerías. Comprende desde el final de la Primera Guerra Mundial hasta la Gran Depresión de 1929.

2.1.3. ETAPA PRE-GUERRA

Abarca desde 1929 a 1949. Es en esta época cuando se desarrollan los primeros coches completamente cerrados y de forma más redondeada.

2.1.4. ETAPA MODERNA

Esta etapa se caracteriza por el desarrollo de motores más seguros, eficientes y menos contaminantes. En la actualidad nos encontramos en esta etapa.

2.2. INTEGRACIÓN DE LA INFORMÁTICA EN EL AUTOMÓVIL

Durante la etapa moderna, la informática ha avanzado y evolucionado a grandes pasos de forma paralela a los automóviles. Estos avances se han aplicado a los nuevos vehículos para mejorar sus sistemas de conducción y el confort de los conductores.

Actualmente la mayoría de vehículos modernos fabricados en los últimos 15 años, integran un ordenador de a bordo que se encarga de monitorizar el estado del vehículo y sus distintos sensores. Dependiendo del vehículo, estos sensores pueden monitorizar más o menos funciones de la conducción, permitiendo al conductor visualizar toda la información en una pantalla. Algunas de las funciones monitorizadas pueden ser:

- Velocidad del vehículo
- Estado de los neumáticos
- Temperatura ambiente
- Temperatura del motor
- Estado de los frenos
- Estado de las luces
- Niveles de líquidos
- Detección de colisiones
- Detección de salida del carril

Toda esta información obtenida no sólo es útil para informar al conductor del correcto funcionamiento de los sistemas en todo momento, si no que en los últimos años se ha comenzado a utilizar para automatizar acciones del vehículo sin la intervención del usuario.

Algunas de las primeras acciones automatizadas implementadas fueron la activación automática de luces y limpiaparabrisas. Las siguientes acciones que se añadieron fueron el aparcamiento automático y la asistencia de frenado en caso de detección de colisión.

Todas estas automatizaciones comenzaron siendo características Premium de vehículos de alta gama, pero actualmente comienzan a ser funciones básicas que se incluyen en todos los modelos, no solo por la comodidad añadida al automóvil, sino también por el aumento en la seguridad que suponen. Debido a la alta tasa de mortalidad en accidentes de tráfico registrada todos los años (habiéndose registrado 1.160 muertes sólo en el último año en España), la seguridad es uno de los factores más importantes en la fabricación de cada vehículo.

El último avance en la automatización de vehículos es la conducción autónoma. Actualmente empresas como Ford, Google o Audi están realizando pruebas de los primeros vehículos autónomos. Esto supone un gran cambio en el paradigma de la conducción, ya que será el propio vehículo el que tome decisiones a partir del análisis de la información recibida de todos los factores exteriores a través de sus sensores.

2.3. EL VEHÍCULO AUTÓNOMO

El primer vehículo autónomo de la historia que se conoce fue presentado por Norman Bel Geddes en la Exposición Universal de 1939 de Nueva York. Consistía en un vehículo eléctrico que era controlado por un circuito eléctrico integrado en el pavimento de la carretera.

En 1980, Emst Dickmanns y su equipo de la Universidad de Múnich, diseñaron una furgoneta guiada mediante visión por computador. Esta furgoneta consiguió alcanzar los 100 Km/h por carreteras sin tráfico. La Comisión Europea se interesó por el proyecto y realizó una inversión de 800 millones de Euros para el proyecto *EUREKA Prometheus*, cuyo objetivo era desarrollar un vehículo autónomo.

El mismo año, *Defense Advanced Research Projects Agency* (DARPA) consiguió desarrollar el primer vehículo que funcionaba mediante un radar láser y visión computarizada.

Durante 1994 y 1995 los vehículos autónomos siguieron evolucionando consiguiendo circular por carriles libres y en convoy. También se automatizaron los cambios de carril a izquierda y derecha y los adelantamientos. Se realizaron diversas pruebas de conducción con tráfico real con éxito en las que las intervenciones humanas fueron mínimas.

En el verano de 2015, la Universidad de Michigan puso en funcionamiento un pueblo de utilería denominado MCity, en cuyas calles se pueden probar vehículos autónomos. El lugar cuenta con una vía de más de un kilómetro y medio de largo, curvas de diferentes radios, rotondas, semáforos, pavimentos de diferentes superficies, etc... Por otro lado, existen otros proyectos que complementarán la MCity. Uno de ellos es la puesta en marcha de 9.000 vehículos interconectados en la gran superficie de la ciudad de Ann Arbor, así como otros 20.000 en carreteras al sureste del estado.

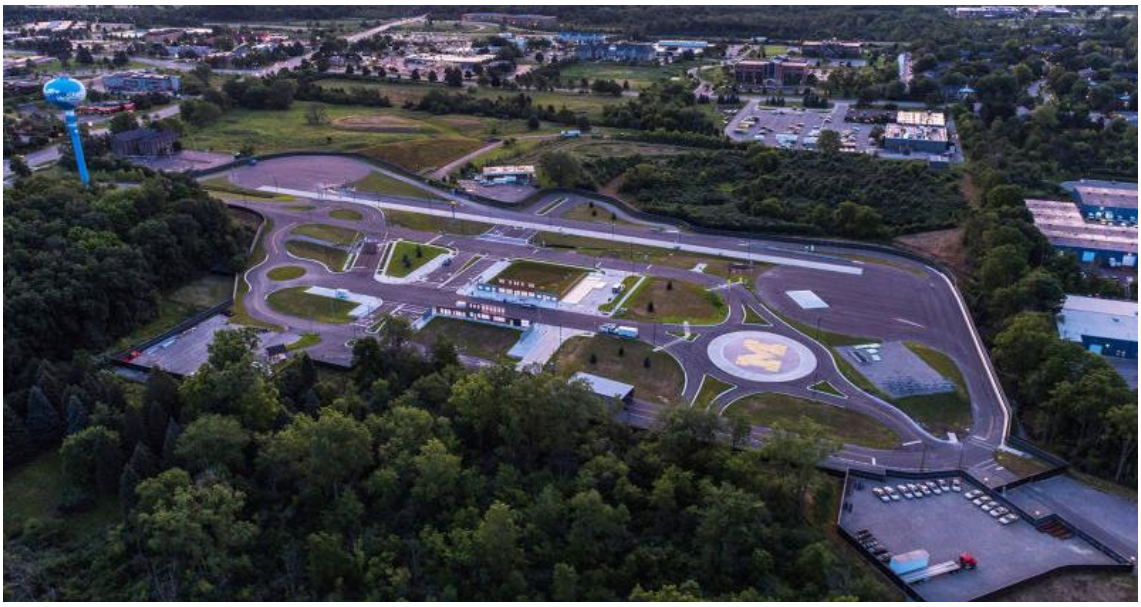

Figura 3- Vista aérea de la ciudad de pruebas MCity(Michigan)

2.3.1. LA CONDUCCIÓN AUTÓNOMA

Para habilitar la conducción autónoma en un vehículo, se requieren los siguientes elementos:

- Sensores:
  - LIDAR: Miden distancias por láser en 360 grados de visión.
  - Cámaras: Identifican objetos.
  - Radar: Para percibir lluvia, niebla, nieve…
- Algoritmos: Determinan la localización, selección de rutas…
- Visión por computador y entrenamiento de modelos para la toma de decisiones.
- Mapas 3D de alta definición.
- Alta potencia de cálculo para una rápida toma de decisiones.

Igual que los seres humanos conducimos utilizando la vista y tomamos decisiones a partir de los estímulos visuales, el vehículo autónomo utiliza los datos que recibe de los sensores para tomar decisiones en su ordenador central.

Un vehículo autónomo conoce en todo momento su situación y lo que va a encontrar durante su trayecto gracias a los mapas 3D de alta resolución. Los datos de los mapas son comparados con los recibidos a través de los sensores LIDAR, de esta manera se asegura de que su trayecto es correcto y que no hay obstáculos u otro tipo de peligros inesperados.

Figura 4- Prototipo del Ford Mondeo autónomo

Toda la información que percibe el coche autónomo y la que tiene almacenada, es procesada por el ordenador central de alta potencia de cálculo. Para detectar qué está ocurriendo alrededor del vehículo, se utilizan técnicas de aprendizaje automático. Los sistemas se entrenan con imágenes de todo tipo de situaciones posibles para identificar cual es la que está ocurriendo en cada momento. De la misma forma se entrenan los algoritmos de toma de decisiones. Para distintas situaciones se proporciona la acción correcta a ejecutar. En función de si se acierta o no en la toma de decisiones, el algoritmo se irá modificando para mejorar su índice de acierto.

Otra de las técnicas que se usan en la toma de decisiones son los modelos probabilísticos. El coche necesita predecir su propia posición y la de todos los objetos que identifica, como peatones u otros vehículos. Dado que los objetos pueden estar en movimiento, se intenta predecir su posición utilizando distribuciones de probabilidad.

2.3.2. PROBLEMAS DE LA CONDUCCIÓN AUTÓNOMA

Hoy en día el principal obstáculo para la implementación de los vehículos autónomos no se deriva de las limitaciones de la tecnología, sino de factores políticos, jurídicos, de regulación, de infraestructura y de responsabilidad que deben abordarse.

Cuando el conductor del vehículo ya no es un ser humano, en caso de accidente no existe un procedimiento para determinar quién ha sido el responsable del mismo. Esto aún debe ser especificado en las leyes y realizarse la adaptación de los seguros a esta nueva normativa.

De la misma manera que la ley debe adaptarse, también la infraestructura de tráfico debe adaptarse al nuevo paradigma de conducción. Además de ofrecer un tráfico más seguro, puede ofrecer un tráfico mejorado, más fluido, que permita evitar atascos y accidentes utilizando

técnicas de inteligencia artificial que no tomen decisiones sobre los vehículos individualmente, sino en grupo, haciendo que varios vehículos involucrados en un mismo evento, se coordinen entre ellos tomando decisiones globales.

Un ejemplo de evento en el que interviene más de un vehículo, es el cruce de una intersección. Actualmente se gestiona utilizando semáforos u otro tipo de señalización. Estos mecanismos suponen la interrupción del tráfico en una de las carreteras que se cruzan mientras permite el paso a la otra. En muchos de estos casos, mientras la circulación está abierta en una línea, no circula ningún vehículo por ella o mientras una línea está cerrada durante un tiempo, se acumula el tráfico innecesariamente. En otros casos en los que un semáforo no interviene, el factor humano puede ser el que provoque los atascos e incluso accidentes por imprudencias.

En este trabajo, se presentará una mejora respecto al sistema actual adaptada a los vehículos autónomos. El nuevo sistema coordinará el cruce de los vehículos por las intersecciones tratando de calcular el mejor orden de paso minimizando el tiempo de espera de los vehículos. Esto permitirá evitar tráfico innecesario y evitar accidentes entre los vehículos.

3- TRABAJOS PREVIOS

Ya se han realizado algunos trabajos anteriormente relacionados con la optimización del tiempo de paso en el cruce de intersecciones. A continuación se resumen las dos publicaciones más relevantes sobre este tema:

3.1- DISCRETE INTERSECTION SIGNAL CONTROL

En el año 2007, Jia Wu, Abdeljalil Abbas-Turki, Aurelien Corréïa y Abdellah El Moudni, presentaron el estudio "Discrete intersection signal control". En este trabajo propusieron un nuevo sistema de control de semáforos en una intersección simple entre dos carriles. El sistema tiene en cuenta la información obtenida de los vehículos para ajustar los cambios de estado de los semáforos.

Se considera la intersección como un recurso compartido entre todos los vehículos, mientras que las señales del semáforo tienen en cuenta la llegada de cada uno de los vehículos individualmente.

Por una parte esto permite distinguir diferentes tipos de vehículos como pueden ser los vehículos públicos o de emergencias. Por otra parte, aunque el objetivo principal del trabajo es minimizar el tiempo de evacuación de todos los vehículos del cruce, también el tiempo de espera de los vehículos en los semáforos se ve drásticamente reducido.

Aunque este trabajo no estaba enfocado a vehículos autónomos, la técnica de cálculo aplicada para obtener la secuencia de semáforos óptima, se puede aplicar a este tipo de vehículos que no necesitarían semáforos para cruzar de forma óptima la intersección.

El siguiente modelo de intersección fue diseñado para implementar el sistema:

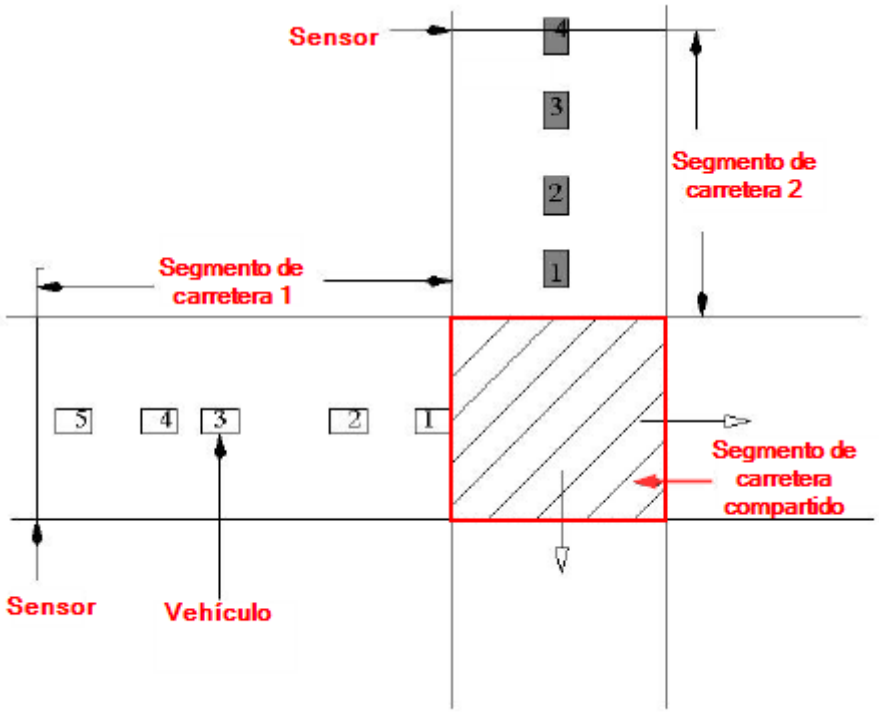

Figura 5- Modelo del estudio "Discrete intersection signal control"

En este modelo se observan 2 carreteras con una única dirección. Los vehículos que se aproximan al segmento compartido de las carreteras (el cruce), son detectados por el sistema una vez pasan por el sensor de cada una de las vías. Cuando se detectan, entran a formar parte del listado de vehículos para el que se realizará el cálculo de la secuencia óptima de semáforos. Este cálculo estima el tiempo de llegada de cada vehículo al cruce y utilizando estos tiempos, calcula la secuencia que minimizará el tiempo de evacuación de todos los vehículos del sistema. Se define el tiempo de evacuación como el tiempo que tardan en cruzar todos los vehículos desde que llega el primer vehículo al cruce hasta que termina de cruzar el último.

Para realizar el cálculo, se obtienen todas las secuencias posibles en las que pueden cruzar los vehículos y se selecciona aquella cuyo tiempo de evacuación es el menor. Un ejemplo de secuencia para el modelo de la figura 4 es el siguiente:

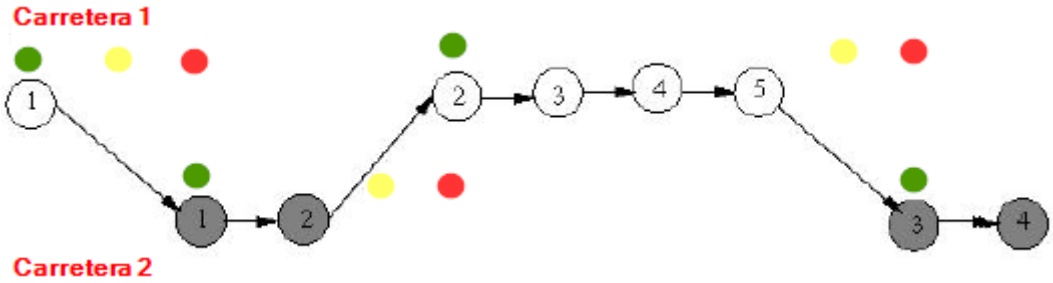

Figura 6- Secuencia resultado del modelo de la figura 4

Los círculos coloreados representan la secuencia de cada semáforo para permitir el paso de los vehículos según la secuencia de cruce calculada previamente.

Mientras el semáforo se encuentra amarillo, ninguno de los vehículos tiene permitido cruzar. El tiempo que permanece el semáforo es este estado, es el suficiente para evacuar el cruce antes de que se ponga verde el semáforo en el otro carril.

Las conclusiones de este trabajo fueron que utilizando la información del tiempo de llegada de los vehículos, se podía ajustar la secuencia de semáforos óptima para evacuar todos los vehículos del cruce en un tiempo mucho menor que utilizando el método tradicional con secuencias de tiempo fijas. El inconveniente era que solo funcionaba con 2 carriles de una única dirección.

3.2- A BRANCH AND BOUND ALGORITHM FOR NEW TRAFFIC SIGNAL CONTROL SYSTEM OF AN ISOLATED INTERSECTION

Este trabajo fue presentado en el año 2009 por Fei Van, Mahjoub Dridi y Abdellah El Moudni. El objetivo del mismo era minimizar el tiempo en el que cruzan todos los vehículos por un cruce utilizando un método de control capaz de comunicarse con los vehículos que se aproximan al cruce mediante señales inalámbricas.

En el sistema presentado se considera que todos los vehículos son autónomos y no se utilizan semáforos, el sistema manda señales de paso a cada vehículo una vez calculada la secuencia óptima de paso y estos reaccionan sin intervención humana. La comunicación entre el sistema de control y los vehículos es bidireccional, y son los propios vehículos los que avisan al sistema

de su aproximación e informan de su estado. Esto significa que el sistema obtiene la velocidad y posición precisa de cada vehículo, además de cualquier otra información que pueda ser de utilidad.

Básicamente, el sistema está compuesto por un controlador en el centro del cruce que recibe las señales de los vehículos, y por algún medio de comunicación inalámbrica instalado en cada vehículo que permita enviar las señales como una PDA.

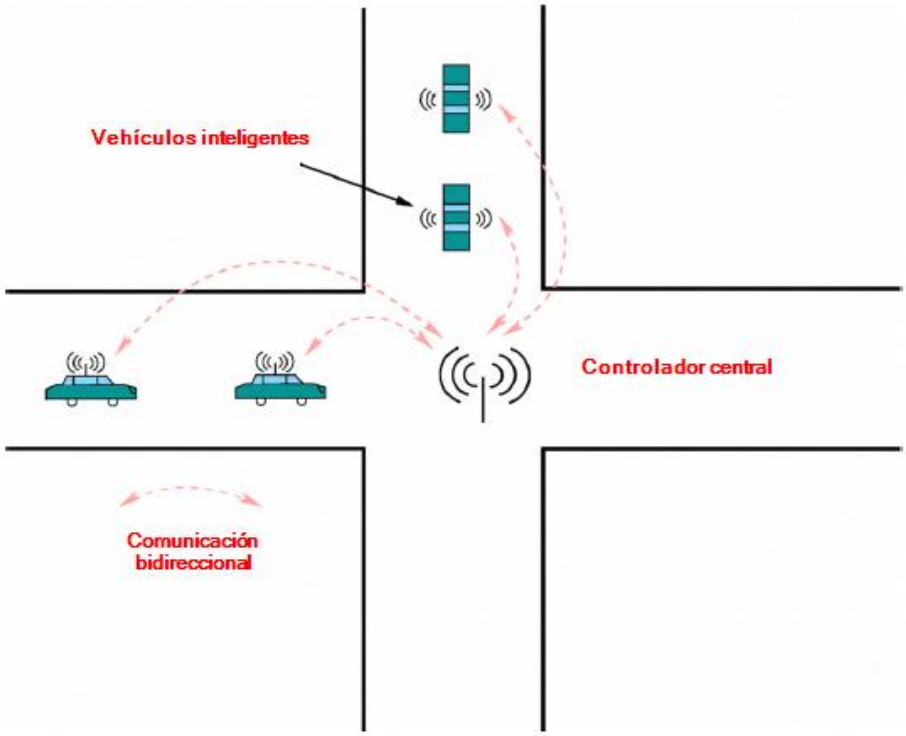

Figura 7- Componentes del sistema inalámbrico

El problema principal del sistema presentado, es obtener una secuencia de paso óptima de los vehículos, que minimice el tiempo de cruce de todos los vehículos que han mandado sus señales al controlador central.

Para resolver el problema, consideran al controlador como una única máquina capaz de gestionar procesos paralelamente y a cada vehículo como un trabajo.

De cada vehículo, se obtiene la siguiente información:

- Tiempo de llegada a la intersección. Considerado el instante de ejecución del trabajo.
- Tiempo de aceleración requerido por el vehículo desde que pasa de un estado de parada a una distancia segura del vehículo que tiene detrás. Modelado como el tiempo de proceso de trabajo.
- Intención del vehículo (seguir recto, girar a la derecha, girar a la izquierda…). Se considera como la clase del trabajo.

El sistema define que solo una clase de trabajos puede ser procesada al mismo tiempo (trabajos paralelos). Lo que se traduce como que solo atravesarán el cruce simultáneamente aquellos vehículos que tengan la misma intención.

En la siguiente imagen se muestra un ejemplo de vehículos que han llegado al cruce y cómo el controlador los ha dividido en grupos dependiendo de su intención.

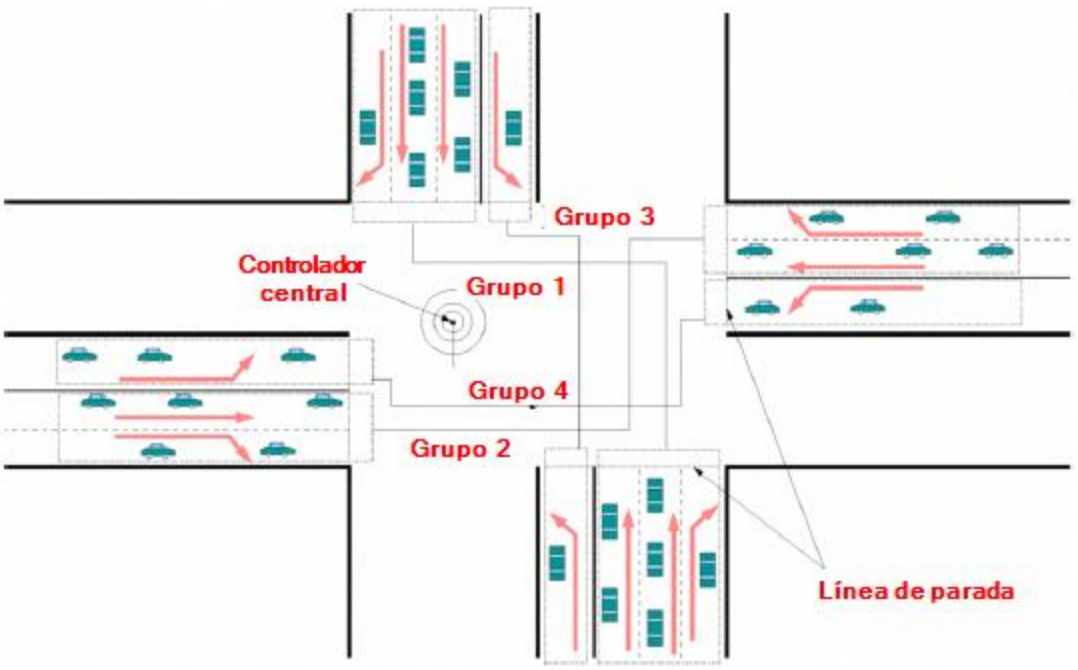

Figura 8- División por grupos de trabajos

Para obtener la solución óptima, el sistema calcula el instante y tiempo de ejecución de los posibles grupos a partir de los trabajos que los componen. Una vez definidos los grupos, se calcula su cadena de ejecución que minimiza el tiempo de ejecución total de todos los trabajos.

Dado que puede existir un número muy alto de combinaciones de trabajos y grupos a procesar, se requiere utilizar una técnica de cálculo óptima que permita calcular las secuencias en tiempo real. En el trabajo presentado se utilizó un algoritmo de optimización combinatoria *Branch & bound.*

Tras una serie de pruebas realizadas sobre el sistema, se concluyó que era posible gestionar el paso de vehículos por intersecciones aisladas utilizando el sistema presentado, optimizando el tiempo de cruce de los vehículos utilizando un algoritmo *Branch & bound*.

4- SIMULADOR DE TRÁFICO EN CRUCES

Para poder comparar distintos métodos de gestión del tráfico en los cruces, en este trabajo se ha implementado un simulador que permite visualizar las distintas técnicas de gestión en tiempo real. El simulador crea un cruce entre cuatro carriles a partir de una serie de parámetros y genera tráfico aleatoriamente por cada uno de ellos. Un controlador virtual monitoriza el tráfico y gestiona el cruce de los vehículos por la intersección, permitiendo al usuario ver el flujo de vehículos. Durante la simulación, todos los eventos son registrados permitiendo generar gráficas que representan el comportamiento de los métodos de control utilizados.

La implementación se ha realizado utilizando el lenguaje de programación Python debido a las siguientes razones:

- Existe un amplio número de librerías para realizar todo tipo de acciones, como pueden ser la programación gráfica o la generación de gráficas.
- Al ser un lenguaje interpretado, no requiere compilar cada vez que se realiza un cambio y agiliza el desarrollo.
- Los programas se ejecutan mediante el intérprete Python, lo que permite ejecutarlo en distintos sistemas sin modificar el código.

4.1- DISEÑO

El simulador está formado por los siguientes componentes:

- El motor gráfico: Se encarga de dibujar los elementos en pantalla.
- El generador de tráfico: Genera vehículos en circulación aleatoriamente a partir de parámetros pre-configurados.
- El simulador de tráfico: Calcula las posiciones y estados de los vehículos a lo largo del tiempo de simulación.
- El registro de eventos: Registra los eventos que se generan durante la simulación en un fichero de texto.
- El generador de gráficas: Genera gráficas a partir de los eventos registrados.
- El sistema GIA (Gestor de intersección autónoma): Controla las órdenes de paso del cruce.

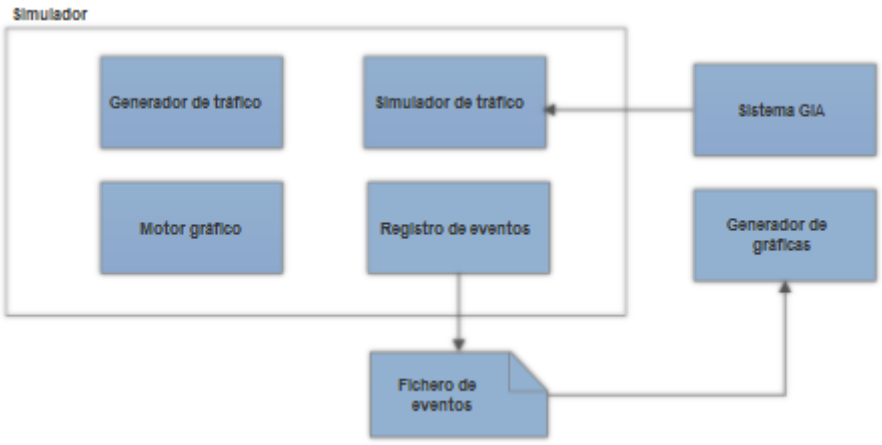

Figura 9- Esquema de componentes del simulador

El siguiente diagrama de flujo representa el funcionamiento de la función principal del simulador. Esta es la encargada de inicializar los distintos componentes y controlar el bucle principal de la simulación:

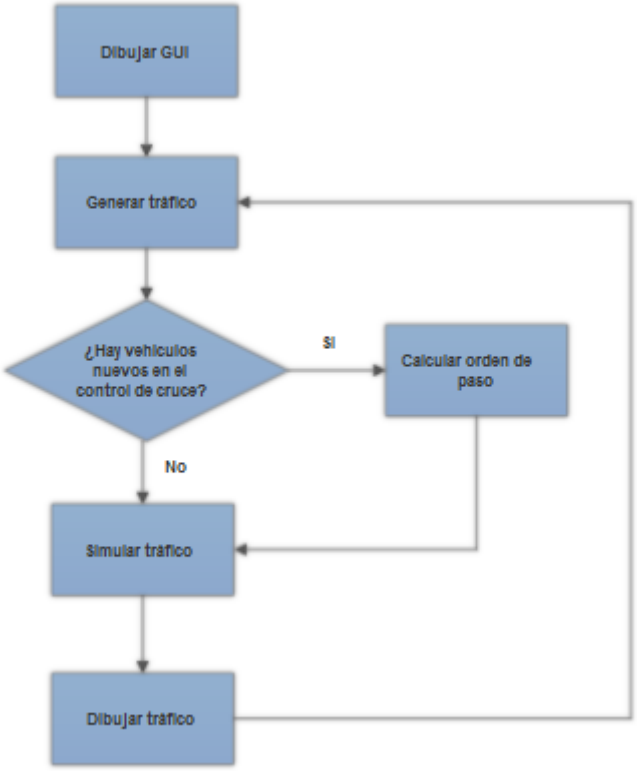

Figura 10- Diagrama de flujo de la función principal del simulador

En los puntos siguientes se explicará el funcionamiento de cada uno de los componentes del sistema con más detalle.

4.2- EL GENERADOR DE TRÁFICO

Como su nombre indica, este componente se encarga de generar nuevos vehículos al principio de cada iteración en el bucle principal. Los vehículos pueden ser inicializados de forma aleatoria en cualquiera de los cuatro carriles del cruce dependiendo de los siguientes parámetros:

- T: (T > 0) Tiempo mínimo entre la llegada de vehículos al cruce
- α: ($\alpha \in [0, 1]$) Probabilidad de llegada de vehículo

En cada iteración del bucle principal se hace una comprobación del tiempo transcurrido desde la entrada del último vehículo al cruce. Si el tiempo transcurrido es mayor que T, se genera un número aleatorio entre 0 y 1. Si este número es mayor que α, se inyectará un nuevo vehículo en el cruce. El carril en el que se insertará el vehículo, se seleccionará aleatoriamente usando la misma probabilidad de un 25% en cada uno de los 4 carriles.

4.3- EL SIMULADOR DE TRÁFICO

Para simular el comportamiento de los vehículos, se ha utilizado el *Intelligent-Driver Model (IDM)*.

4.3.1- DEFICIÓN

El IDM es un modelo de control de tráfico microscópico en el que cada vehículo es un componente independiente de la simulación. El modelo permite conocer el estado del tráfico en cada momento determinando las posiciones y las velocidades de todos los vehículos simulados.

Concretamente, el IDM es un modelo de seguimiento. La decisión de cada vehículo de acelerar o frenar depende únicamente de su propia velocidad y de la posición y velocidad del vehículo que circula delante de él. El modelo también permite simular las decisiones de cambio de carril de los vehículos, pero dado que en este trabajo no se realizan, no se detallará esta funcionalidad.

El modelo se puede describir como sigue:

- Los factores que influyen en cada vehículo (entrada del modelo) son:
  - Su propia velocidad *v*
  - La distancia *s* entre el propio vehículo y el que tiene delante
  - La diferencia de velocidades Δv entre el propio vehículo y el que tiene delante (Positiva cuando se realiza un acercamiento)
- La salida del modelo es la aceleración *dv/dt* elegida para el vehículo en cada instante
- Otros parámetros del modelo son el estilo de conducción de los vehículos, como por ejemplo si el conductor conduce rápido o lento, agresivo o cuidadoso…

Las ecuaciones que definen el modelo son las siguientes:

$$\frac{dv}{dt} = a\left[1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*(v,\Delta v)}{s}\right)^2\right]$$

Donde

$$s^*(v,\Delta v) = s_0 + \max\left[0,\ \left(vT + \frac{v\Delta v}{2\sqrt{ab}}\right)\right]$$

Figura 11- Ecuaciones del modelo IDM

La aceleración de un vehículo se divide para conseguir su aceleración “deseada” entre $[1-(v/v_0)^{delta}]$, y la deceleración de frenado es inducida por el vehículo que tiene delante. La aceleración desciende desde la aceleración inicial a 0 cuando se ha aproximado a la velocidad “deseada” v0.

El término de frenado está basado en la comparación entre la distancia deseada entre vehículos s* y la distancia actual *s*. Si la distancia actual es aproximadamente igual a s*, entonces la deceleración de frenado compensa la aceleración del vehículo para que sea cercana a 0. Esto significa que la distancia s* corresponde a la distancia entre vehículos cuando el tráfico está siendo fluido. Adicionalmente, s* crece dinámicamente cuando el vehículo está circulando lento

y decrece si el vehículo de delante es más rápido. Como consecuencia, la deceleración impuesta crece en los siguientes casos:

- Cuando la distancia entre el vehículo actual y el que tiene delante decrece (Los vehículos tienen que mantener una distancia de seguridad)
- Cuando crece su propia velocidad (La distancia de seguridad decrece)
- Cuando crece la diferencia de velocidad entre los vehículos (Si se aproxima demasiado al vehículo de delante, puede ocurrir una situación de peligro)

4.3.2- PARÁMETROS

Los parámetros para cada vehículo del IDM son los siguientes:

- *v0:* Velocidad de circulación deseada
- *T:* Tiempo de seguridad deseado entre 2 vehículos
- *a:* Aceleración común en circulación
- *b:* Deceleración de frenado común en circulación
- *s0:* Mínima distancia de seguridad entre vehículos
- delta: Exponente de aceleración

Los parámetros estándar usados para cada vehículo son los siguientes:

Parámetro	Valor	Notas
v0	120 Km/h	Velocidad en carretera. Se debe adaptar para ciudades u otro tipo de vías. Por ejemplo 50 en ciudad o 90 en carreteras comarcales españolas…
T	1,5 s	Es el tiempo recomendado por las autoescuelas alemanas. Los valores reales oscilan entre 2 y 0,8 llegando a ser inferiores.
s0	2 m	Distancia de seguridad incluso cuando los vehículos están parados.
A	0,3 m/s^2	Valores muy bajos generan tráfico no deseado. Los valores reales oscilan entre 1 y 2 m/s^2
B	0,3 m/s^2	Valores muy altos generan tráfico no deseado. Los valores reales oscilan entre 1 y 2 m/s^2

Tabla 1- Parámetros del IDM

4.3.3- MODELO BALÍSTICO

El simulador integra el IDM resolviendo sus ecuaciones diferenciales en cada instante de tiempo de forma aproximada. Para ello se define un intervalo de tiempo de actualización Δt y se añade al modelo asumiendo aceleraciones constantes. A este modelo se le denomina modelo balístico del IDM y se define como sigue para cada vehículo:

Nueva velocidad	$v(t+\Delta t)=v(t) + (dv/dt)\,\Delta t$
Nueva posición:	$x(t+\Delta t)=x(t)+v(t)\Delta t+1/2\,(dv/dt)\,(\Delta t)^2$
Nueva distancia entre vehículos	$s(t+\Delta t) =x_l(t+\Delta t)- x(t+\Delta t)- L_l$

Tabla 2 - Modelo balístico IDM

Donde

- *dv/dt*: Es la aceleración IDM calculada en el instante t
- *x*: Es la posición del vehículo delantero
- L_1: Es la logitud del vehículo delantero

El modelo se considera suficientemente bien definido para todos los casos en los que haya un vehículo delante y no exista ningún objeto que impida la circulación. Sin embargo existen las siguientes generalizaciones:

- Si no existe vehículo delantero ni ningún otro objeto obstruyendo el camino, se debe ajustar la distancia entre vehículos a un valor muy alto como 1000m.
- Si se pretende simular una obstrucción en el trafico, se debe utilizar un vehículo virtual con velocidad, aceleración y longitud 0 en la posición del bloqueo.
- Si se quiere aplicar un límite de velocidad en tramos concretos, se debe ajustar la velocidad deseada a un valor inferior.

Para vehículos aproximándose a otro vehículo parado, el método de actualización balístico puede resultar en velocidades negativas. Si la distancia entre un vehículo y otro parado es menor a la distancia mínima de seguridad, el cálculo de la aceleración IDM puede devolver un resultado negativo haciendo que la velocidad sea negativa en el siguiente intervalo. En un caso real, el vehículo sigue circulando aunque la distancia sea menor y decelera, pero no circula marcha atrás. Este comportamiento se añade al modelo de la siguiente forma:

Si $v(t) + (dv/dt)\,\Delta t < 0$, entonces:

Nueva velocidad	$v(t+\Delta t)=0$
Nueva posición:	$x(t+\Delta t)=x(t) - 1/2\, v^2(t) / (dv/dt)$
Nueva distancia entre vehículos	$s(t+\Delta t) = x_l(t+\Delta t) - x(t+\Delta t) - L_l$

Tabla 3- Modelo balístico IDM adaptado a situaciones de parada

En el simulador implementado, cuando un vehículo llega a la intersección del cruce y no tiene asignada la orden de paso, se inserta un vehículo virtual delante de él hasta que se le asigna la orden. En la interfaz gráfica del simulador, se pueden visualizar los vehículos virtuales representados por una señal de STOP dentro de la intersección en el cuadrante que pertenece al carril del vehículo.

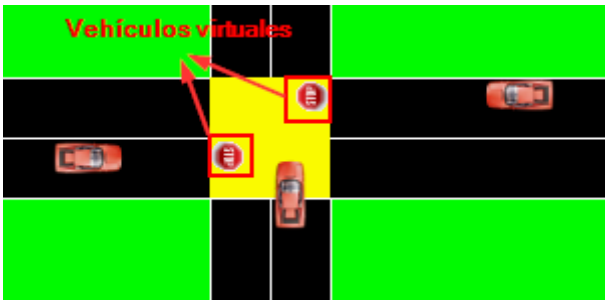

Figura 12- Representación de vehículos virtuales

4.4- EL SISTEMA GIA

El sistema GIA es el componente que determina el orden por el que deben cruzar los vehículos. El componente es independiente del resto del simulador para facilitar la adición de nuevos métodos de cálculo sin tener que modificar el resto de componentes.

Para añadir un sistema GIA, se debe crear una nueva clase con los siguientes métodos:

- initialize(): Método de inicialización del sistema
- run(): Método de ejecución del sistema. Éste método devolverá la secuencia de paso.

Esta nueva clase se añadirá al simulador en el método *calculate_order* de la case *GuiCrossControl*.

Para que el sistema GIA sea compatible con el simulador, debe tener unos parámetros de entrada y salida determinados:

4.4.1- PARÁMETROS DE ENTRADA

Como parámetro de entrada debe aceptar un objeto de tipo vo.InnerData.

Este objeto tiene 4 atributos:

- d_value: Tiempo mínimo de seguridad entre coches del mismo carril
- s_value: Tiempo mínimo de seguridad entre coches de carriles perpendiculares
- car_list: Lista de vehículos presentes en el cruce. Cada vehículo contiene su identificador, carril al que pertenece y tiempo estimado de llegada al punto de intersección.
- lines_list: Lista de carriles presentes en el cruce. Cada carril contiene su identificador y una lista con los carriles perpendiculares al mismo.

4.4.2- PARÁMETROS DE SALIDA

La salida del sistema GIA debe ser una lista de objetos de tipo vo.Car. Esta lista contendrá los vehículos del objeto de entrada ordenados según la secuencia de paso óptima calcula, siendo el vehículo en el índice 0 el primero en pasar.

4.5 EL MOTOR GRÁFICO

El motor gráfico es el componente encargado de dibujar todos los elementos del simulador en pantalla.

La implementación del motor gráfico se ha realizado utilizando la librería *Pygame* cuyas características principales son:

- Es gratuita y de código abierto
- Está destinada a la implementación de aplicaciones multimedia como los videojuegos
- Está implementada sobre la librería SDL (Simple DirectMedia Layer). Esta librería de C fue diseñada para acceder a los dispositivos de audio, teclado, ratón y gráficos utilizando OpenGL y Direct3D.
- Es multiplataforma

4.5.1 DISEÑO

El motor gráfico se ha diseñado de forma que consuma el menor tiempo posible de dibujado en cada iteración del simulador. El dibujado de componentes se realiza en 2 fases:

1- Durante la primera fase, se dibuja la interfaz gráfica y toda la infraestructura del cruce (carretera, marcas de control, etc.) Esta fase se ejecuta únicamente durante la inicialización del simulador y todos los objetos son estáticos.
2- La segunda fase consiste en el dibujado de los vehículos y las marcas de parada. En cada iteración del simulador se ejecuta el redibujado después de calcular las nuevas posiciones de cada vehículo.

La velocidad de simulación está limitada a 30 frames por segundo, pudiendo ser modificada en las propiedades del sistema. Se ha seleccionado este frame rate porque es suficiente para visualizar una simulación fluida permitiendo tener un intervalo de tiempo mayor entre cada iteración. Si el valor de este intervalo fuera demasiado cercano a 0, el modelo balístico del simulador de tráfico devolvería siempre el mismo valor en cada iteración, haciendo que los vehículos no avancen.

4.6 EL REGISTRADOR DE EVENTOS

El registrador de eventos se encarga de monitorizar todos los eventos que ocurren durante la simulación para poder generar gráficas del comportamiento de cada sistema GIA. También permiten generar el tráfico de la simulación a partir de los datos registrados en vez de usar el generador aleatorio de tráfico. Los siguientes eventos son registrados:

- Entrada de un vehículo en el sistema
- Salida de un vehículo del cruce
- Cambio en los parámetros del sistema GIA

Todos los eventos registrados son almacenados en un fichero de texto con los siguientes formatos:

- Cuando se inicia el simulador, todos los parámetros son almacenados con su valor inicial siguiendo el siguiente patrón:

  INI:PARÁMETRO:VALOR

  Por ejemplo:
  INI;ANT_NUMBER:5
  INI;ITERATIONS_NUMBER:5
  INI;Q_VALUE:0.1

- Si se modifica un parámetro durante la simulación, el registro utiliza el siguiente formato:

  CHANGE;PARÁMETRO:NUEVO_VALOR

  Por ejemplo:

CHANGE;ITERATIONS_NUMBER:13
CHANGE;ITERATIONS_NUMBER:7

- Cada vez que un vehículo ha cruzado la zona de conflicto, se registra con el siguiente formato:

  ID:TIEMPO_DE_CRUCE:INSTANTE_DE_ ENTRADA:DIRECCION

  Siendo el tiempo de cruce el tiempo que transcurre desde que el vehículo entra al cruce hasta que sale; el instante de entrada, el instante de la simulación en el que el vehículo llega al control; y la dirección, el carril por el que entra. Los valores de tiempo valores se registran en ms. Ejemplo:

  11:1809:8908:S
  0:11349:1752:E
  2:10139:2962:W

4.7 EL GENERADOR DE GRÁFICAS

El generador de gráficas es una pequeña aplicación independiente al simulador que genera gráficas a partir del fichero del registro de eventos. Las gráficas generadas permiten comparar el rendimiento de los sistemas GIA así como el impacto producido cuando se cambia algún parámetro del sistema.

La implementación del componente utiliza la librería Matplotlib para generar las gráficas. Matplotlib es una librería de generación de gráficas en 2D para Python que permite generar distintas figuras para representar los datos. Es multiplataforma y permite al usuario tener un control total sobre los estilos de las figuras dibujadas haciendo uso de funciones muy similares a las que usa MATLAB para generar gráficas.

El generador, tiene 3 opciones de generación:

1- Cambio de parámetros: Muestra el tiempo medio de espera de todos los vehículos procesados en intervalos marcados por algún cambio en los parámetros del sistema.
2- Grupo de vehículos: Muestra el tiempo de espera medio de cada secuencia de X vehículos procesados por el sistema. El número de vehículos X puede ser modificado por el usuario.
3- Intervalo de tiempo: Muestra el tiempo de espera medio de los vehículos en cada intervalo de tiempo X. El intervalo de tiempo puede ser modificado por el usuario.

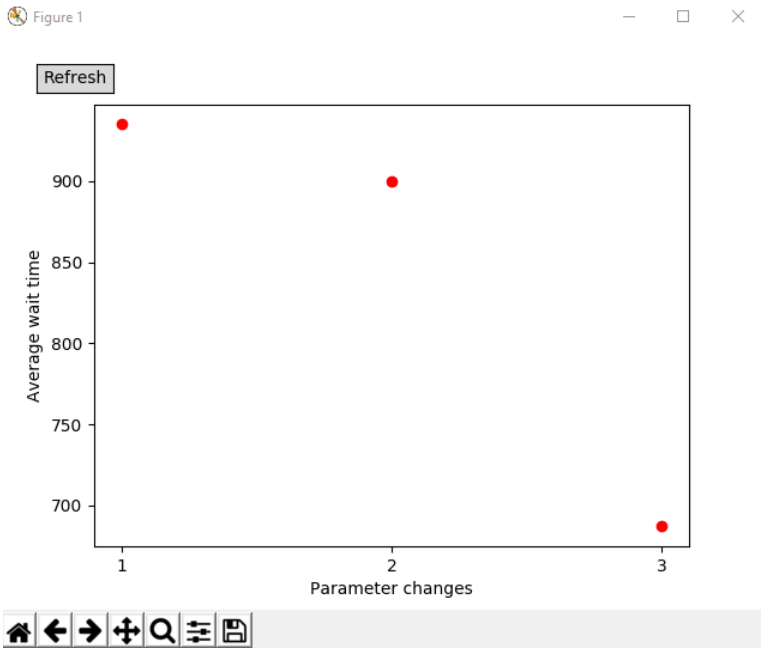

Figura 13- Ejemplo de gráfica generado para la opción "Cambio de parámetros"

5- GESTIÓN DE INTERSECCIÓN AUTÓNOMA

La gestión de intersección autónoma (GIA), presenta un nuevo sistema para dirigir a los vehículos en una intersección. En este sistema, la decisión de cruzar o parar no la toma el vehículo, si no un controlador de intersección que conoce todos los vehículos que van a cruzar en un determinado radio de acción.

La finalidad principal del sistema es determinar el orden de cruce de los vehículos que suponga el menor tiempo desde que llega el primer vehículo hasta que cruza el último de los vehículos detectados por el controlador.

Para probar el simulador, se han implementado 2 sistemas GIA. El primer sistema, simula el uso de semáforos convencionales, mientras que el segundo utiliza técnicas de inteligencia artificial para tratar de mejorar la eficiencia del sistema tradicional. Ambos sistemas están diseñados para funcionar con vehículos autónomos, no tienen en cuenta las posibles intervenciones de un ser humano sobre el control del vehículo.

5.1- DISEÑO Y COMPONENTES

Para la implementación del sistema GIA se definen los siguientes conceptos que serán requeridos para obtener la secuencia de cruce óptima de los vehículos:

- Controlador: Es el actor principal del sistema. Se encargará de recopilar los datos de los vehículos y determinar la secuencia óptima.
- Radio de acción: Determina la distancia a partir de la cual un vehículo comienza a ser monitoreado por el controlador.
- Zona de conflicto: Es la superficie del cruce en la que los carriles se superponen.
- Zona de almacenamiento: Es el tramo de carretera que comienza en el límite del radio de acción y termina en la entrada a la zona de conflicto.
- Zona de salida: Se considera zona de salida al lado opuesto a la zona de almacenamiento una vez termina la zona de conflicto.

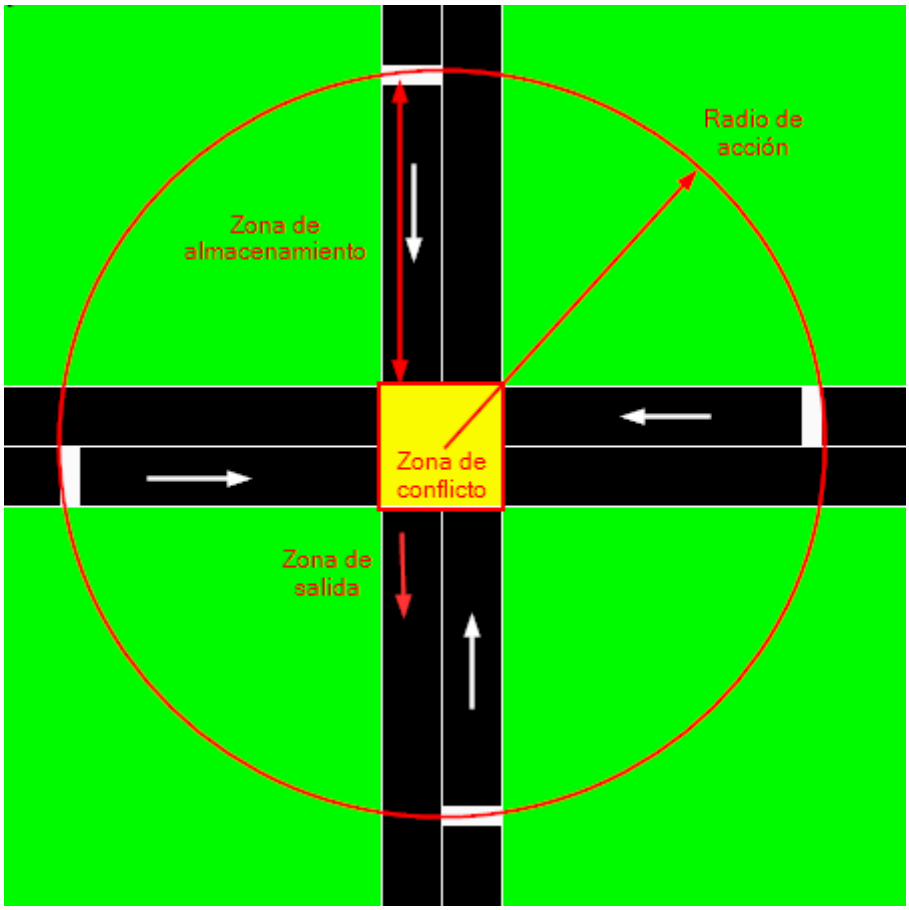

Figura 14- GIA, esquema de conceptos

- Movimiento conflictivo: Un movimiento conflictivo, es toda aquella maniobra en la que se ven implicados dos vehículos de dos carriles que se cruzan. Por ejemplo, un vehículo esté cruzando la zona de conflicto en dirección oeste, mientras otro vehículo llega por el norte.

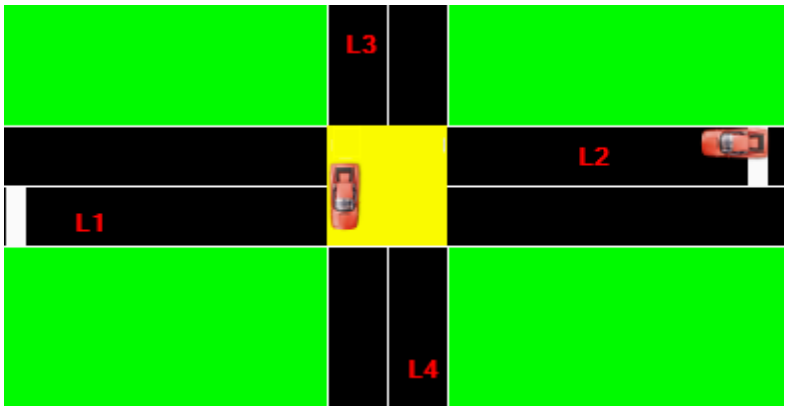

Figura 15- Movimiento conflictivo

- Movimiento seguro: Un movimiento seguro, es toda aquella maniobra en la que se ven implicados 2 o más vehículos del mismo carril. Por ejemplo, dos vehículos cruzan la zona de conflicto en dirección Oeste.

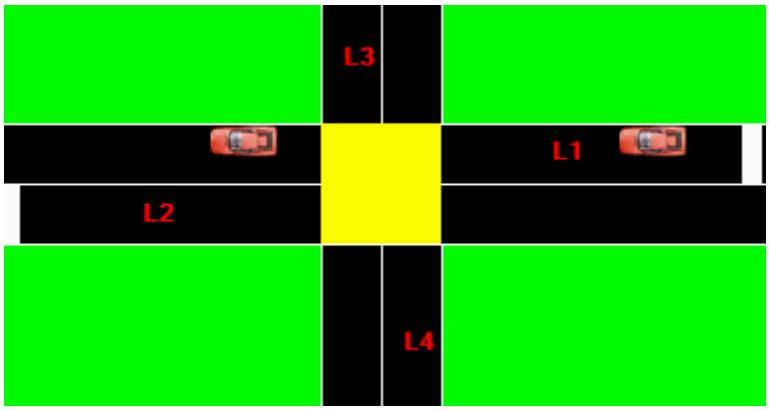

Figura 16- Movimiento seguro

Para la detección de los vehículos, se requiere de una serie de sensores que determinen la posición y estado de cada uno. Estos sensores pueden ser cámaras, sensores de presión integrados en el pavimento, sensores láser… Por cada carril que interviene en el cruce, se deben incluir los siguientes:

- Sensor de entrada a la zona de almacenamiento.
- Sensor de entrada a la zona de conflicto.
- Sensor de entrada a la zona de salida.

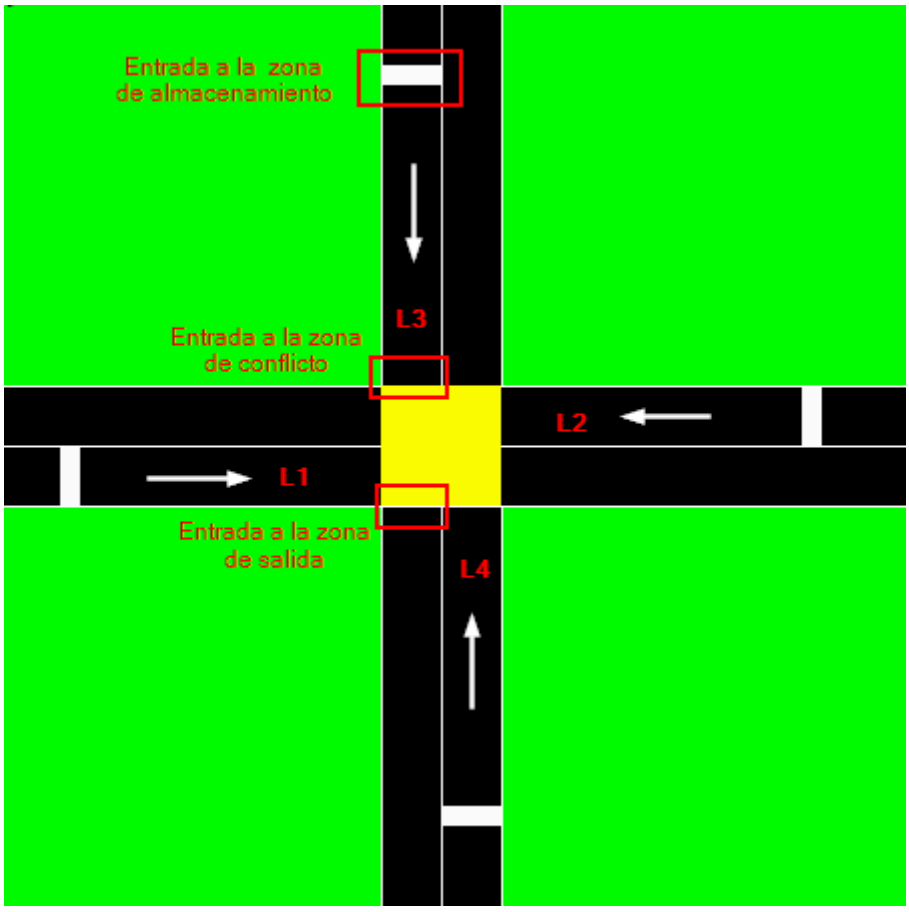

Figura 17- GIA Esquema de sensores

Por último, debido a que los vehículos deben estar en constante comunicación con el controlador, se requiere un canal de comunicación. Es te canal puede ser implementado con cualquier tecnología inalámbrica como el Bluetooth.

5.2- GESTIÓN CON SEMÁFOROS

El sistema GIA controlado por semáforos es muy sencillo. El controlador simula a los semáforos tradicionales. En este sistema no se tiene en cuenta la zona de almacenamiento ya que la información obtenida de los vehículos no afecta al cálculo de las órdenes de paso.

Al inicio de la simulación, se determina un intervalo de tiempo T y cada intervalo de tiempo T, se permite el paso por carriles que no generen conflictos entre ellos. Mientras estos carriles permanecen abiertos, el resto de carriles permanecerán cerrados.

Cada vez que finaliza un intervalo T, se cierran todos los carriles durante un tiempo *d* para permitir la evacuación de la zona de conflicto en el caso de que aún haya algún vehículo cruzando. Así evitamos que mientras un vehículo esté cruzando, se produzcan movimientos conflictivos causando posibles accidentes.

El valor del intervalo T, puede ser modificado por el usuario en cualquier momento de la simulación.

5.3- GESTIÓN CON INTELIGENCIA ARTIFICIAL

En este punto se describe el proceso que sigue el sistema para gestionar el tráfico en el cruce utilizando técnicas de inteligencia artificial. Se presenta un método originalmente creado por JiaWu, Abdeljalil Abbas-Turki y Abdellah El Moudni que fue publicado en el paper *Cooperative driving: an ant colony system for autonomous intersection management* el 12 de Octubre del 2011. Éste método es la evolución de los métodos descritos en el apartado de trabajos previos de este documento.

5.3.1 DISEÑO

Inicialmente el controlador se encuentra en espera de la llegada de vehículos manteniendo una cola virtual vacía. En el momento que un vehículo atraviesa la entrada a la zona de almacenamiento, el controlador registra el vehículo en el sistema y calcula el orden que deben seguir los vehículos incluidos en la cola para cruzar. Mientras se realiza el cálculo, ningún vehículo de la cola tendrá la orden de cruce.

Cuando los vehículos se aproximan a la entrada de la zona de conflicto, comienzan a frenar si el controlador no les ha dado aún la orden de paso. Si han recibido la orden de paso, el vehículo cruzará la zona de conflicto y este será eliminado de la cola virtual.

Una vez que el vehículo termina la maniobra de cruce, el vehículo atraviesa la entrada de la zona de salida notificando al controlador que ya ha cruzado. El controlador deja de monitorear el vehículo

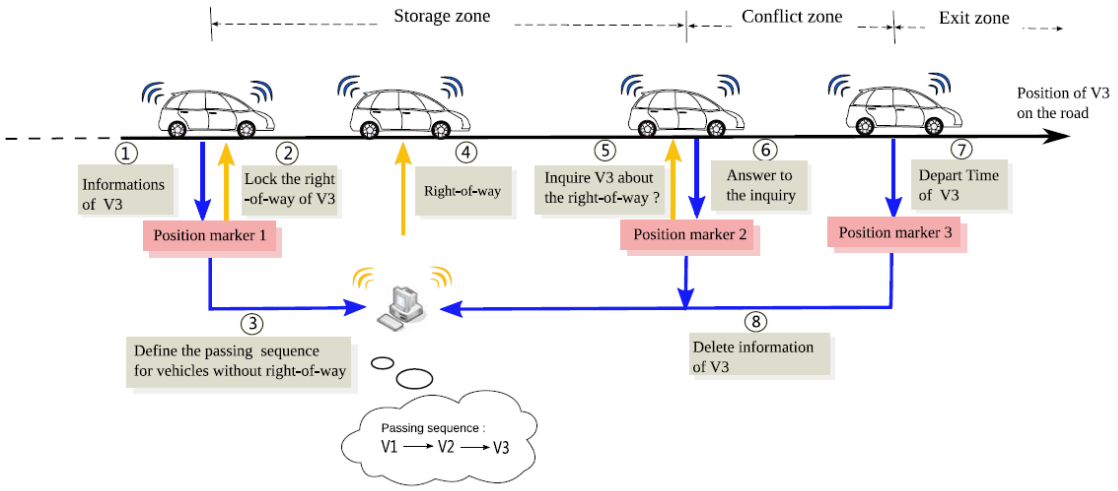

Figura 18- Representación gráfica de la estrategia de control

Por seguridad, el sistema implementa las siguientes restricciones:

- Mientras un vehículo cruza la zona de conflicto, ningún otro vehículo de las carreteras perpendiculares puede recibir una orden de paso.
- Durante el cálculo de la secuencia de paso, ningún vehículo incluido en la cola puede entrar en la zona de conflicto.
- Ya que los vehículos siguen avanzando mientras se realiza el cálculo de la secuencia de paso, por defecto irán frenando gradualmente mientras no tengan la orden de paso. Si llegan a la entrada de la zona de conflicto sin haber recibido la orden, quedarán parados. Así se pretende evitar que los vehículos frenen bruscamente.

5.3.2- ESTRATEGIA

El principal problema que debe resolver el sistema GIA, es el cálculo de la secuencia de paso óptima. Para realizar el cálculo, el sistema sólo necesita la estimación del instante en el que cada vehículo de la cola entrará en la zona de conflicto. El instante de acceso se puede aproximar a partir de los siguientes parámetros:

- *r:* Es el tiempo mínimo que tarda un vehículo en atravesar la zona de almacenamiento.
- *d:* Es el tiempo mínimo de seguridad que debe transcurrir desde que un vehículo entra en la zona de conflicto hasta que entra el siguiente por el mismo carril (Movimiento seguro).
- *s:* El tiempo de seguridad que debe transcurrir desde que un vehículo entra en la zona de conflicto hasta que entra el siguiente por uno de los carriles perpendiculares (Movimiento conflictivo).

Estos parámetros dependen de diversos factores como las condiciones del pavimento, el perfil de los conductores y el clima. Dado que la finalidad de este trabajo no es calcular estos valores, se utilizarán unos valores estándar previamente obtenidos mediante observaciones empíricas [1].

Mediante la lista de vehículos del controlador y la aproximación del tiempo de llegada a la zona de conflicto de cada coche, el sistema GIA tratará de obtener la secuencia de paso de vehículos

que minimice el máximo tiempo de salida, es decir, el tiempo que tarda en salir el último vehículo de la zona de conflicto.

5.3.3- FORMULACIÓN

La formulación del sistema GIA es la siguiente:

Dado $r(i,1,...,r(i,n_i), i\epsilon[1,L]$

Siendo $r(i,q_1) < r(i,q_i+1), q_i\epsilon[1,n_i-1]$

Encontrar $min\left\{\underset{i\epsilon[1,L]}{max}\{e(i,n_i)\}\right\}$

Restricciones

$$|e(i,q_i)-e(j,q_j)| \geq \begin{cases} \mathrm{d},\ i=j \wedge q_i = q_j+1, \\ \mathrm{s},\ i \overset{\wedge}{\neq} j, \end{cases}$$

$$e(i,q_i) \geq r(i,q_i)$$

Donde $\overset{\wedge}{\neq}$ indica dos movimientos conflictivos desde los carriles Li y Lj

Definición de la notación:

Notación	Significado
L	Número total de carriles
n_i	Número de vehículos presentes en la línea Li
(i,q_i)	Vehículo q_i en la línea Li
$r(i,q_i)$	El tiempo de llegada teórico del vehículo (i,q_i)
$e(i,q_i)$	Tiempo de llegada del vehículo (i,q_i) a la zona de conflicto
D	Tiempo de seguridad para movimientos no conflictivos
S	Tiempo de seguridad para movimientos conflictivos

Tabla 4- Notación en la formulación del sistema GIA

Para explicar el proceso de control de vehículos, se utilizará un ejemplo en el que 9 vehículos se encuentran presentes en un una intersección de cuatro carriles como la mostrada en la figura 7.

La siguiente tabla muestra los vehículos dentro de la zona de control de cada carril y el tiempo estimado en el que estarían listos para atravesar la zona de conflicto. Estos tiempos están calculados a partir de las observaciones obtenidas en el punto de entrada a la zona de control combinadas con los parámetros *d* y *s*.

Carril 1 (L1)	Carril 2 (L2)	Carril 3 (L3)	Carril 4 (L4)
r(1,1) = 0	r(2,1) = 1	r(3,1) = 4	r(4,1) = 6

r(1,2) = 3	r(2,3) = 5	r(3,2) = 7
r(1,3) = 8	r(2,3) = 10	

Tabla 5- Vehículos de ejemplo

Siendo los parámetros *d = 2* y *s = 6*, la siguiente figura muestra una secuencia óptima de paso para los vehículos de la figura 10. El resultado obtenido es un tiempo de 17 segundos hasta que el último vehículo cruza la zona de conflicto.

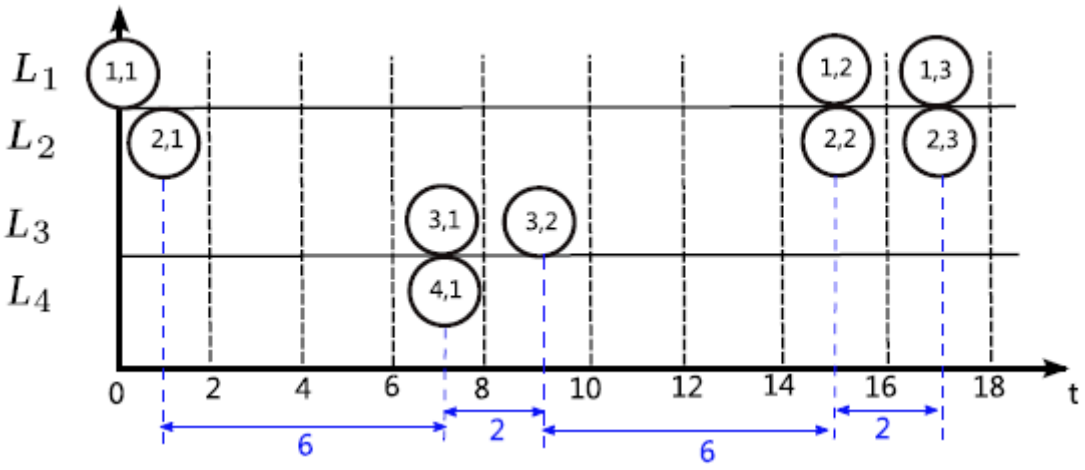

Figura 19- Representación gráfica de secuencia óptima para el ejemplo de la figura 10

Uno de los problemas del sistema GIA es la complejidad es su complejidad ya que el controlador considera cada vehículo individualmente. La siguiente fórmula muestra cuantas posibles secuencias pueden obtenerse para el problema.

$$\frac{\left(\sum_{i=1}^{L} n_i\right)!}{\prod_{i=1}^{L}(n_i!)}$$

Figura 20- Fórmula del número de posibles soluciones para un sistema GIA

Se observa que la complejidad del problema crece con el número de carriles y el de vehículos que intervienen en el cálculo, haciendo imposible encontrar la solución óptima mediante una búsqueda sobre todas las posibles soluciones en tiempo real.

Debido a la restricción del tiempo de cálculo, se puede hacer uso de técnicas meta heurísticas, las cuales tienen la reputación de resolver problemas complejos en un tiempo reducido. Estas técnicas ya han sido aplicadas con éxito a otro tipo de problemas de optimización como la búsqueda de rutas en mapas o el ajuste de calendarios.

5.3.4- ALGORITMO DE COLONIA DE HORMIGAS

El algoritmo de la colonia de hormigas (Ant Colony Optimization, ACO) es una técnica probabilística para solucionar problemas computacionales que pueden reducirse a buscar los mejores caminos o rutas en grafos.

Las hormigas deambulan de forma aleatoria para buscar comida. Una vez encontrada comida regresan a su colonia dejando un rastro de feromonas. Si otras hormigas encuentran dicho rastro, es probable que estas sigan el rastro de feromonas, regresando y reforzándolo si estas encuentran comida finalmente.

Sin embargo, con el paso del tiempo el rastro de feromonas comienza a evaporarse, reduciéndose su fuerza de atracción. Cuanto más tiempo le tome a una hormiga viajar por el camino y regresar, más tiempo tienen las feromonas para evaporarse. Un camino corto, es usado de forma más frecuente, y por lo tanto su densidad de feromonas es mayor que en los caminos largos.

La evaporación de feromonas también tiene la ventaja de evitar convergencias a óptimos locales**.** Si no hubiese evaporación, los caminos elegidos por la primera hormiga tenderían a ser excesivamente atractivos para las siguientes hormigas y el espacio de búsqueda de soluciones sería limitado.

Por tanto, cuando una hormiga encuentra un buen camino entre la colonia y la fuente de comida, hay más posibilidades de que otras hormigas sigan este camino y con una retroalimentación positiva se conduce finalmente a todas las hormigas a un solo camino.

La idea del algoritmo colonia de hormigas es imitar este comportamiento con "hormigas simuladas" caminando a través de un grafo que representa el problema en cuestión.

5.3.5- REPRESENTACIÓN GRÁFICA DEL PROBLEMA

Para poder aplicar el algoritmo de colonia de hormigas, es necesario representar el problema como un grafo. El grafo G = (V, A, w) es acíclico y dirigido y se compone de los siguientes elementos:

- (V) n + 2 nodos: Representan los vehículos involucrados en el cálculo y dos nodos adicionales que representan el inicio y final de la secuencia.
- (A) arcos: Indican la relación entre dos vehículos.
- (w) pesos: Corresponden a los pesos de las aristas (tiempo de seguridad)

Un nodo (i, q_i) representa la q_i llegada de un vehículo del carril L_i, donde $1 \leq q_i \leq n_i$, i $\in [1, L]$. Los pesos de los arcos (v_1, v_j) se definen con las siguientes premisas:

$$w(v_i, v_j) = \begin{cases} d, & \text{if } v_i = (i, q_i) \text{ and } v_j = (i, q_i + 1) \\ s, & \text{if } v_i = (i, q_i), v_j = (j, q_j) \text{ and } i \overset{\wedge}{=} j \\ 0 & \text{En cualquier otro caso} \end{cases}$$

Donde $\overset{\wedge}{\neq}$ indica dos movimientos conflictivos desde los carriles Li y Lj

La siguiente figura representa el grafo del problema utilizando como los ejemplo los datos de la tabla 5.

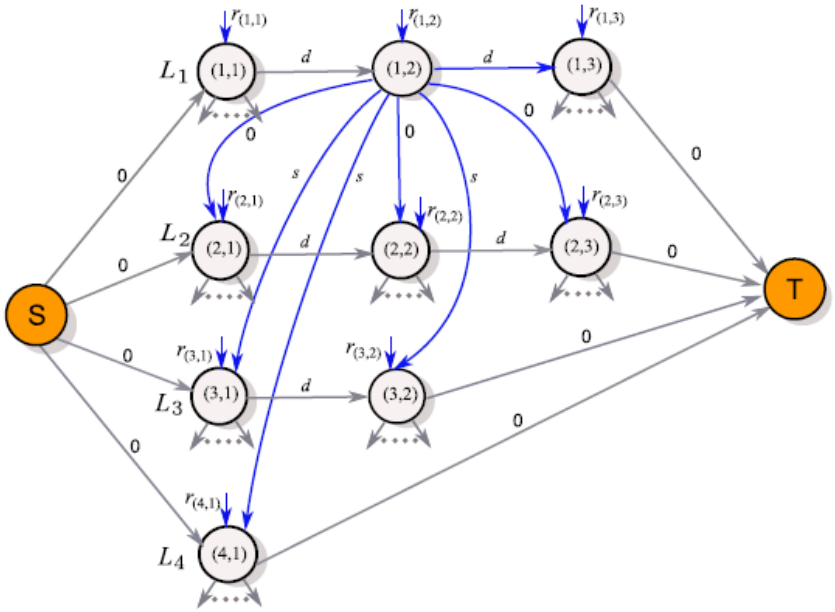

Figura 21- Representación gráfica del problema

En la figura sólo se muestran los arcos del vehículo (1, 2) para simplificar la imagen.

No pueden existir arcos desde un vehículo hasta un vehículo anterior en el mismo carril ya que el orden de los vehículos en un carril no puede variar. Por el mismo motivo, no podría existir una transición desde el vehículo (1, 1) al (1, 3).

Se observa que los arcos desde los vehículos (Li, q_i) a (Li, q_i + 1) tienen un peso *d* debido a que es el tiempo mínimo de seguridad para movimientos no conflictivos. En cambio, los arcos desde los carriles L1 y L2 a los carriles L3 y L4, tienen un peso *s* porque generan un movimiento conflictivo.

Dado que los vehículos en carriles paralelos pueden cruzar la zona de conflicto a la vez sin generar conflictos, el peso de los arcos desde L1 a L2 y desde L3 a L4 tienen peso 0.

Todos los movimientos descritos entre distintos carriles tienen el mismo peso si se realizan a la inversa, por ejemplo el arco de L3 a L4 tendría también un peso 0.

Las posibles soluciones al problema, son un *camino* P(S, T) desde el nodo inicial S al nodo final T que visite todos los nodos del grafo una única vez transitando por los arcos definidos.

Basándose en la definición del grafo G, la longitud de un camino parcial desde S a un nodo (i, q_i) (representada como $L_{(i,q_i)}$) indica el tiempo de acceso del vehículo (i, q_i) a una secuencia. Sin embargo $L_{(i,q_i)}$ debe ser mayor o igual que el tiempo teórico de llegada del vehículo (i, q_i) representado en el grafo como $r_{(i,q_i)}$. Con esta premisa, la longitud del camino parcial desde un nodo v_i a otro nodo v_j se calcula mediante la siguiente fórmula:

$$L(v_j) = max\left\{L(v_i) + w(v_i, v_j), r(v_j)\right\}$$

Figura 22- Cálculo de longitud de caminos parciales

Un camino $P^*(S,T)$ es una solución óptima para el problema, si para todos los caminos $P(S,T)$ tenemos que $L_T^* \leq L_T$.

5.3.6- IMPLEMENTACIÓN

El siguiente algoritmo es aplicado para obtener el camino más corto del grafo $P^*(S,T)$.

Inicializar valores de feromona basados en una heurística.
Inyectar *m* hormigas en el nodo *S*

for *i* = 1, …, *C* **do** (*C* número de iteraciones)
  for *j* = 1, …, *m* **do** (*m* número de hormigas)
    La hormiga *j* aplica las reglas de transición para **obtener una solución**
    Se aplica regla para **actualizar la feromona local**
  end for
  Se aplica regla para **actualizar la feromona global**
end for

Se deben especificar los siguientes criterios:

- Inicialización de la feromona
- Regla de transición entre nodos
- Regla de actualización de feromona

5.3.7- INICIALIZACIÓN DE FEROMONA

Para inicializar la feromona se utiliza la misma heurística aplicada para resolver el problema del viajero descrito en la referencia [1].

La feromona se calcula como $\tau_0 = (n * L_0)^{-1}$, donde *n* es el número total de nodos y L_0el máximo tiempo de salida de una secuencia obtenida por la heurística propuesta.

Esta heurística asigna la orden de paso a los vehículos que realizan movimientos no conflictivos cuyo tiempo de seguridad es igual al mínimo valor *d*. La siguiente figura muestra la secuencia obtenida por la heurística descrita para los datos de ejemplo de la figura 10.

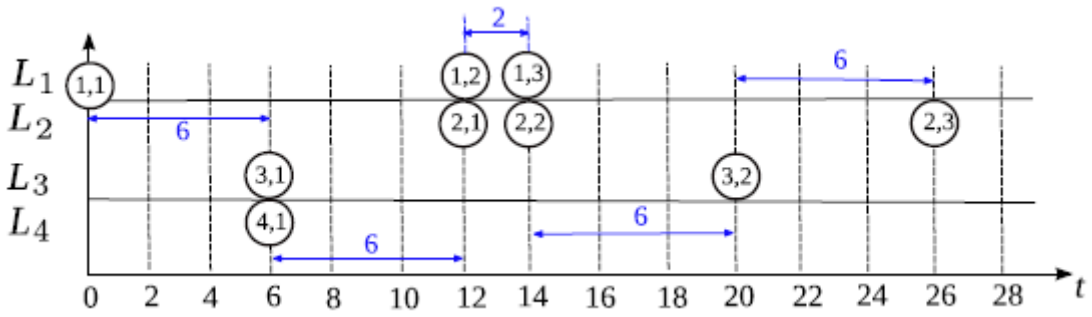

Figura 23- Secuencia resultado de la heurística de inicialización de la feromona

5.3.8- REGLA DE TRANSICIÓN DE ESTADOS

Siendo:

- a: La hormiga actual obteniendo una solución
- j: El nodo actual en el que se encuentra la hormiga
- k: El nodo destino de la transición
- (j,u): El arco desde el nodo j a un nodo candidato u
- $\tau(j,u)$: La feromona del arco (j,u)
- $n(j,u)$: La información heurística si la hormiga a transita de j a u. Representa el impacto en la longitud del camino parcial L_k tras visitar el nodo k. Se calcula como $(L_k - L_j + 1)^{-1}$ donde L_k y L_j se calculan como se indica en la figura 14. Se suma 1 al cálculo para evitar que el denominador sea 0.
- $V_a(j)$: El conjunto de nodos pendientes de visitar por la hormiga a

Y los parámetros:

- β: *($\beta > 0$)* es un parámetro que denota la importancia de la información heurística
- q_0: $(0 \leq q_0 \leq 1)$ Siendo q un número aleatorio entre 0 y 1, si es menor que q_0, entonces la hormiga a seleccionará el siguiente nodo que genere el máximo valor de $\tau(j,u) * n(j,u)^{\beta}$, a esto se le llama *explotación*. En el caso de que q sea mayor o igual que q_0, el siguiente nodo se seleccionará aleatoriamente usando la siguiente distribución de probabilidad, a esto se le denomina *exploración*:

$$p_a(j,k) = \begin{cases} \frac{\tau(j,k)\cdot\eta(j,k)^{\beta}}{\sum_{u\in V_a(j)}\tau(j,u)\cdot\eta(j,u)^{\beta}}, & \text{if } k \in V_a(j) \\ 0, & \text{otherwise} \end{cases}$$

Figura 24- Distribución de probabilidad cuando se aplica exploración

Se aplica la siguiente regla de transición:

$$k = \begin{cases} \arg\max_{u\in V_a(j)} \{\tau(j,u)\cdot\eta(j,u)^{\beta}\}, \\ \quad \text{if } q \leq q_0 \textit{ exploitation} \\ K, \quad \text{otherwise } (\textit{biased exploration}) \end{cases}$$

Figura 25- Regla de transición entre nodos

Esta regla de transición, favorece a las transiciones entre nodos conectados por arcos con pesos bajos y con un valor de feromona alto.

5.3.9- ACTUALIZACIÓN DE LA FEROMONA

Durante el proceso de creación de las posibles soluciones, las hormigas actualizan la feromona de los arcos que intervienen en los caminos encontrados. Ésta es la actualización local de feromonas. Para ello se aplica la siguiente fórmula:

$$\tau(j,k) = (1-\rho)\cdot\tau(j,k) + \rho\cdot\tau_0$$

Figura 26- Fórmula de actualización local de feromonas

Siendo:

- τ_0: El valor inicial de la feromona

Y el parámetro:

- ρ: $(0 < \rho < 1)$ Es el valor de evaporación de la feromona. Cuanto menor es ρ antes converge la solución, mientras que cuanto más alto es el valor, mayor es la diversidad de soluciones.

Después de cada iteración del algoritmo, cuando todas las hormigas han encontrado una solución, se aplica una actualización a las feromonas de forma global. En este caso, sólo la hormiga que ha encontrado la mejor solución actualiza las feromonas. Esto trata de dirigir la búsqueda de las hormigas en las siguientes iteraciones por los nodos vecinos de la mejor solución actual.

La fórmula de actualización de feromonas global es la siguiente:

$$\tau(j,k) = (1-\alpha) \cdot \tau(j,k) + \alpha \cdot \Delta\tau(j,k)$$

Figura 27- Fórmula de actualización global de feromonas

Siendo:

- $\Delta\tau(j,k)$: La longitud de la mejor solución encontrada $(L_{gb})^{-1}$ si el arco cuya feromona está siendo actualizada pertenece a la solución. Si el arco no pertenece a la solución, no se actualiza.

Y el parámetro:

- α: $(0 < \alpha < 1)$ Valor de evaporación de la feromona.

5.3.10-VALORES ÓPTIMOS DE LOS PARÁMETROS

De la definición del algoritmo definido en los puntos anteriores, se observa que distintos parámetros son necesarios para realizar los cálculos. Estos parámetros se deben especificar al inicio de cada cálculo de secuencia de vehículos que cruzan la zona de conflicto y sus valores son permanentes durante la ejecución del sistema GIA.

A partir de una serie de experimentos realizados con el algoritmo y distintos valores aplicados a cada uno de los parámetros, se han obtenido los valores óptimos que permiten realizar el cálculo de secuencias en un tiempo reducido sin sacrificar la calidad de los resultados.

Estos experimentos están descritos en el documento de referencia **[2]**. En la tabla siguiente se muestran los valores óptimos para cada parámetro:

Parámetro	Descripción	Valor
$\boldsymbol{\alpha}$	Evaporación de la feromona global	0,3
$\boldsymbol{\rho}$	Evaporación de la	0,1

	feromona global	
$\boldsymbol{\beta}$	Valor que denota la importancia de la información heurística en el cálculo	3
$\boldsymbol{q_0}$	Probabilidad de utilizar exploración VS explotación	0,1
$\boldsymbol{m}$	Número de hormigas	5
$\boldsymbol{C}$	Número de iteraciones	5

Tabla 6- Valores óptimos para el algoritmo de cálculo de secuencias

6- RESULTADOS OBTENIDOS

Para comprobar el correcto funcionamiento del simulador, se han realizado distintos experimentos modificando los parámetros del sistema y comparando los dos sistemas de gestión de cruce implementados utilizando el generador de gráficas del simulador.

En todos los casos se ha utilizado un procesador Intel Core i5-4570 a 3.20 GHz, 8GB de RAM y el sistema operativo Windows 10.

Inicialmente se validará el sistema con una intersección simple y a continuación se simulará una pequeña ciudad compuesta por cuatro intersecciones conectadas entre sí.

6.1. CRUCE INDIVIDUAL

Los primeros experimentos se han realizado con un único cruce y una densidad de tráfico baja.

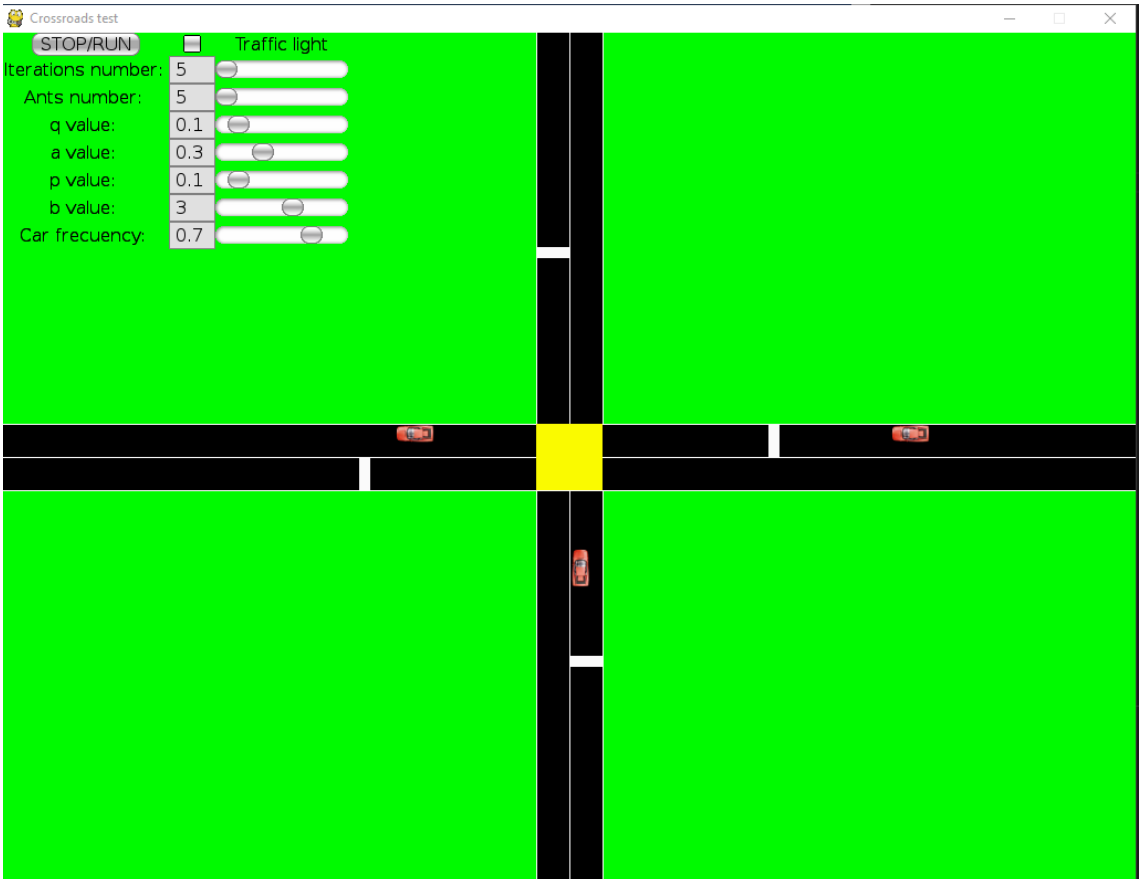

Figura 28- Configuración para pruebas de cruce individual

6.1.1. EXPERIMENTO 1

Se ha simulado tráfico aleatoriamente durante un minuto utilizando el sistema GIA de colonia de hormigas (ANT). Los parámetros utilizados han sido los siguientes:

Parámetro	Valor
α	0,3
ρ	0,1

$\boldsymbol{\beta}$	3
$\boldsymbol{q_0}$	0,1
$\boldsymbol{m}$	5
$\boldsymbol{C}$	5

Tabla 7- Parámetros utilizados en el experimento 1

A continuación se ha realizado el mismo experimento utilizando un semáforo para gestionar el cruce. El tiempo de transición del semáforo ha sido de 10s.

La siguiente gráfica muestra los resultados obtenidos utilizando ambos sistemas sobre el mismo tráfico:

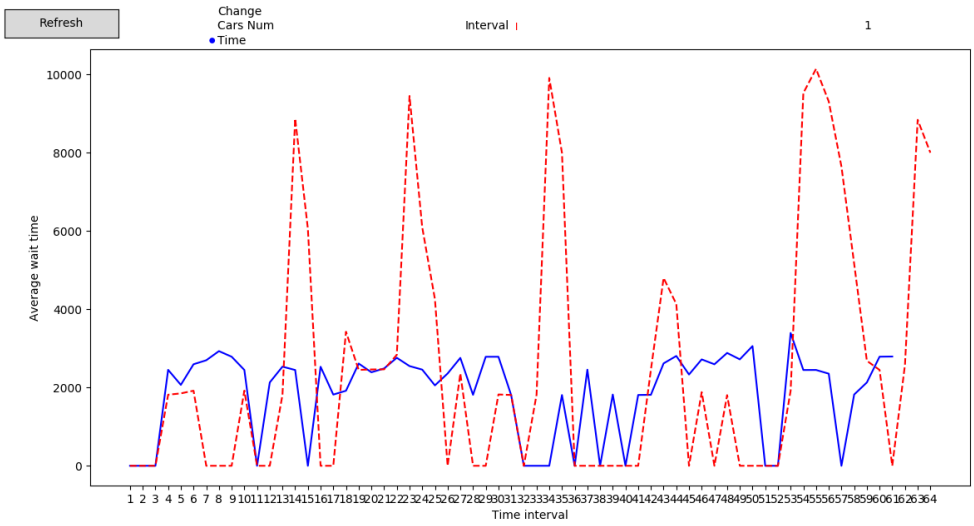

Figura 29- Gráfica de resultados del experimento 1 por intervalo de tiempo

En la gráfica se observa el tiempo medio que han tardado en cruzar los vehículos en cada intervalo de 1s. Los valores 0 se dan cuando ningún vehículo a cruzado en el último intervalo.

Los valores de la línea continua en azul corresponden al sistema GIA del algoritmo de hormigas, los valores en rojo y con línea discontinua, los del semáforo.

Los resultados muestran que utilizando un semáforo, mientras el cruce está cerrado, los tiempos que tardan en cruzar los vehículos pueden llegar a ser el tiempo de transición del semáforo. Debido a ello se observan los picos más altos en intervalos de 10s. Por otra parte, cuando el cruce está abierto, los tiempos medios pueden ser menores que usando el sistema ANT.

También se observa que utilizando el sistema ANT, los tiempos de cruce son más estables, lo que significa que el tráfico es más fluido y provoca menores esperas para los vehículos.

La siguiente gráfica muestra el tiempo que tardado cada vehículo en cruzar:

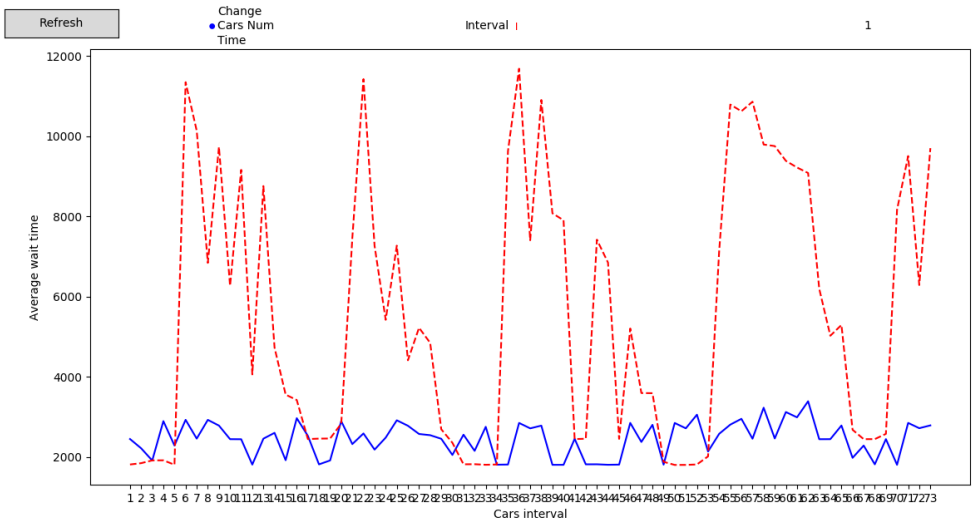

Figura 30- Gráfica de resultados del experimento 1 por intervalo de vehículos

6.1.2. EXPERIMENTO 2

En este experimento, se han utilizado los mismos datos y parámetros que en el experimento anterior. Sólo se ha cambiado el tiempo de transición del semáforo a 5 segundos para reducir los altos tiempos de espera producidos en el experimento anterior.

La siguiente gráfica muestra los resultados obtenidos:

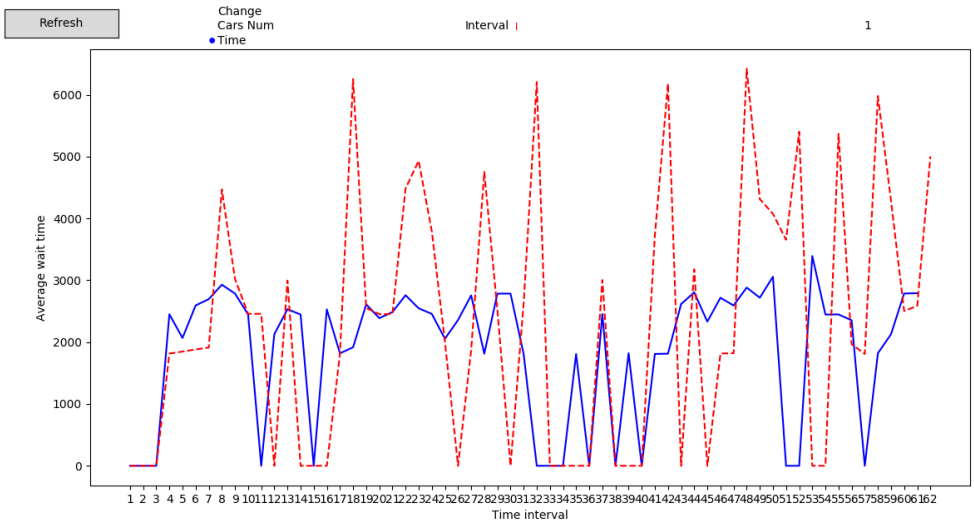

Figura 31- Gráfica de resultados del experimento 2 por intervalo de tiempo

En este caso, los valores obtenidos con el semáforo muestran tiempos de espera menores en los picos, pero hay más picos debido a que el tiempo de transición es menor.

Los tiempos de espera por vehículo han sido los siguientes:

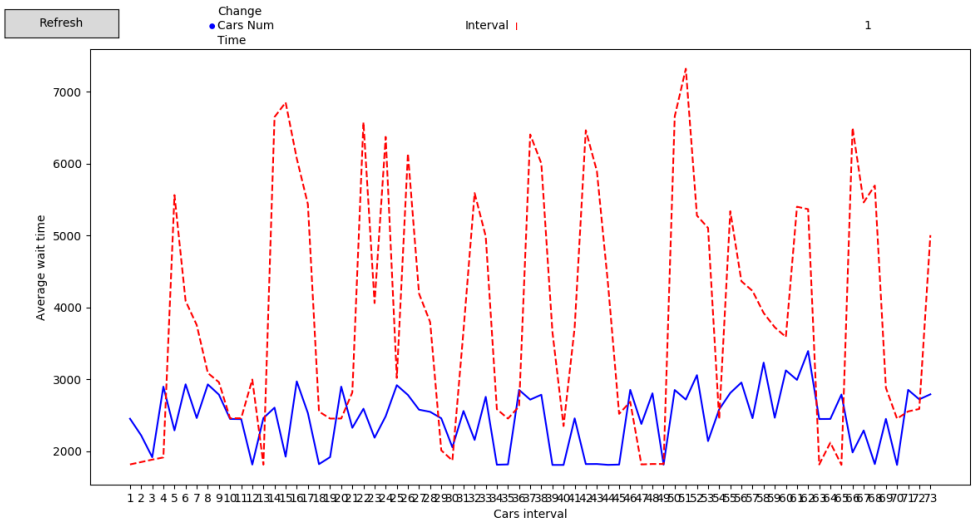

Figura 32- Gráfica de resultados del experimento 2 por intervalo de vehículos

6.1.3. EXPERIMENTO 3

En el experimento 3, se ha generado tráfico aleatorio con una alta densidad durante un minuto. A continuación se ha seguido los pasos del experimento 1 para obtener los resultados.

La siguiente figura muestra los tiempos medios de cruce en cada intervalo de 1 segundo:

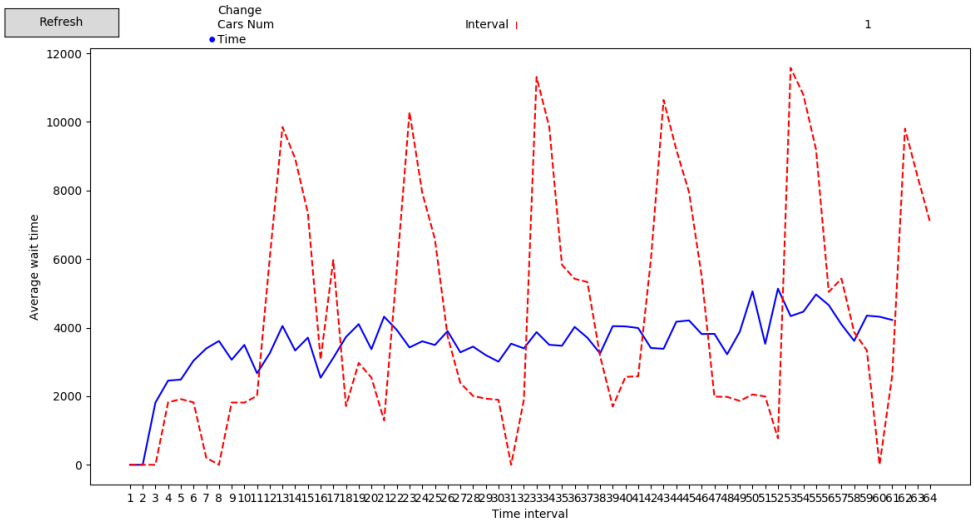

Figura 33- Gráfica de resultados del experimento 3 por intervalo de tiempo

Se observa que debido a que la densidad de tráfico es mayor, los tiempos de cruce del sistema ANT son mayores que en el experimento 1. Por otra parte, los tiempos de cruce usando el semáforo se mantienen próximos a los obtenidos en el experimento 1.

A continuación se muestran los tiempos de espera medios para cada grupo de 2 vehículos:

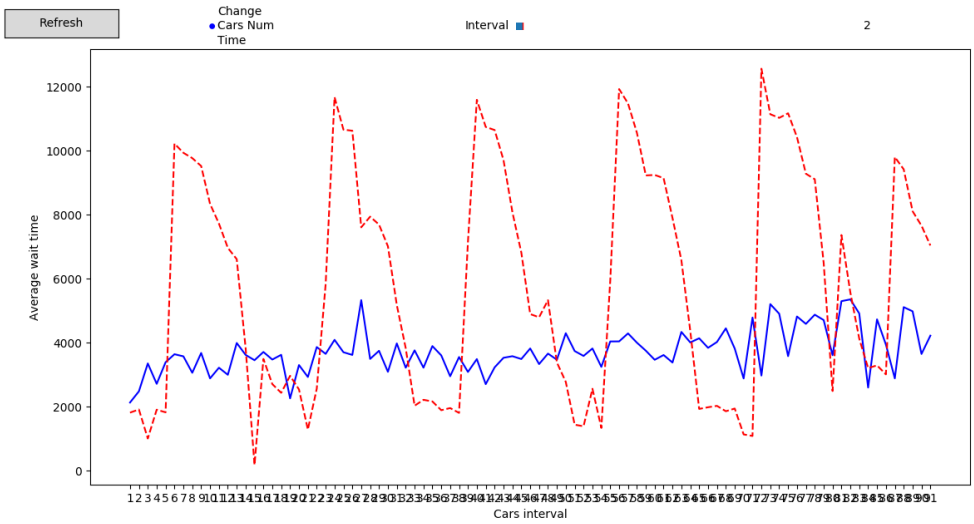

Figura 34- Gráfica de resultados del experimento 3 por intervalo de vehículos

En general, se siguen obteniendo mejores tiempos de cruce utilizando el sistema ANT que el semáforo.

6.1.4. EXPERIMENTO 4

En el experimento 4, se ha tratado de mejorar el tiempo de cruce de los vehículos modificando los parámetros del sistema. Para ello se ha utilizado el mismo tráfico generado en el experimento 3 y se han modificado los siguientes parámetros:

Parámetro	**Valor**
Longitud de la zona de control	150 -> 250
C	5 -> 10

Tabla 8- Parámetros modificados en el experimento 4

El número de vehículos que intervienen en el cálculo del orden de paso está limitado por la longitud de la zona de control. Cuando la zona de control está llena de coches en espera, el resto de vehículos quedan parados fuera de la zona de control y hasta que no entran, no se recalcula el orden de paso. Para abarcar un mayor número de vehículos en cada cálculo del algoritmo ANT y tratar de mejorar sus resultados, se ha incrementado la longitud de la zona de control y además se ha incrementado el número de hormigas. Con estos cambios se espera que los resultados del algoritmo ANT, mejoren los resultados obtenidos en el experimento anterior.

En la siguiente gráfica se muestran los resultados obtenidos con el algoritmo ANT en el experimento 3 en azul, y los del experimento actual en rojo:

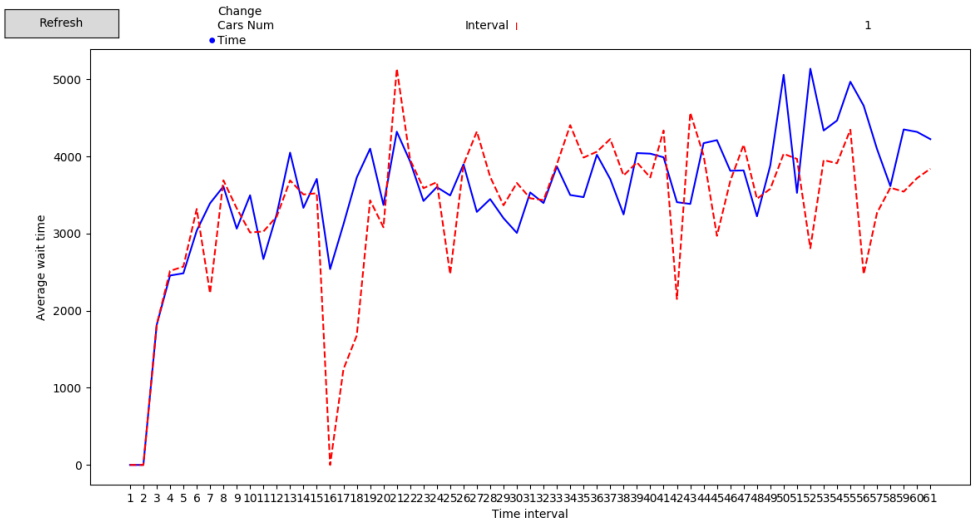

Figura 35- Gráfica de resultados del experimento 4 por intervalo de tiempo

Se observa que en algunos casos el tiempo de cruce ha empeorado, pero en general los resultados son mejores en el experimento 4.

A continuación se muestran los tiempos de espera medios por cada grupo de dos vehículos:

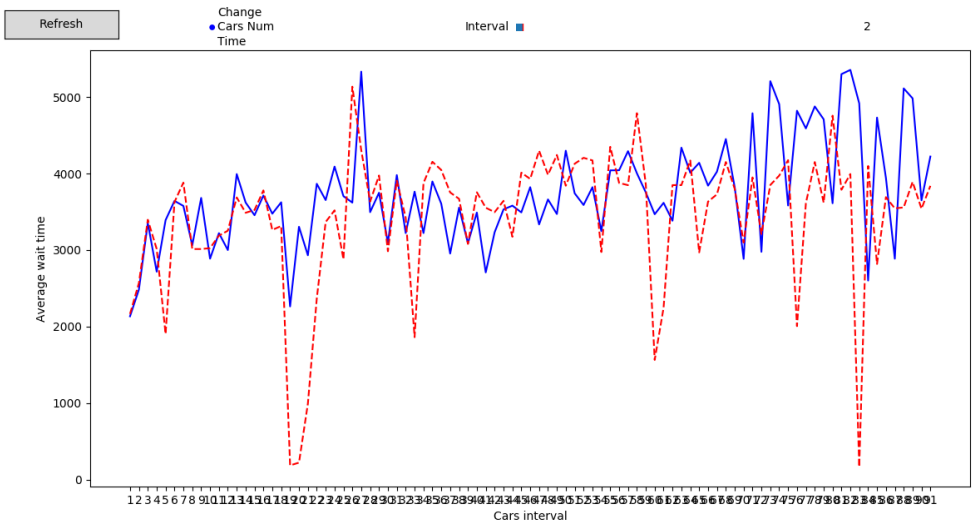

Figura 36- Gráfica de resultados del experimento 4 por intervalo de vehículos

6.1.5. CONCLUSIONES CRUCE INDIVIDUAL

Tras realizar los experimentos anteriores y estudiar los resultados obtenidos, se puede afirmar que la utilización de un algoritmo de hormigas gestiona un cruce de forma más eficiente que un semáforo.

Aunque en algunos casos el tiempo de cruce de los vehículos sea menor con el semáforo, en general, usando el sistema ANT los vehículos cruzan antes el cruce suponiendo un tiempo menor de espera para los conductores.

Además, ejecutando la simulación en tiempo real, se puede observar que mientras usando un semáforo los vehículos pueden llegar a quedar totalmente parados, usando el sistema ANT, mientras los vehículos esperan la orden de paso siguen circulando a baja velocidad realizando la aproximación a la zona de conflicto. Esto supone un ahorro de combustible ya que el mayor consumo se da en la aceleración desde que el vehículo está parado hasta que estabiliza su velocidad.

Con las pruebas realizadas, se confirma que los resultados obtenidos son los esperados y que el simulador funciona correctamente.

6.2. MULTICRUCE

Una vez comprobado el correcto funcionamiento del simulador utilizando un único cruce, se ha extendido la implementación del simulador para conectar distintos cruces y simular una pequeña ciudad en la que poder observar y comparar el comportamiento del tráfico usando los dos sistemas de gestión de cruce implementados.

Se han conectado cuatro cruces iguales con sus respectivos sistemas de control. La siguiente imagen muestra la configuración de cruces realizada:

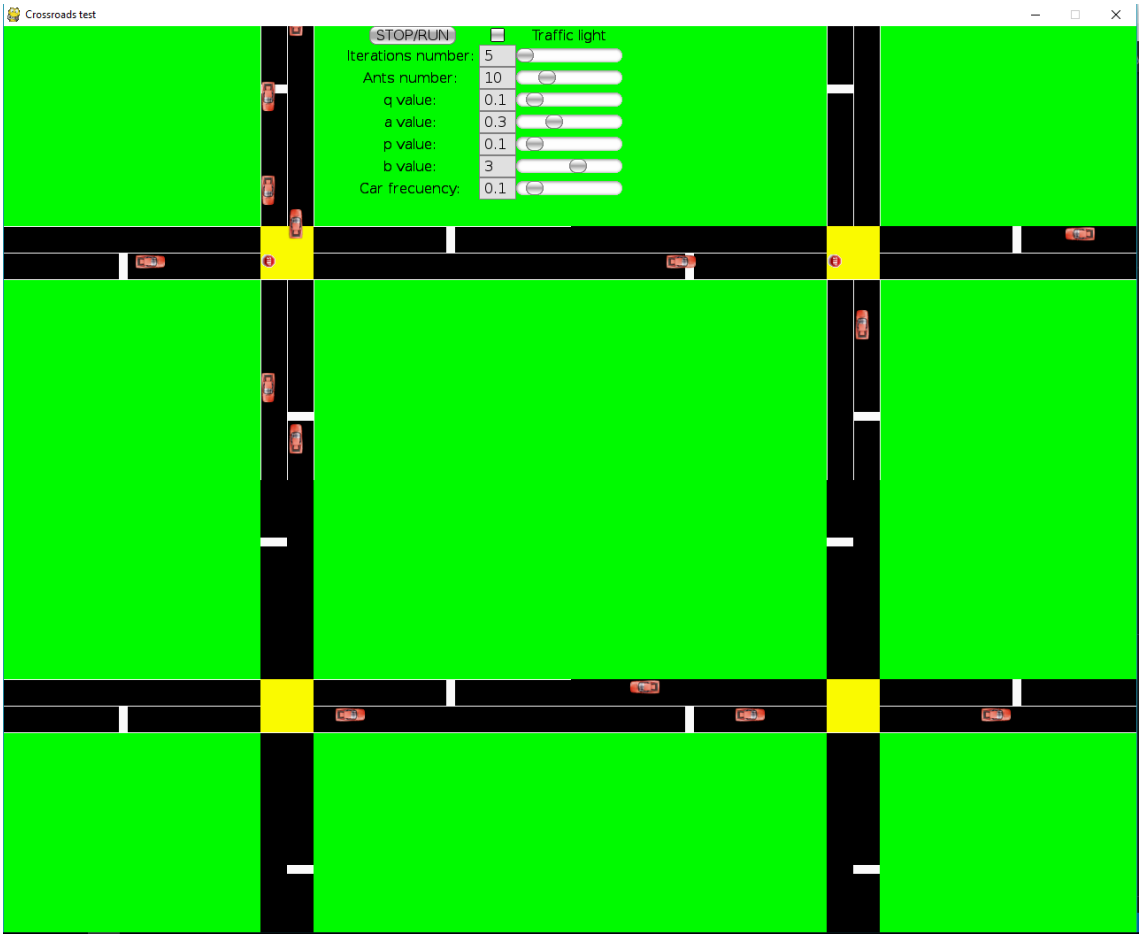

Figura 37- Configuración para pruebas multicruce

En este caso, sólo se realizará un experimento para comprobar que el sistema puede gestionar más de un cruce y conectarlos para crear configuraciones más complejas. Se espera que los

resultados obtenidos sean muy similares a los de los experimentos anteriores, ya que aunque haya más de un cruce, cada uno funciona de forma individual.

6.2.1. EXPERIMENTO 1

Para la realización del experimento, se ha generado tráfico de densidad alta aleatoriamente durante un minuto. Primero se ha generado usando el sistema de gestión ANT y a continuación el semáforo. En este experimento no se ha reutilizado el tráfico en ambos sistemas ya que el simulador no está adaptado para repetir experimentos usando más de un cruce.

En este experimento, los datos de tiempo de cruce abarcan la ciudad completa, es decir, que el tiempo de cruce se calcula desde que el vehículo entra a la ciudad hasta que sale. El tiempo total de cruce debe ser cercano a los tiempos obtenidos en experimentos anteriores ya que la longitud que recorre cada vehículo es la misma aunque haya más cruces.

Otro detalle a destacar es que cuando se usan semáforos, todos los semáforos están sincronizados. Todos tienen el mismo tiempo de transición y cambian en el mismo sentido. En todo momento sólo una dirección de la ciudad está abierta, la vertical o la horizontal.

A continuación se muestran los resultados obtenidos. Los valores en azul representan los resultados del sistema ANT y en rojo los de los semáforos.

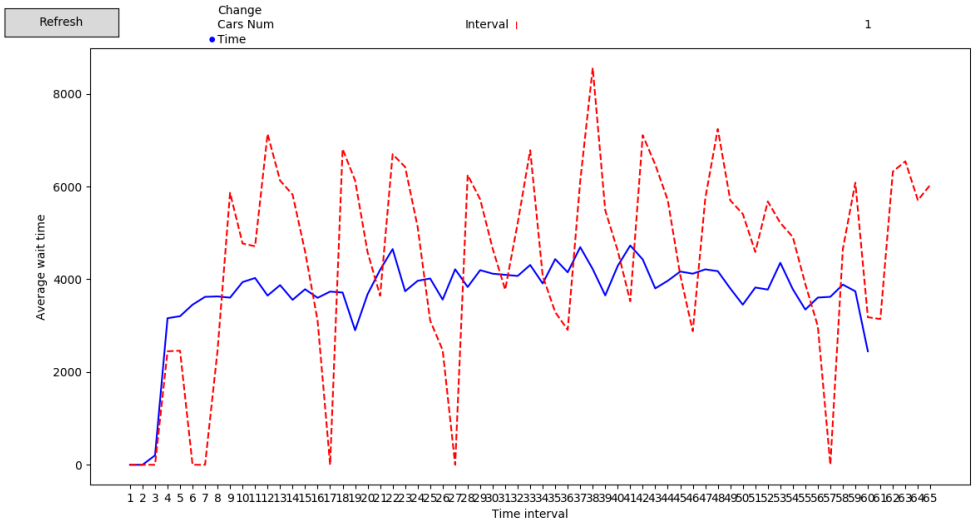

Figura 38- Gráfica de resultados del experimento 4 por intervalo de tiempo

Se puede observar que los resultados son muy parecidos a los resultados obtenidos en los experimentos anteriores. Una diferencia importante se observa en los datos de los semáforos. Hay menos caídas a valores 0 debido a que en la transición de estado algún vehículo puede quedar parado en medio de 2 cruces.

La siguiente gráfica muestra los resultados para cada grupo de dos vehículos:

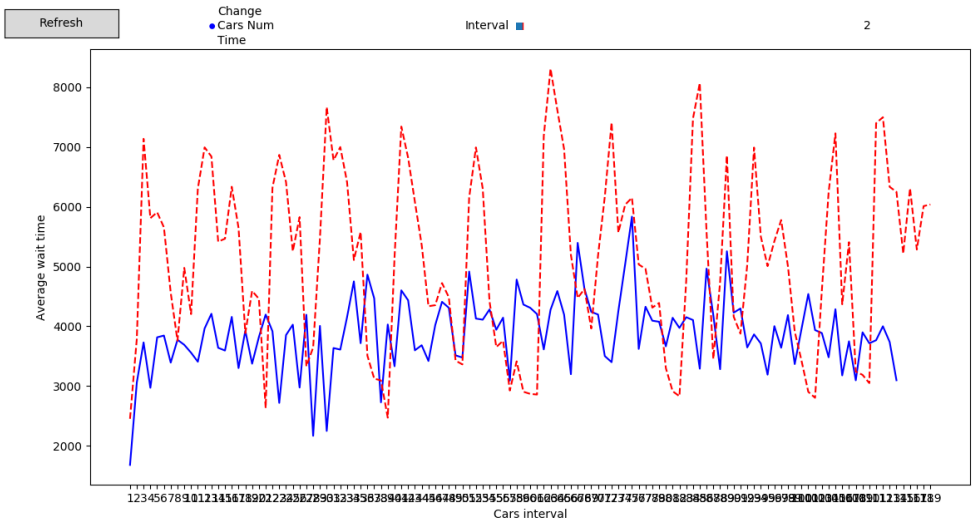

Figura 39- Gráfica de resultados del experimento 4 por intervalo de vehículos

6.2.2. CONCLUSIONES MULTICRUCE

Hemos podido observar con el experimento anterior que usar un algoritmo ANT para gestionar los cruces de una ciudad es más eficiente que usar semáforos. Los resultados obtenidos son muy próximos a los obtenidos en los experimentos realizados con un único cruce. Como diferencia a destacar, se detecta que los tiempos de cruce utilizando semáforos empeoran con el número de cruces por los que circula el vehículo. Cuantos más cruces, hay más posibilidades de que un vehículo quede parado entre dos cruces antes de llegar al final de la ciudad.

De la misma forma que se ha concluido con los experimentos realizados sobre un cruce individual, concluimos con el experimento usando multicruce confirmando el correcto funcionamiento del simulador. Debido a que los resultados son los esperados y a las observaciones realizadas en la interfaz gráfica del simulador durante su ejecución, se puede realizar la confirmación.

7- CONCLUSIONES FINALES

En este último capítulo, se concluye resumiendo el trabajo realizado, los problemas que se han encontrado a lo largo de su desarrollo y las posibles mejoras que se pueden realizar en trabajos futuros.

7.1. TRABAJO REALIZADO

Se ha desarrollado un simulador de tráfico que permite comparar diferentes estrategias o algoritmos de coordinación en los cruces. Además se puede observar en tiempo real como se comportan los vehículos a través de la intersección con cada técnica aplicada.

Usando este simulador, se pueden generar distintas gráficas que representan los tiempos de cruce de los vehículos y modificar los parámetros del sistema para realizar distintos experimentos.

Para poder validar su funcionamiento y realizar la comparación de distintos sistemas de gestión de cruce, se han implementado dos sistemas de gestión, el control por semáforos y el control mediante distintas técnicas de inteligencia artificial.

La implementación del sistema usando IA, se basa en el trabajo realizado previamente por Jia Wu, AbdelJalil Abbas-Turki, Abdellah El Moudni llamado *Cooperative driving: an ant colony system for autonomous intersection management.* En este trabajo se presenta una técnica de gestión de las órdenes de cruce sobre los vehículos utilizando un algoritmo de colonia de hormigas. Con esta referencia se ha implementado el algoritmo presentado y adaptado al simulador de tráfico.

Además de los dos sistemas de gestión implementados, el simulador permite añadir nuevos sistemas de gestión fácilmente cumpliendo únicamente que los parámetros de entrada y salida sean los correctos.

Todo el código del simulador implementado se puede encontrar en el siguiente repositorio:

https://github.com/aferrandoa/TFM1617-CrossScheduleSim

Una vez el simulador y los sistemas de gestión estaban finalizados, se ha procedido a validar su funcionamiento. Se realizaron distintos experimentos variando los parámetros del sistema. Los resultados obtenidos has sido satisfactorios, confirmando que el uso de técnicas de inteligencia artificial supone una mejora en la eficiencia de la gestión de cruce de vehículos por una intersección.

Llegados a este punto del trabajo, los objetivos propuestos han sido cumplidos y se ha procedido a ampliar la implementación del simulador, creando una pequeña ciudad compuesta por varios cruces.

Tras finalizar la nueva implementación del simulador. se ha realizado un nuevo experimento usando el conjunto de cruces como base. Los resultados obtenidos han sido los esperados, confirmando el correcto funcionamiento del simulador. Además, se ha confirmado que el uso de técnicas de inteligencia artificial para gestionar los cruces de la ciudad, es más eficiente que el uso de semáforos.

7.2. PROBLEMAS ENCONTRADOS

Durante la realización del trabajo se han detectado distintos problemas que han quedado sin resolver:

1- La escala de los cruces no es configurable, esto produce que para crear ciudades con más de cuatro cruces, se necesite una resolución mayor a 1920x1080 píxeles.
2- Cuando se acumulan muchos vehículos en espera en un cruce, si no caben más vehículos en pantalla, estos comienzan a superponerse. Aunque no causa ningún problema a la simulación, es físicamente imposible y quita realismo al simulador.
3- Cuando se dibujan 4 cruces, los cruces de la parte inferior de la pantalla no dibujan correctamente sus líneas blancas de las direcciones verticales.

Dado que los problemas encontrados son problemas de la interfaz gráfica y no afectan a la simulación, se han dejado para resolver en futuras ampliaciones y que los cambios pueden ser bastante complejos.

7.3. POSIBLES AMPLIACIONES

Para finalizar, se comentarán distintas ampliaciones al trabajo que podrían realizarse en el futuro. Estas propuestas son fruto de las distintas ideas que han surgido durante la implementación de los distintos componentes del trabajo.

La primera ampliación propone incrementar el número de carriles en cada dirección del cruce. Esto permitirá crear simulaciones más realistas y complejas. Actualmente el sistema de gestión ANT permite añadir nuevos carriles sin modificar el algoritmo. El mayor esfuerzo en esta implementación deberá realizarse en la interfaz gráfica del simulador para dibujar el cruce correctamente y adaptar el loop principal para que soporte los nuevos carriles.

Otra implementación importante sería permitir que los vehículos hagan giros en el cruce. Igual que en la mejora anterior, el mayor esfuerzo se debería realizar en la interfaz gráfica ya que actualmente los vehículos sólo circulan en línea recta. Respecto al sistema de gestión ANT, soportaría este cambio únicamente configurando correctamente los vehículos que entran en el cálculo del orden. Por ejemplo, si un vehículo en dirección Este gira a la izquierda en dirección Norte, el carril con dirección Oeste, pasaría a ser un carril de conflicto y no permitiría la circulación simultánea con el carril dirección Este.

Por último, como mejora a los sistemas de gestión, se propone distinguir vehículos VIP del resto. Estos vehículos representarían servicios de urgencias como ambulancias, bomberos, policía... Los sistemas de gestión podrían adaptar sus cálculos para beneficiar a los carriles con vehículos VIP presentes y así permitir que atraviesen la intersección en el menor tiempo posible.

En el momento de calcular la secuencia óptima, como primer paso se deberían identificar los vehículos especiales dentro del conjunto de coches bajo control. Una vez se han identificado, dejaría de buscarse la secuencia óptima de cruce para todos los vehículos, y se trataría de obtener aquella secuencia en la que los vehículos VIP cruzan lo antes posible. En el algoritmo ANT, se podrían actualizar las feromonas en mayor o menor grado dependiendo de la posición de los vehículos VIP en las secuencias de paso, así, aunque los vehículos especiales tengan preferencia, el resto de vehículos también cruzarían en el menor tiempo posible. Otra opción es

abrir el carril por el que circule un vehículo VIP e ignorar el resto de carriles hasta que haya cruzado.

8- BIBLIOGRAFÍA

[1] Cohen S *1993, Ingénieriedu trafic routier.*

[2] Jia Wu, AbdelJalil Abbas-Turki, Abdellah El Moudni. *2012, Cooperative driving: an ant colony system for autonomous intersection management.*

[3] Fei Van, Mahjoub Dridi and Abdellah EL Moudni. *A Branch and Bound Algorithm for New Traffic Signal Control System of an Isolated Intersection.*

[4] Marco Dorigo and Luca Maria Gambardella. 1997, *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem*

[5] Jia Wu, Abdeljalil Abbas-Turki, Aurelien Corre&ia and Abdellah El Moudni, *Discrete Intersection Signal Control*

[6] Wikipedia: Automóvil *https://es.wikipedia.org/wiki/Autom%C3%B3vil*

[7] Macrosimulation of traffic flow *http://www.traffic-simulation.de/*

Historia del automóvil:

[8] Monografias.com *http://www.monografias.com/trabajos15/automovil-historia/automovil-historia.shtml*

[9] historiadelautomov.blogspot.com.es *http://historiadelautomov.blogspot.com.es/*

[10] elperiodico.com *http://www.elperiodico.com/es/motor/noticias/innovacion/conectividad/los-mejores-avances-tecnologicos-para-coches-actuales-4358768*

[11] Wikipedia *https://es.wikipedia.org/wiki/Historia_del_autom%C3%B3vil*

[12] política.elpais.com *http://politica.elpais.com/politica/2017/01/03/actualidad/1483437138_795013.html*

El vehículo autónomo:

[13] Wikipedia *https://es.wikipedia.org/wiki/Veh%C3%ADculo_aut%C3%B3nomo*

[14] autocasion *https://www.autocasion.com/actualidad/reportajes/funciona-coche-autonomo*

[15] Wikipedia: LIDAR *https://es.wikipedia.org/wiki/LIDAR*

[16] Wikipedia: Algoritmo de colonia de hormigas *https://es.wikipedia.org/wiki/Algoritmo_de_la_colonia_de_hormigas*

[17] Python *https://www.python.org/*

[18] Pygame *https://www.pygame.org/news*

[19] Pyplot *https://matplotlib.org/api/pyplot_api.html*

ANEXO A- INSTALACIÓN

A continuación se describirá el proceso a seguir para instalar y ejecutar el simulador implementado en el trabajo.

A.1- DESCARGA DEL CÓDIGO FUENTE

El código fuente se encuentra disponible en el siguiente repositorio de github:

https://github.com/aferrandoa/TFM1617-CrossScheduleSim

Usar la opción de descargar en ZIP y descomprimirlo en la carpeta de trabajo que se desee.

A.2- INSTALACIÓN DE PYTHON Y LIBRERÍAS NECESARIAS

Para instalar python, se requiere descargar el instalador de la version 2.7 desde el siguiente enlace:

https://www.python.org/ftp/python/2.7.13/python-2.7.13.msi

Instalar el paquete descargado y acceder a la consola del sistema para instalar las dependencias necesarias. Se deben ejecutar los siguientes comandos:

- Pygame: *python –m pip install pygame –user*
- Pyplot: *python -m pip install -U pip setuptools*
  python -m pip install matplotlib

Para ejecutar el simulador se requiere además la librería de componentes de interfaz de usuario PGU. Esta librería no se encuentra disponible en el instalador de dependencias PIP. Para su instalación se debe descargar la última versión de la librería desde el siguiente enlace:

https://code.google.com/archive/p/pgu/downloads

Una vez descargado el ZIP, se debe ejecutar el siguiente comando en la consola para añadirlo a las librerías de PYTHON:

python pip install "ruta-al-fichero/pgu-X.XX.zip

A.3- EJECUCIÓN DEL SIMULADOR

Existen 3 ficheros principales para ejecutar el simulador. Los ficheros se encuentran en la ruta “pythonProject\gui” y son los siguientes:

- GuiMain.py: Ejecuta el simulador con 1 cruce.
- GuiMainMulti2.py: Ejecuta el simulador con 2 cruces.
- GuiMainMulti4.py: Ejecuta el simulador con 4 cruces.

Para ejecutar el simulador se debe navegar al directorio “pythonProject\gui” usando la consola del sistema y ejecutar el siguiente comando:

python GuiMain.py

Para ejecutar el generador de gráficas, se debe navegar al directorio “pythonProject” y ejecutar el siguiente comando:

python plot\plotresults.py

En el caso de que haya ejecutado el simulador previamente, cargará por defecto el último experimento.

Los ficheros de resultados generados se pueden encontrar en la ruta “\pythonProject\gui\experiments”. Si se desea comparar 2 ficheros de resultados usando el generador de gráficas, se deben copiar los 2 ficheros en la carpeta “experiments” con los nombres “LOGS.out” y “LOGS_REP.out”

A.4- CONFIGURACIÓN INICIAL

La configuración inicial del simulador se encuentra en el fichero:

pythonProject\gui\GuiConstants.py

En este fichero se pueden modificar los valores iniciales de los parámetros del simulador antes de ejecutarlo. Los valores más importantes son los siguientes:

- RUN_CONTROL: Si es 1, inicia el simulador en marcha. Si es 0, se inicia en pausa.
- USE_TRAFFIC_LIGHT: Si es 0, el simulador se inicia utilizando el algoritmo ANT por defecto. Si es 1, utiliza el semáforo.
- CAR_ADD_MODE: Si es “R”, El simulador genera tráfico aleatorio. Si es “F”, carga el tráfico utilizando los datos de un fichero de logs previo. Para poder utilizar la opción “F”, debe existir un fichero de eventos con el nombre “LOGS_REP.out” en la carpeta de experimentos del simulador.
- RANDOM_RATE : Tasa de aleatoriedad del tráfico inicial. Debe ser un valor en 0 y 1. Cuanto más próximo a 0, mayor es la densidad de tráfico.
- CONTROL_DISTANCE: Determina la distancia a partir de la cual los vehículos entran en la zona de control de los cruces. Por defecto es 150.

ANEXO B- CONFIGURACIÓN MULTICRUCE

En este anexo se explicará como añadir cruces al simulador y como conectarlos entre ellos.

B.1- CREACIÓN DE CRUCES

El primer paso para crear un cruce, es definir los carriles que contendrá. Para definir los carriles, se debe crear un array de objetos GuiLine.

El constructor de GuiLine requiere los siguientes parámetros:

- Direction: Sentido de la circulación. Puede tomar los valores (N, S, E, W)
- Length: Longitud del carril
- Position: Coordenadas (X, Y) del punto central del carril

En el simulador implementado, la función ini_lines de la clase principal, se encarga de crear grupos de cuatro carriles para cada uno de los cruces existentes.

Una vez se han definido los carriles del cruce, se debe crear el objeto que representa al cruce. Este objeto es de tipo GuiCross y requiere el parámetro:

- Center: Coordenadas (X, Y) del centro del cruce

Para terminar, se debe crear el objeto encargado de de controlar el tráfico por el cruce. Este objeto es de tipo GuiCrossControl y requiere un único parámetro que el objeto GuiCross creado en el paso anterior.

Para definir más de un cruce, se deben seguir los pasos anteriores tantas veces como cruces se quieran añadir a la simulación. Para realizar la conexión entre los cruces creados, únicamente hay que añadir todos los grupos de carriles creados al controlador del evento que detecta la salida de los vehículos de un cruce. Si el vehículo que ha salido de un cruce colisiona con un nuevo cruce, éste es añadido al cruce correspondiente.

El controlador del evento de salida se encuentra en el bucle principal:

```
if event.type == pygame.USEREVENT:
    add_car_event([lines, lines2, lines3, lines4], event.car)
```

Los objetos “linesX”, repesentan cada uno de los grupos de carriles creados para 4 cruces distintos.

B.2- SIMULACIÓN DE LOS CRUCES

Durante el bucle principal del simulador, se debe llamar a distintas funciones para que se simule el tráfico en cada uno de los cruces creados y se ejecute el gestor de cruce seleccionado cuando sea necesario.

Para todos los grupos de carriles creados y para cada uno de los carriles que contienen, se debe llamar a la función update. La función update tiene un único parámetro que corresponde con el tiempo transcurrido desde el último frame en milisegundos. Esta función se encarga de actualizar las físicas del simulador y reposicionar los objetos dependiendo del tiempo transcurrido.

```
for current_line in lines:
   current_line.update(dt)
for current_line in lines2:
   current_line.update(dt)
```

Una vez la simulación ha sido actualizada, se debe comprobar si el controlador debe actuar y calcular la secuencia de paso por el cruce en el caso de que sea necesario. La función que ejecuta el controlador es check_control_entries. Esta función tiene como parámetro el grupo de carriles del cruce bajo control. Debe llamarse a esta función una vez por cada cruce de la simulació.

```
cross_control.check_control_entries(lines)
cross_control2.check_control_entries(lines2)
```

Tras los pasos de simulación, se deben redibujar todos los objetos en pantalla. Para ello se llama a las siguientes funciones draw de los carriles y cruces:

```
#Dibujado de los carriles
for current_line in lines:
    current_line.draw()
for current_line in lines2:
    current_line.draw()

#Dibujado de los cruces
rossroad.draw()
crossroad2.draw()
```

Para terminar requiere llamar a la función render de los carriles. Esta función dibuja los vehículos sobre el sistema de cruces.

```
for current_line in lines:
    current_line.render()
for current_line in lines2:
    current_line.render()
```

En el siguiente enlace se puede consultar la implementación del simulador para un sistema de cuatro cruces interconectados. Todo el código de ejemplo mostrado en este anexo, se ha extraído del fichero referenciado:

https://github.com/aferrandoa/TFM1617-CrossScheduleSim/blob/master/pythonProject/gui/GuiMainMulti4.py