UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Coordinación inteligente en vehículos autónomos

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Blasco Peiró, Andrea

Tutor: Palanca Cámara, Javier
Julián Inglada, Vicente Javier
Carrascosa Casamayor, Carlos

Curso 2016-2017

Resum

La finalitat d'aquest projecte es la creació d'una eina disenyada per a poder simular el comportament d'una flota de vehicles, en la que es poden provar distintes estrategies de coordinació en diferents contextos i veure quina es més òptima a diferents nivells.

Els vehícles s'han implementat com sistemes intel·ligents capaços de comunicar-se i coordinarse. Cada vehicle es un agent del sistema que deu comunicarse amb la resta per a poder dur a terme objectius complexos.

S'han implementat i provat tres estrategies amb les que s'ha fet un anàlisi dels resultats combinant-les amb distintes simulacions.

El sistema està desenvolupat en Python i la interfície visual de l'aplicació està feta amb Javascript.

Paraules clau: Agents Intel·ligents, Sistemes Multiagent, Vehícles autònoms, Coordinació

Resumen

La finalidad de este proyecto es la creación de una herramienta diseñada para poder simular el comportamiento de una flota de vehículos, en la que se pueden probar distintas estrategias de coordinación en diferentes contextos y ver cuál es más óptima a diferentes niveles.

Los vehículos se han implementado como sistemas inteligentes independientes capaces de comunicarse y coordinarse. Cada vehículo es un agente del sistema que debe comunicarse con el resto para poder llevar a cabo objetivos complejos.

Se han hecho y probado tres estrategias con las que se ha realizado un análisis de los resultados combinándolas con distintas situaciones.

El sistema esta desarrollado en Python y la interfaz de usuario de la aplicación esta hecha en Javascript.

Palabras clave: Agentes Inteligentes, Sistemas Multiagente, vehículos autónomos, Coordinación

Abstract

The purpose of this project is the creation of a tool designed to be able to simulate the behavior of a fleet of vehicles, in which different strategies can be tested in different contexts and see which one is more optimal according to different levels.

The vehicles have been implemented as intelligent systems capable of communicating and coordinating. Each vehicle is an agent of the system that must communicate coordinate whith the rest to carry out complex objectives.

Three strategies have been done and tested. Thanks to that, an analysis of the results has been made combining them with different situations.

The system is developed in Python and the user interface of the application is made in Javascript.

Key words: Intelligent agents, Multiagent systems, Autonomous vehicles, Coordination

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

En este capítulo vamos a hablar de la motivación y los objetivos de este proyecto.

1.1 Motivación

La motivación de este trabajo viene dada por la necesidad de optimización de los medios de transporte, tanto a nivel de satisfacción del cliente como para ahorrar el máximo posible en carburante y, a su vez, transformar las ciudades en sitios más limpios y sanos.

Actualmente, la cantidad diaria de automóviles dedicados al transporte público, privado o de mercancías circulando por las calles contribuye a crear un entorno en el que el aire en algunos sitios no se puede respirar. Para mejorar los niveles de polución y reducir el tráfico, una buena solución es fomentar el uso del transporte público. En este trabajo en concreto vamos a centrarnos en mejorar el uso del taxi y hacer que su uso sea lo más eficiente posible mirando desde varios puntos de vista.

Todos hemos podido observar hoy en día la potencia de la inteligencia artificial en fábricas de producción en las que hay robots haciendo un mismo trabajo. Cuando piensas en este tipo de labores hechas por humanos es fácil a veces olvidar un apartado muy importante, la coordinación.

No es que se olvide, se obvia. Ahora, si todos los robots que podemos ver en una fábrica haciendo complejos trabajos conjuntos pueden coordinarse y así hacer un mejor trabajo y más eficiente, también pueden coordinarse todos esos automóviles circulando durante horas todos los días.

Si todos los automóviles pudiesen coordinarse, elegir las rutas más rápidas y más eficientes, los tiempos de circulación serían menores y los gases emitidos a la atmósfera diariamente se reducirían, esto se resume en un ambiente más saludable, menos ruidoso y, por supuesto, económicamente hablando menos dinero invertido en carburantes.

Existen muchas empresas dedicadas al transporte que podrían mejorar su rendimiento coordinando las distintas flotas de vehículos. Solamente aplicando estas técnicas en este sector ya influiría para mejorar la movilidad sostenible (reduciendo el tráfico y la polución)

Además, en los últimos años empresas como Uber o Cabify están ganando mucho terreno a los taxis locales tradicionales, esto es gracias a la adaptación a los nuevos tiempos y a saber aprovechar las tecnologías existentes.

Ya han habido varias manifestaciones debidas al descontento de los taxistas ante la invasión del sector por parte de dichas empresas. Vista la situación, una posible solución para los taxis independientes sería colaborar entre ellos y facilitar a los clientes el

contacto con ellos a través de aplicaciones móviles. Y otra forma de poder optimizar el rendimiento de los transportes es el uso de aplicaciones como esta para la planificación de trayectos.

1.2 Objetivo

El objetivo de este proyecto es la implementación de una herramienta para probar distintas estrategias de coordinación de taxis para asignarlos a los clientes y ver cual es la que mejor funciona. En este caso, nos hemos centrado en coordinar una flota de taxis que deben recoger y dejar a los clientes que van apareciendo en tiempo real en sus respectivos destinos, minimizando así tanto los tiempos de circulación como los tiempos de espera de los clientes. La finalidad de esta herramienta es que las personas que deseen probar estrategias y ver su rendimiento, puedan usar esta aplicación de simulación que calcula tiempos específicos y generales además de distancias recorridas.

1.2.1. Objetivos específicos

A continuación vamos a detallar los objetivos específicos de este trabajo.

- Crear una interfaz en la que se pueda visualizar sobre un mapa una simulación de taxis y clientes y su interacción.
- Poder configurar la simulación desde la interfaz y que se visualicen los resultados al terminar.
- Crear una estructura por módulos que puedan ser modificados y reemplazados.
- Implementar distintas estrategias de coordinación de taxis.
- Hacer pruebas de las distintas estrategias implementadas con diferentes situaciones y escenarios.
- Hacer un análisis de los resultados obtenidos de las pruebas.

1.3 Estructura de la memoria

El documento se ha organizado en 8 capítulos.

En el primer capítulo hablamos de la motivación del proyecto y su objetivo junto a los objetivos específicos. se hablado del contexto en el que nos encontramos y la motivación de este proyecto. En el estado del arte nos introducimos en el contexto actual y en los agentes inteligentes.

El capítulo de análisis y requisitos se harán las especificaciones de la aplicación y en el cuarto se hablará de las tecnologías empleadas y sobre como se ha montado junto con algunas imágenes de la aplicación.

El cuarto capítulo se explicarán todas las tecnologías que se han usado para la implementación de la herramienta.

En el quinto capítulo se hablará del diseño de la aplicación y se explicarán todas sus partes.

El sexto se explicarán las estrategias de coordinación implementadas y como se van a organizar las pruebas. en el séptimo se hará un análisis de los resultados obtenidos en las pruebas.

En el capítulo siete se expondrán las conclusiones sacadas de este proyecto.

Además se añadirán dos apéndices, en el primero se hará un manual de uso en el que se explicará cada uno de los módulos detalladamente, con el fin de que los usuarios puedan probar la aplicación y hacerle las modificaciones que quieran. En el segundo se explicará la configuración del sistema para su uso y manipulación.

CAPÍTULO 2
Estado del arte

En este capítulo vamos a hablar del estado del arte de los agentes inteligentes, la coordinación y de los automóviles autónomos.

2.1 Agentes inteligente

Existen diversas definiciones distintas de agente inteligente:

- "Un agente inteligente, es una entidad capaz de percibir su entorno, procesar tales percepciones y responder o actuar en su entorno de manera racional, es decir, de manera correcta y tendiendo a maximizar un resultado esperado."[1]

- "Un agente inteligente es un programa especialmente concebido para realizar ciertas tareas de manera autónoma en una red por encargo de un usuario" [2]

- "Como una entidad software que, basándose en su propio conocimiento, realiza un conjunto de operaciones destinadas a satisfacer las necesidades de un usuario o de otro programa, bien por iniciativa propia o porque alguno de éstos se lo requiere." - Hípola y Vargas-Quesada (1999) [3]

- "Pieza de software que ejecuta una tarea dada utilizando información recolectada del ambiente, para actuar de manera apropiada hasta completar la tarea de manera exitosa. El software debe ser capaz de auto-ajustarse basándose en los cambios que ocurren en su ambiente de forma tal que un cambio en las circunstancias producirá un resultado esperado." - Jiménez y Ramos (2000)[4]

En resumen, la misión de un agente inteligente es la ejecución de unas tareas en provecho de los usuarios en un entorno conocido y del cual perciben cambios y actúan en consecuencia a ellos de manera racional y tendiendo a maximizar el resultado esperado.

Las características de los agentes inteligentes son:

- **Continuidad temporal**: se considera un agente un proceso sin fin, con un objetivo continuo no finito, están en ejecución continuamente desarrollando su función.

- **Autonomía**: un agente no necesita la intervención de un humano directamente ya que es capaz de actuar autónomamente basándose en su experiencia siendo capaz de adaptarse al entorno en el que se encuentra aunque este cambie.

- **Sociabilidad**: el agente es capaz de comunicarse no solamente con otros agentes, también con otras entidades, incluidas las personas, por medio de un lenguaje común.
- **Veracidad**: se asume que un agente siempre dirá la verdad.
- **Benevolencia**: los agentes se ayudarán unos a otros siempre que al hacerlo no generen conflictos con sus propios objetivos.
- **Racionalidad**: el agente siempre realiza «lo correcto» a partir de los datos que percibe del entorno.
- **Reactividad**: un agente debe percibir su entorno y actuar en base a esos cambios, es decir, el agente reacciona ante los cambios.
- **Pro-actividad**: el agente es pro-activo cuando es capaz de controlar sus propios objetivos a pesar de cambios en el entorno, es decir, tiene un propósito y emprenderá las acciones necesarias hasta conseguirlo.
- **Adaptabilidad**: está relacionado con el aprendizaje que un agente puede tener basándose en las acciones y las consecuencias que la experiencia le ha dado e intenta cambiar su comportamiento.
- **Movilidad**: capacidad de un agente de moverse a través de una red.

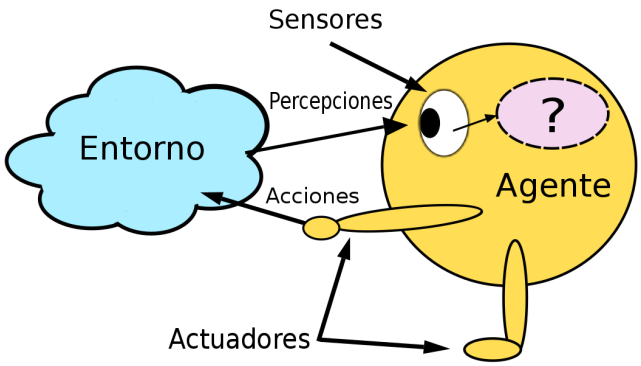

Figura 2.1: Representación gráfica de un agente inteligente

Los sistemas multi-agente, son conjuntos de agentes inteligentes trabajando juntos con un mismo objetivo. Y, como hemos podido ver en sus características, estos agentes poseen capacidades de comunicación, negociación y coordinación que son aprovechadas para poder llevar a cabo tareas conjuntamente con otros agentes.

Esto se consigue mediante las características de sociabilidad, ya que necesitan comunicarse, benevolencia, cada agente ayudará a los demás si lo requieren y la veracidad, todos los agentes asumirán que toda información que reciban de otros agentes será cierta.

2.2 Vehículos autónomos

Los coches autónomos ya son una realidad, hay algunas ciudades, aunque pocas, que ya cuentan con ellos y siguiendo la tendencia de la tecnología en general probablemente acaben sustituyendo a los automóviles conducidos por personas, la ventaja de ello, además de la reducción de accidentes causados por factores humanos, es la posible implementación de una estrategia de coordinación que haga posible todas las mejoras comentadas anteriormente en la introducción.

Existen empresas como nuTonomy, compañía derivada del MIT, pioneras en el lanzamiento de tecnologías *software* para construir automóviles de conducción autónoma como un servicio. Sus clientes llaman a los coches a través de una aplicación móvil diciéndole dónde están y dónde quieren ir y un taxi autónomo, sin conductor, irá a recogerlos. En agosto de 2016, lanzó su servicio, llamado proyecto piloto (*pilot project*) en Singapur con el coche Mitsubishi i-MiEV aunque actualmente todavía se encuentra en fase de investigación y desarrollo.

Google tiene también un proyecto similar, antes llamado Google self-driving car project, actualmente conocido como Waymo, es una empresa desarrolladora de vehículos autónomos perteneciente a Alphabet Inc. La tecnología desarrollada por Waymo permite a un automóvil moverse autónomamente por ciudad y por carretera, detectando otros vehículos, señales de tráfico, peatones, etc.

El primer estado estadounidense que aprobó una ley que permite la conducción de vehículos sin conductor fue Nevada, entró en vigor en marzo de 2012 y la primera licencia que expidió el *Nevada Department of Motor Vehicles* fue en mayo de 2012 en un Toyota Prius modificado con la tecnología experimental *driverless* de Google.

Los coches robotizados de Google tienen cerca de 200.000 dólares en equipo. Cuenta con un telémetro en la parte superior, es un Velodyne de 64 rayos láser que permite generar un mapa 3D detallado de la zona. Combinando este mapa con otros mapas de alta resolución para combinarlos, produciendo diferentes tipos de modelos de datos que le permiten conducir solo.

CAPÍTULO 3
Análisis y requisitos

En este apartado vamos a hacer una descripción del análisis y los requisitos de la aplicación.

3.1 Análisis

3.1.1. Propósito

El propósito de la herramienta es poder simular todo tipo de situaciones que puedan encontrarse organizaciones de flotas de vehículos, para poder prever la situación y poder anticiparse a ella y así, planificarse con antelación.

Esto puede aplicarse no solamente al contexto de los taxis en el que nos hemos centrado, también es útil para sistemas de reparto de correo o reparto de comida a domicilio.

El propósito no es solamente ayudar a planificarse, también ayudar a que los tiempos de circulación sean lo más pequeños posible para reducir el uso de carburantes, y tener las ciudades más limpias.

3.1.2. Ámbito

La aplicación debe ofrecer un abanico de situaciones y estrategias que puedas combinar como desees para poder probar y hacer comparativas. Además si estas situaciones no se acoplan a lo que el usuario quiere, debe poder añadir los archivos, situaciones y estrategias que desee. Incluso si solamente quiere usar una parte de la aplicación porque ya tiene las otras, debe ser fácilmente adaptable.

3.1.3. Perspectiva del producto

Ésta herramienta está orientada para, en caso de necesitar modificaciones, ser usada o "puesta a punto"por un programador.

Esto se debe a que, en caso de querer probar otras combinaciones que no vienen en la aplicación, estas deben ser implementadas. Del mismo modo que si se quiere sustituir algún módulo, el nuevo debe acoplarse al sistema de comunicación ya creado para que se cumplan los estándares de entrada y salida.

Para facilitar esta tarea se escribirá un manual de usuario.

3.1.4. Funciones básicas del producto

Las funciones básicas que debe tener la herramienta son:

- Interfaz desde la que configurar la ejecución de la simulación.
- Poder ver en un mapa lo que está pasando durante la ejecución.
- Al finalizar, recoger los datos generados y mostrar dicha información.

3.1.5. Restricciones

El simulador debe ser multi-plataforma, es decir, debe poder ejecutarse desde los sistemas operativos Windows 8 o superior, iOS 10.0.1 o superior y Linux.

La interfaz debe ser clara y sencilla y se debe visualizar un mapa con los clientes donde aparecen y por donde están los taxis moviéndose para tener en todo momento una imagen de lo que está haciendo el simulador.

3.2 Requisitos

3.2.1. Requisitos de contenido

La aplicación contará con una sola vista en la que el foco de atención debe centrarse en el mapa donde se podrá hacer el seguimiento de la ejecución. Contará con un apartado de configuración y otro para mostrar los resultados de los tiempos de la ejecución. Deberá tener un botón para empezar la simulación y otro para cancelarla.

3.2.2. Requisitos de interfaz

Interfaz de usuario

La interfaz de usuario debe ser simple y sencilla, las posibilidades de configuración deben ser completas sin llegar a ser pesadas y la muestra final de los resultados no debe abrumar al usuario con muchos datos, solamente los relevantes.

Un posible diseño de la interfaz podría ser el siguiente:

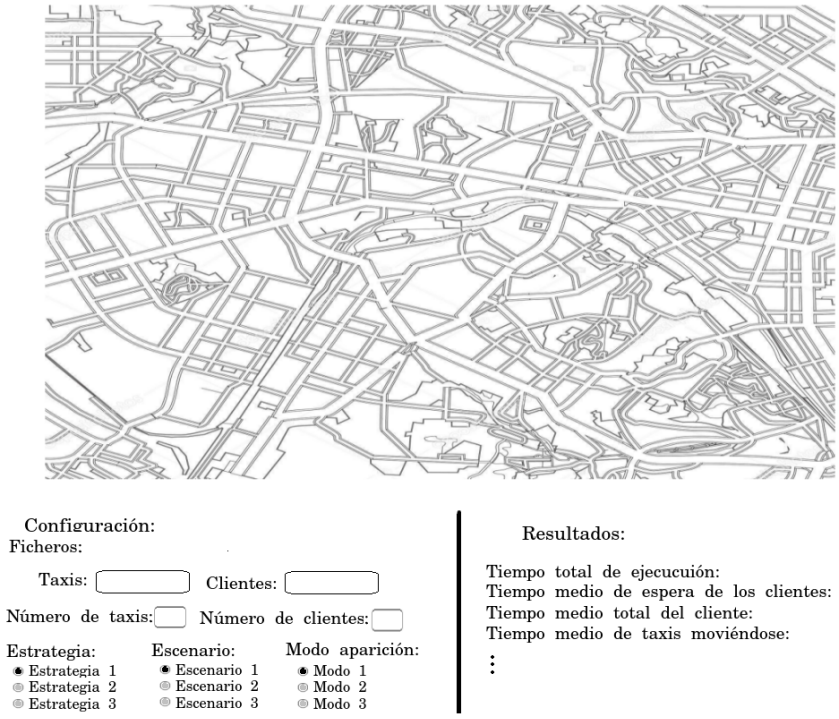

Figura 3.1: Boceto de una posible interfaz

Interfaz hardware

Cualquier usuario con un ordenador con acceso a internet debe poder ejecutarlo, independientemente del sistema operativo que utilice.

3.2.3. Requisitos funcionales:

- Hacer un simulador con vehículos como procesos concurrentes independientes.
- Hacer una interfaz de usuario desde la que puedas configurar la simulación con parámetros. Esos parámetros serán el número de clientes, de taxis, los ficheros de datos de ambos, la estrategia a utilizar y la configuración de apariciones de clientes.
- Montar la aplicación a base de módulos de modo que se dé facilidad a la hora de reemplazar componentes.
- Una vez terminada la ejecución de la simulación se debe obtener como resultado los datos con los tiempos de espera del cliente, el tiempo de trayecto, el tiempo medio de taxis parados y en movimiento de modo que se pueda hacer un análisis de estos resultados.
- Debe poder ejecutarse desde cualquier sistema operativo, Windows 8+, iOS o Linux.

CAPÍTULO 4
Tecnologías y herramientas

En este capítulo vamos a explicar varias tecnologías, tanto las que hemos usado como las que habríamos podido usar.

Para hacer una aplicación multi-agente existen diversas opciones, entre ellas:

- **NetLogo**: es un entorno de programación multi-agente diseñado por Uri Wilensky, fue una de nuestras principales opciones ya que, además de estar disponible de forma gratuita y de código abierto, integra una amplia variedad de contextos, entre ellos químicos, físicos, económicos, etc. Nos permite crear un entorno basado en un mapa bastante completo como el que necesitábamos, sin embargo, tiene un lenguaje propio (Logo) y haría nuestra aplicación menos flexible y se perdería la implementación por módulos que permite la combinación de estos módulos con otros o directamente cambiar algunos de ellos.[6]

- **SimPy**: es una librería de código abierto bajo la Licencia MIT basada en Python, es un marco de simulación de eventos basado en procesos por lo que puede utilizarse para la programación de sistemas multi-agente. Los procesos en SimPy son funciones del generador de Python y se utilizan para modelar componentes activos como clientes, vehículos o agentes. Además, a partir de la versión 3.1 proporcionará herramientas para monitorizar los procesos y así ayudar a la recolección de estadísticas de uso de recursos, tiempos, etc.[7]

- **Anylogic**: es una herramienta de modelado de simulaciones multi-método desarrollada por The AnyLogic Company con programación orientada a objetos en lenguajes como Java, C#, etc. Soporta metodologías basadas en agentes, simulaciones dinámicas y eventos. Incluye un lenguaje de modelado gráfico y permite la ampliación de los modelos de simulación con código Java, aunque es libre para aplicaciones educativas, el objetivo es usar código libre de modo que cualquiera pueda usarlo en el ámbito que quiera y no restringirlo solamente a un contexto educativo.[8]

- **Recursive Porous Agent Simulation Toolkit (Repast)**: es un conjunto de herramientas de simulación y modelado basado en agentes multiplataforma, libre y de código abierto. Repast tiene múltiples implementaciones en varios idiomas e incorpora características adaptativas, como algoritmos genéticos y regresión.[9]

Al final, nos decantamos por hacer una aplicación web con una arquitectura cliente-servidor ya que la potencia que ofrece el navegador es muy grande. Además, para seguir una estructuras por módulos que además sean adaptables, es más sencillo montar tu propio núcleo sin atarte a herramientas implementadas por terceros, de este modo tienes más libertad.

4.1 Tecnologías del cliente

Aqui presentamos una explicación de las tecnologías utilizadas en el cliente.

4.1.1. JavaScript

JavaScript (JS), lenguaje derivado de ECMAScript, es ligero e interpretado, orientado a objetos (OO), basado en prototipos, débilmente tipado e imperativo, pero también soporta estilos de programación funcional.

Aunque existe un JavaScript orientado a implementar el lado servidor, el server-side JavaScript, SSJS, se utiliza mayormente en el lado del cliente para la creación de páginas web dando funcionalidad a las vistas. También se usa en entornos sin navegador como Node.js, aplicaciones de escritorio, etc.

A pesar de que el diseño de la sintaxis es muy parecido al del lenguaje C, adopta nombres y concenviones de Java. Sin embargo Java y Javascript tienen semánticas y propósitos diferentes.

Actualmente el código JavaScript integrado en webs es interpretado en todos los navegadores modernos, se le pasa el Document Object Model (DOM) que es una interfaz estándar para representar documentos basados en marcas como HTML, a JavaScript para que pueda interactuar con él.

Antes, se utilizaba únicamente para realizar algunas operaciones solamente en en el lado del cliente sin acceso a funciones del servidor, actualmente se utiliza para comunicarse también con el servidor mediante tecnologías como AJAX (Asynchronous JavaScript And XML), es una tecnología que permite de forma asíncrona, en segundo plano, comunicarse con el servidor para poder obtener información o enviarla. Puede configurarse para que sea síncrona, sin embargo, en el momento en que se hiciera alguna petición al servidor la visualización de la interfaz se detendría hasta recibir la respuesta del servidor. Por lo general, se usan asíncronas, ya que mejora mucho el rendimiento de la aplicación y hace más amena su interacción.

En nuestro caso, se ha usado AJAX asíncrono para la comunicación entre el cliente y el servidor y la obtención de información sobre taxis, clientes y la ejecución en general.

4.1.2. Leaflet

Leaflet es una librería de mapas interactivos de código libre muy ligera hecha en JavaScript. Diseñada con simplicidad y con una documentación muy sencilla y práctica. Cuenta con una gran variedad de elementos e implementaciones que hacen que sea una librería realmente completa pero que, además cuenta con una colección de módulos que puedes usar para añadir más funcionalidad.

Figura 4.1: Logo de Leaflet

Decidimos usar esta librería para mostrar y manejar el mapa principalmente por la diversidad de posibilidades que añaden los módulos y por ser código libre.
Los marcadores puramente de Leaflet son fijos, no puedes ponerles movimiento, al estar tratando con automóviles necesitábamos que los marcadores pudiesen moverse por el mapa, para ello usamos el módulo (plugin) movingMarker [11], solamente se dedica a dar distintos tipos de movimiento según el que necesites y lanza eventos cuando un marcador aparece, empieza a moverse, se detiene, etc. La puedes encontrar en el apartado "Plugins"de Leaflet. [10]

4.1.3. HTML

HTML significa, en castellano, lenguaje de marcas de hiper texto, utilizado para la creación de páginas web. Es el elemento de construcción más básico de una página web. Todo el contenido de la web está representado en HTMLs pero no su funcionalidad

Hiper texto significa que hay enlaces que conectan una página web con otra, estos vínculos son fundamentales en la web. De esta forma puedes navegar de una web a otra.

Define una estructura básica y un código para la definición de contenido de una página web. Las marcas que usa denotan distintos tipos de elementos, texto, imágenes, vídeos, etc.

Es un estándar a cargo de la World Wide Web Consortium (W3C), una organización dedicada la estandarización de casi todas las tecnologías orientadas al desarrollo web. Se ha impuesto como lenguaje para la visualización de páginas web y todos lo navegadores actuales lo han adoptado.

4.1.4. CSS

Las Hojas de Estilo en Cascada o CSS, es un lenguaje utilizado para la definición de la apariencia de un documento en la pantalla, describe como debe ser renderizado cada elemento.

Usando una combinación de HTML y CSS, tenemos separada la definición del contenido de la definición de la apariencia de dicho contenido. Varios documentos pueden basar su estilo de la misma hoja CSS, esto ayuda a que en una web todas las vistas que haya mantengan una coherencia sin tener que especificar para cada elemento su estilo individual.

Este mecanismo funciona a base de reglas sobre el estilo de uno o más elementos. Las reglas tienen dos partes, el selector y la declaración. El selector dice qué elemento es el que debe seguir la declaración y la declaración dice como debe ser, por ejemplo: h1 color:red, esto es, todos los elementos de tipo h1 serán de color rojo.

Algo interesante de CSS es el significado de cascada, esto significa que, excepto en algunos casos, siempre tendrá más importancia una norma escrita más abajo que una escrita más arriba. Esto es muy útil en páginas web o aplicaciones grandes, ya que tendrán también hojas de estilos grandes, si en algún momento se quiere modificar el estilo de

algún elemento concreto, para no cambiar la configuración general del tipo de elemento, ya que esto afectaría a todos los objetos de ese tipo y no solamente al que nosotros queremos, se puede añadir abajo una nueva regla que haga referencia al elemento concreto, de este modo la última configuración sería la que prevalecería como principal y la anterior como secundaria.

4.2 Tecnologías del servidor

Estas son las tecnologías y herramientas que se han usado en el servidor.

4.2.1. Python

Python es un lenguaje de programación multiparadigma ya que soporta orientación a objetos, programación imperativa y funcional. Es un lenguaje interpretado con tipado dinámico y es multiplataforma.

El diseño de su sintaxis está hecho con el objetivo de que el código quede lo más legible posible haciendo obligatoria la tabulación del código.

Python es administrado por la Python Software Foundation y posee una licencia de código abierto.

4.2.2. Flask

Flask es un framework para implementar servidores web basado en la especificación WSGI de Werkzeug y el motor de Jinja2 con licencia BSD, permite hacer un servidor en apenas unas pocas lineas de código Python.

Se autodenomina micro-framefork porque no requiere herramientas o bibliotecas particulares y no tiene ninguna capa de abstracción de bases de datos, validación de formularios o cualquier otro componente en el que las bibliotecas de terceros preexistentes proporcionan funciones comunes. Sin embargo, existen extensiones para validación de formularios, varias tecnologías de autenticación y varias herramientas relacionadas con el marco común. [12]

Figura 4.2: Logo de Flask

En Python la creación de servidores suele hacerse con frameworks como Flask, no es que no se pueda hacer sin uno pero resulta más complejo, por eso decidimos usar Flask, hay otras opciones más completas, sin embargo, no necesitamos más de lo que este nos ofrece.

4.2.3. Google Maps API

Para escoger las rutas que deberán seguir los taxis, una posible opción es la API de Google.

Además de su versatilidad, es de las APIs que más información tienen del entorno que hay actualmente para el cálculo de rutas ya que tiene en cuenta muchos factores como por ejemplo, el tráfico.

Las Google Maps API son gratuitas para una gran variedad de casos de uso, aunque con límites:

- Máximo de 2500 solicitudes diarias, después cada 1000 solicitudes 0.50 dólares, hasta 100.000 diarias.
- Hasta 23 puntos de parada (waypoints) permitidos por cada solicitud.
- 50 solicitudes por segundo.

Para poder usarla debes obtener una clave, existen dos tipos, la estándar y la premium. También tiene tarifas planas si vas a hacer un uso exhaustivo de ella. [13]

Figura 4.3: Logo de Google Maps API

4.2.4. MapQuest

MapQuest es un programa de mapas que ofrece servicios de geolocalización. Tiene aplicación móvil desde la que puedes trazar tus rutas y compartirlas.

Consta también de varias librerías enfocadas todas al trazado de rutas y manejo de mapas interactivos. Para poder hacer uso de ellas debes registrarte y obtener una clave que debes usar al hacer las peticiones.

Su servicio gratuito te permite hacer 15.000 peticiones mensuales, en caso de necesitar más debes acogerte a un plan de pago.[15] [14]

Figura 4.4: Logo de MapQuest

4.2.5. OpenStreetMap (OSM)

OSM es un proyecto colaborativo para crear mapas libre y editables.

Los usuarios pueden subir rutas recogidas desde GPS y corregir si encuentran datos erróneos mediante herramientas creadas por la comunidad OpenStreetMap, cada vez más usuarios se unen a esta iniciativa de software libre colaborando para mejorarla.

Cuenta con aplicaciones de mapas en línea, cálculo de rutas y navegación y software cartográfico, todas ellas distribuidas bajo una licencia abierta.

Dada la orientación de código abierto que pretendemos darle a nuestra aplicación, usar esta librería habría sido la mejor opción, sin embargo, no conseguimos que nos dieran una clave para poder hacer peticiones de obtención de ruta mediante su servidor web. [16]

Figura 4.5: Logo de Open Street Maps

4.2.6. Docker

Docker es un software que automatiza el despliegue de aplicaciones y proporciona una capa de abstracción adicional.

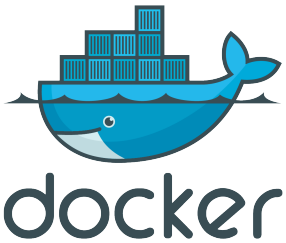

Figura 4.6: Objeto cliente

Dada la estructura de microservicios independientes, he creado un sistema de Dockerfiles para cada servicio, de este modo, si se van a hacer un número muy elevado de peticiones a un servicio puedes, de una manera sencilla, replicarlos y poner un intermediario que haga un balance de la carga y elija a cual de las replicas hacerle la petición.[17]

Dockerfile

Para poder arrancar los contenedores estos deben tener una imagen con un sistema operativo que proporcione a la aplicación los recursos necesarios para su despliegue, para ello se crea la imagen con un sistema operativo y durante la creación de la imagen puedes configurar muchas cosas como la exposición de puertos al exterior, variables de entorno, añadir archivos y ejecutar comandos.[18]

Docker-compose

Compose es una herramienta para gestionar sistemas multi-contenedor, es decir, si se van a utilizar distintos contenedores de docker para tener desplegados servicios que van a estar comunicándose entre si, mediante docker-compose puedes gestionar todos los contenedores desde un fichero docker-compose. A partir de ese fichero se crea una red de contenedores que puedes arrancar y parar todos a la vez o uno por uno. [19]

Una de las ventajas para sistemas de microservicios es que los contenedores dentro de una red de docker-compose se pueden ver entre ellos y llamarse, cosa que facilita la comunicación entre servicios.

En nuestro caso, dada la arquitectura por módulos que tiene el proyecto, nos era una facilidad poder gestionar todos los servicios a la vez con un solo comando en lugar de tener que ir uno por uno.

CAPÍTULO 5
Diseño de la aplicación

Una vez explicadas las tecnologías usadas en el apartado anterior, vamos a estructurar y explicar la implementación de la aplicación.

La estructura de la aplicación podría verse como la figura siguiente:

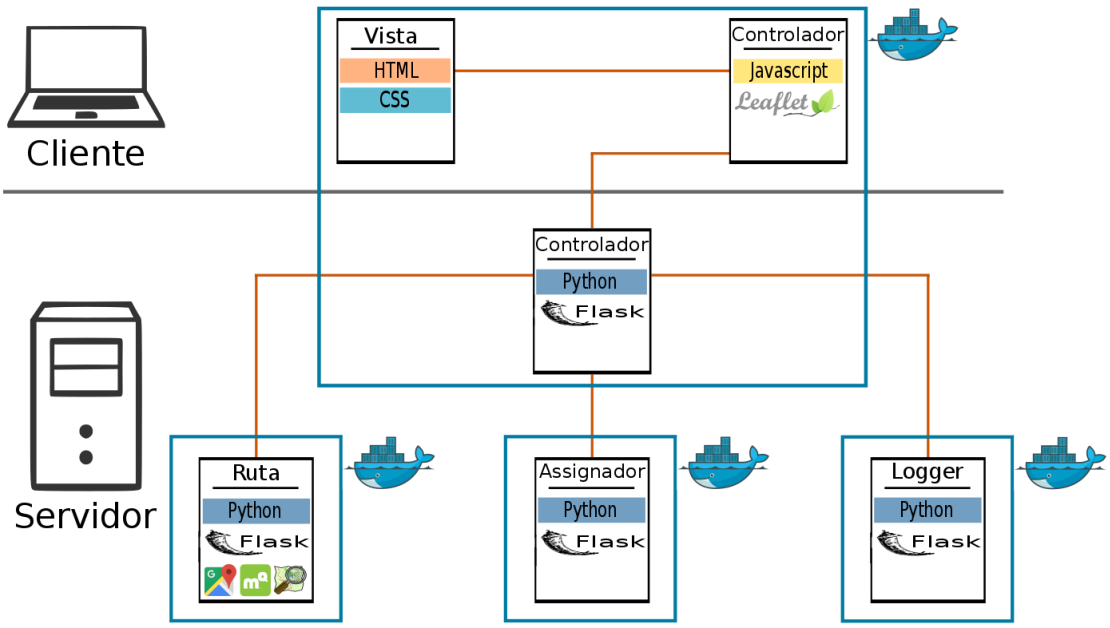

Figura 5.1: Estructura de la palicación

En la que cada uno de los modulos puede ser reemplazado o modificado independientemente del resto.

5.1 Cliente

El concepto de cliente se conoce comúnmente como aplicación informática que consume un servicio remoto, este servicio es ofrecido por un servidor.

En esta aplicación el cliente es la parte más visual, la que puede ver y manejar el usuario. Es decir, la vista y su controlador.

Para la creación de la vista se han usado plantillas HTML con hojas de estilo CSS para modificar la apariencia de la interfaz y, para la implementación de su funcionalidad hemos desarrollado un controlador usando JavaScript.

5.1.1. Estructura

Vamos a empezar a explicar la estructura del cliente con una imagen que representaría el cliente con sus tecnologías principales.

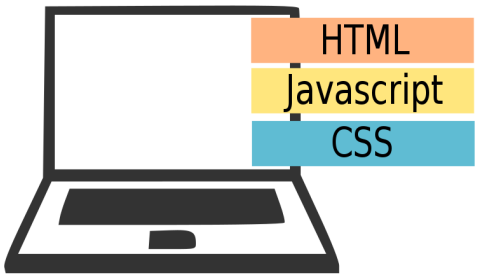

Figura 5.2: Imagen resumen de las tecnologías principales usadas en el cliente

Vista

Para la vista, se ha hecho una plantilla *HyperText Markup Language* (HTML) muy sencilla en la que en el centro se muestra un mapa. Este será el apartado principal de la vista, también hay un formulario en el que introducirás la configuración de la ejecución y unas tablas de texto en las que se mostrarán los resúmenes de los datos generados por el simulador cuando finalice el proceso. Estos datos serán los tiempo calculados que se han utilizado. La herramienta usada para crear el estilo o apariencia de la página web es *Cascading Style Sheets*(CSS)

Finalmente la vista de la aplicación es la siguiente:

Figura 5.3: Interfaz final del simulador

Controlador

La parte del controlador es la encargada del manejo de la vista y está implementada en JavaScript. Esta se comunicará con el servidor, tanto para avisarle de eventos o acciones llevadas a cabo por el usuario, como para recibir la información que deberá mostrar en la vista.

Su papel es de intermediario entre la vista del cliente y el servidor, de modo que aislemos roles para que cada uno esté enfocado a un objetivo y no se mezclen funcionalidades.

También es el encargado de manejar todos los marcadores que irán apareciendo en el mapa, tanto de taxis como de clientes.

5.2 Servidor

Un servidor es una aplicación en ejecución capaz de atender las peticiones de un cliente y devolverle respuestas en concordancia.

La parte del servidor o backend está implementada en Python, nos decantamos por este lenguaje además de porque lo conocemos bastante, por la gran comunidad y apoyo que tiene detrás.

Aquí están implementados varios servicios, el controlador, donde esta toda la lógica y la gestión de los agentes, el enrutador, que se dedica a encontrar la mejor ruta que deben tomar los taxis, el asignador, decide cual es el mejor taxi para ir a por cada cliente y el analizador de los datos generados por el servidor, que serán los que se le devolverán al usuario al finalizar la ejecución.

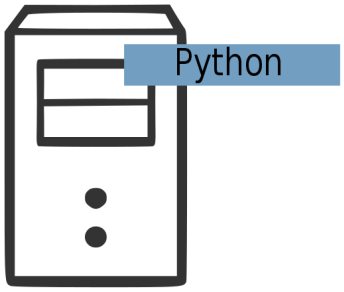

Figura 5.4: Imagen resumen de las tecnologías principales usadas en el servidor

Todos los módulos del servidor están implementados en Python usando el framework para implementación de servidores Flask.

5.2.1. Estructura

Controlador

Esta parte estará escuchando y atendiendo todas las peticiones que se hagan desde el cliente. Manejará todos los ficheros de datos y tendrá una comunicación constante con el cliente.

Su función principal es ser el núcleo de todos los elementos que participan en la ejecución de la simulación. Se encargará de obtener en cada caso qué taxi deberá ir a por cada cliente y obtener la ruta que debe seguir el taxi para ir a por el cliente y luego llevarle a su destino y, una vez haya comprobado que la simulación ha terminado hacer un procesado de datos para mostrarlo al usuario. Cada una de estas funciones están implementadas independientemente, el controlador solamente se encarga de llamarlas.

Si durante la ejecución aparece un cliente y no hay ningún taxi libre para ir a por él, el controlador almacena al cliente en una cola de tipo FIFO, esto son siglas que en inglés significa *First In First Out*, esto es, que cuando un cliente entre en la cola tendrá preferencia para salir antes que todos los que entren detrás de él. Mientras esté en la cola irán terminando taxis de llevar a otros clientes e irán atendiendo a los d ela cola por orden de llegada.

Estrategias

En este módulo es donde están implementadas todas las estrategias de coordinación de vehículos. Se encargará de, dado un cliente encontrar qué taxi de los que hay libres es mejor para él, según la estrategia seleccionada, seguirá unos criterios u otros de elección.

Este servicio será llamado desde el controlador cuando se tenga que asignar algún taxi a algún cliente. Este hará el cálculo y una vez obtenido se lo devolverá al controlador.

Rutas

Este módulo se encarga de hacer el cálculo de la ruta entre un origen y un destino. Una vez se tiene el par cliente y taxi, se deberá consultar por donde debe ir ese taxi hasta el cliente y luego por donde ir hasta el destino que el cliente tenga.

Para esta función hemos utilizado los servicios web descritos anteriormente, la API de Google Directions y la API MapQuest Directions, estos servicios devuelven información sobre la ruta y nosotros seleccionamos la que nos interesa para devolvérsela al controlador.

Principalmente hemos usado MapQuest ya que a pesar de que la cantidad de peticiones mensuales es menor, podías ejecutar de una sola vez.

Procesador del resultado

Este servicio se encarga de hacer un procesado de la información que irá emitiendo el controlador sobre los eventos que van sucediendo durante la ejecución.

Durante la ejecución el controlador irá almacenando en un archivo información sobre lo que va pasando, y en qué momento pasa cada evento (este fichero, o esta información, la llamaremos log a partir de este momento).

Una vez finalizada la simulación, el controlador llamará al procesador del resultado para que recoja y analice el log, sacando de este todos los tiempos y datos necesarios y se los devolverá al controlador.

5.3 Estructuras de datos

Para los datos de los clientes y los taxis, hemos creado ficheros con clientes y con taxis que siguen la siguiente estructura.

5.3.1. Clientes

El fichero que contiene los clientes tiene la siguiente estructura:

```
0,39.47188,-0.37688,39.46429,-0.36684
1,39.47172,-0.37036,39.47543,-0.37611
...
```

Cada línea representa un cliente, el primer número es el identificador del cliente, los dos siguientes números serán la latitud y la longitud de su origen, las dos siguientes de su destino.

Estos datos serán recogidos de un fichero de texto y se mapeará en un diccionario o mapa, estructura de datos en la cual puedes acceder al valor mediante una clave. Para poder manejar los datos, los convertimos en objetos JSON y así poderlos pasar entre los distintos servicios y el cliente.

Aquí vemos el diccionario clientes con el tipo de valor que tiene cada campo:

```
Clientes { 0: {
             origin:      Tupla
             destination: Tupla
             ready:       Booleano
             going:       Booleano
             done:        Booleano
           }
         ...
}
```

Figura 5.5: Estructura del objeto cliente

- **Identificador (id)**: el identificador nos ayudará a poder manejar a los clientes accediendo a ellos en base a su identificador y a tener un orden en las estructuras.
- **Origen (origin)**: aquí se almacenará el origen del cliente, donde aparecerá, en una tupla con dos valores: (latitud, longitud).
- **Destino (destination)**: será una tupla de la misma forma que el origen pero este será el destino, dónde quiere ir el cliente.
- **Listo (ready)**: el tipo de este atributo es un booleano, tomará valor de verdadero o falso en base a si el cliente ha aparecido ya, y está esperando al taxi. Este valor será inicializado a falso.
- **En camino (going)**: este en cambio, tomará valor según si el cliente está ya en un taxi de camino a su destino. Este también se inicializará a falso.

- **Hecho (done)**: este valor se pondrá a verdadero cuando el cliente haya desaparecido, es decir, haya llegado a su destino. Al igual que los dos booleanos anteriores, también se inicia como falso.

Por tanto, la forma del diccionario es la siguiente: se llamará *clientes*, las claves serán identificadores por lo que para acceder a los datos del cliente con identificador 0 se haría así: *clientes[0]* esto nos devolverá a su vez otro diccionario en el que estarán mapeados los datos del cliente, de modo que para acceder, por ejemplo al origen del cliente 0, lo haremos así: *clientes[0]['origen']*

5.3.2. Taxis

Los taxis también estarán almacenados en un fichero que tendrá una forma asi:

```
0,39.47778,-0.37718
1,39.47552,-0.3798
...
```

Al igual que lo clientes, cada línea representa un taxi, el primer valor será su identificador y los dos siguientes serán las coordenadas de su posición inicial.

Estos datos serán procesados separando los valores mediante las comas (,) y mapeados del mismo modo que los de los clientes.
La estructura de datos para los taxis será también un mapa aunque algo más sencillo que el de los clientes:

```
Taxis {   0:  {
                  position:     Tupla
                  assigned:     Booleano
                  taken:        Booleano
              }
          ...
}
```

Figura 5.6: Estructura del objeto taxis

- **Identificador (id)**: al igual que antes, es el identificador del taxi, lo que nos permitirá acceder a ellos y manejarlos.
- **Posición (position)**: aquí estará la posición actual del taxi, tanto si está parado como si está en movimiento, su tipo será una tupla de coordenadas (latitud, longitud).
- **Asignado (assigned)**: este booleano se pondrá a verdadero cuando al taxi se le haya asignado algún cliente a por el que ir. Este valor será asignado inicialmente a falso.
- **Cogido (taken)**: este valor, también booleano, nos dirá si el taxi está ya llevando a algún cliente, de modo que, mientras un taxi aún no tenga este valor a verdadero puede ser que se le asigne otro cliente aunque ya tenga asignado uno. Al igual que el anterior, este valor será inicializado a falso.

Su funcionamiento es igual que el de los clientes.

Para hacer los documentos de datos se han implementado algunos scripts para ayudarnos a crear ficheros de datos con una estructura concreta y que el servidor la procese de forma sencilla para crear los objetos.

CAPÍTULO 6
Preparación de las pruebas

En este capítulo hablaremos de las estrategias que hemos desarrollado y cómo hemos organizado, preparado y ejecutado las pruebas. Aprovecharemos para dar nombre a las estrategias, escenarios y modos de aparición para que en el apartado del análisis podamos usarlos y simplificar la explicación.

6.1 Estrategias

Estas son las estrategias que hemos implementado.

6.1.1. Estrategia 1: El primero libre

La estrategia el primero libre consiste simplemente en escoger el primer taxi que haya libre para el primer cliente que hay en la cola.

Lo que se hará en este caso es recorrer el diccionario de taxis de 0 a la cantidad de taxis que haya y el primero que encuentre libre será el que asigne.

Dado que no tendrá ningún tipo de criterio para agilizar los trayecto, puede que a *priori* ya no parezca muy buena estrategia. Lo más probable es que tienda a sobrecargar más los primeros taxis

6.1.2. Estrategia 2: Aleatorio

En este caso, en lugar de hacer un recorrido del diccionario de taxis, probará números al azar entre 0 y el número de taxis que haya y cuando encuentre uno libre parará de buscar y mandará ese taxi.

Esta estrategia es parecida a la anterior pero, en cambio, al ser una elección aleatoria es más improbable que se acumulen más llamadas en ciertos taxis.

6.1.3. Estrategia 3: El más cercano al cliente

Como dice el nombre, esta estrategia recoge todos los taxis libres, hace el cálculo de las rutas hasta el cliente y escoge el que tenga la ruta más corta.

El principal objetivo de esta estrategia es optimizar el tiempo de espera de los clientes y por ende, el tiempo total de ese cliente.

De esta estrategia se han hecho dos versiones:

- **Distancia real**: la distancia real será la que nos devolverá como resultado de la ruta elegida.
- **Distancia euclídea**: aquí se hará un cálculo de la distancia entre los puntos y se elegirá el menor.

6.1.4. Estrategia 4: El más cercano al cliente con corrección

Esta técnica es la misma que la anterior solo que mientras el cliente no haya subido al taxi ese taxi podrá ser reasignado si el algoritmo considera más óptimo asignarle otro cliente y asignar su anterior cliente a otro taxi.

Esta estrategia no hemos podido probarla para compararla con el resto de pruebas ya que, el hecho de poder corregir la ruta aún cuando ya se está llevando a cabo es muy costoso a nivel de cantidad de peticiones de obtención de rutas que, dadas las limitaciones, no podíamos ejecutar para compararla con el resto de estrategias.

6.2 Escenarios

Los escenarios serán la distribución de clientes que irán apareciendo por el mapa. Los puntos de aparición los hemos limitado a la zona de la ciudad de Valencia. Hemos probado estas tres distribuciones:

6.2.1. Escenario 1: Aleatorio

Este podría ser un escenario normal en el que los clientes van apareciendo por todo el mapa de forma aleatoria.
Un posible ejemplo de esta distribución podría ser el siguiente:

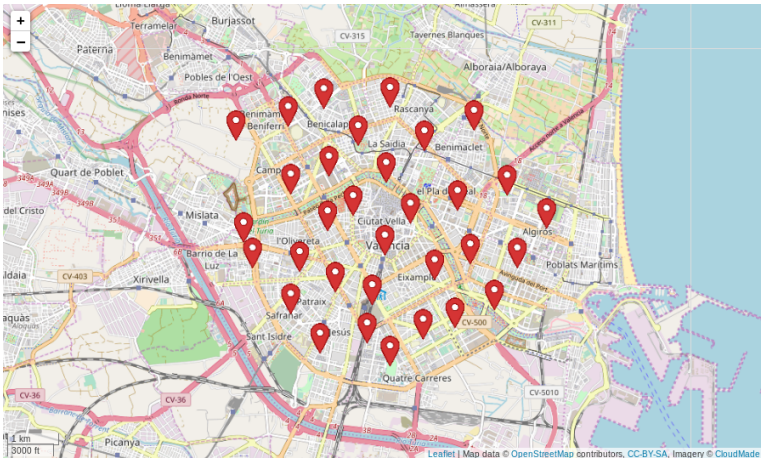

Figura 6.1: Imagen del escenario aleatorio

6.2.2. Escenario 2: Concentración

Este escenario presenta la típica situación en la que muchas personas aparecen en un mismo lugar a causa de algún evento, como por ejemplo un concierto. Concretamente hemos usado un grupo de datos que acumula un 80 % de los clientes en una zona y el otro 20 % por el resto del mapa.
Esta es una posible imagen de este escenario:

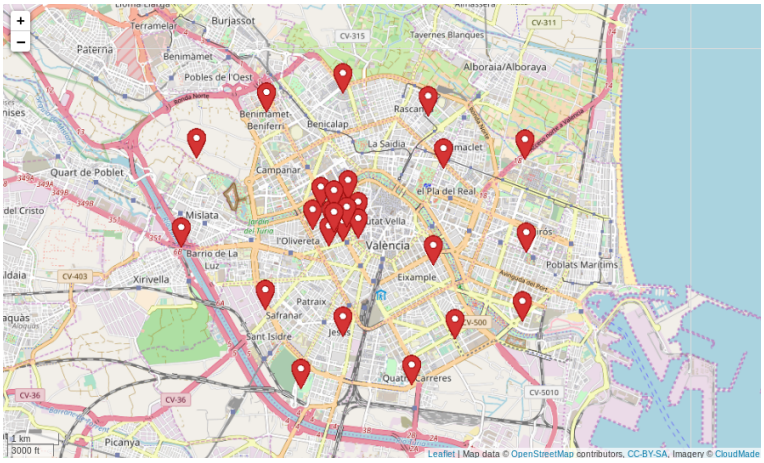

Figura 6.2: Imagen del escenario concentración

6.2.3. Escenario 3: Masivo

La cantidad de clientes aquí se duplica debido también a algún evento especial de más envergadura que el anterior como pueden ser unas fiestas locales. Por ejemplo, en el caso de Valencia, podrían ser las fallas.
Un posible ejemplo de esta distribución podría ser el siguiente:

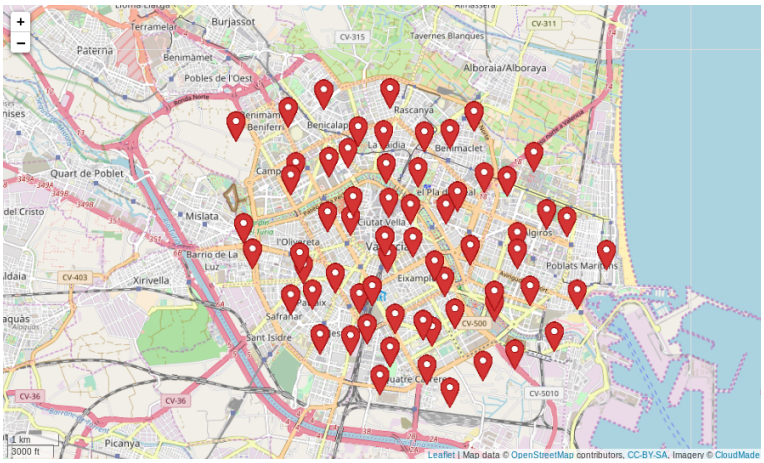

Figura 6.3: Imagen del escenario masivo

6.3 Modos de aparición

Los modos de aparición serán las distribuciones de los clientes pero en el tiempo, es decir, cada cuanto tiempo aparecerá un cliente.

6.3.1. Modo 1: Todos a la vez

En este caso, como bien dice, aparecerán todos a la vez. Puede que esta situación sea un poco extraña que se produzca pero no está de más dado que el escenario 2 (Concentración), si se tratase de la finalización de algún evento, podría ser una combinación perfectamente válida que muchos clientes aparezcan en un mismo lugar a una misma hora.

6.3.2. Modo 2: Constante

En el modo de aparición 2, los clientes irán apareciendo de forma constante, es decir, cada X tiempo aparecerá un cliente, de modo que el tiempo entre la aparición de un cliente y el siguiente será siempre el mismo.

6.3.3. Modo 3: Campana de Gauss

La distribución de una campana de Gauss acumularía los clientes en el centro mientras que irían distanciándose en el tiempo, vamos a poner una imagen para dejar claro este modo. De este modo al empezar los clientes aparecerían más dispersos para ir aumentando la densidad de clientes por unidad de tiempo hasta llegar a la mitad, entonces empieza a bajar del mismo modo que ha subido.

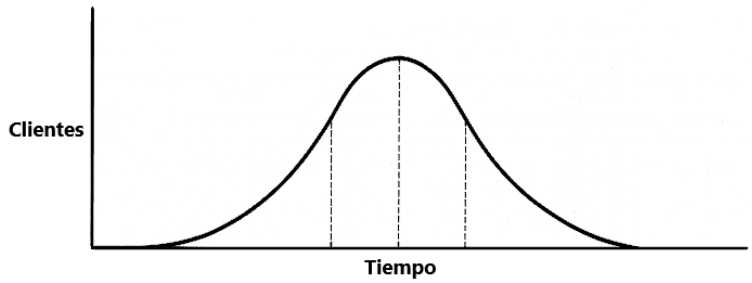

Figura 6.4: Campana de Gauss

6.4 Organización y ejecución de los experimentos

Para la realización de los experimentos hemos usado un set de 167 taxis. Para los clientes se ha hecho un set de 1000 clientes en los que se usarán 500 para los escenarios aleatorio y concentración. Para el masivo se usará el set entero.

Los experimentos los hemos organizado combinando los tres factores que hemos explicado antes, escenario, modo de aparición y estrategia. Es decir, hemos hecho combinaciones de todos los escenarios con los modos de aparición y lo hemos ejecutado con las tres estrategias.
Vamos a enumerar los experimentos para hacer más sencillo el análisis de los resultados:

- **Experimento 1:** este experimento será la combinación del escenario aleatorio, modo de aparición todos a la vez y la estrategia el primero libre.
- **Experimento 2:** en este caso, será como el de arriba cambiando la estrategia, que será la aleatoria.
- **Experimento 3:** e igual en este caso pero con la estrategia el más cercano.
- **Experimento 4:** para este experimento la combinación será también el escenario aleatorio pero con el modo de aparición constante. La estrategia de coordinación será el primero libre.
- **Experimento 5:** aquí la combinación de escenario y modo de aparición será el mismo pero, ejecutado esta vez con la estrategia aleatorio.
- **Experimento 6:** en este experimento será igual que los dos anteriores con la estrategia el más cercano.

- **Experimento 7:** el experimento 7 sigue con el mismo escenario, pero con el modo de aparición campana de Gauss y la estrategia el primero libre.
- **Experimento 8:** el 8 será igual que el anterior pero con la estrategia aleatorio.
- **Experimento 9:** aquí seguiremos con las 2 combinaciones anteriores pero con la estrategia el más cercano.
- **Experimento 10:** en este experimento ya no seguimos con el mismo escenario, ahora pasamos al escenario concentración, con el modo de aparición todos a la vez y la estrategia el primero libre.
- **Experimento 11:** en este seguiremos con la misma combinación de escenario y modo de aparición pero con la estrategia aleatorio.
- **Experimento 12:** seguimos igual que antes pero con la estrategia el más cercano.
- **Experimento 13:** aquí seguiremos en el escenario concentración pero el modo de aparición será constante y la estrategia el primero libre.
- **Experimento 14:** al igual que antes pero usando la estrategia aleatorio.
- **Experimento 15:** y en este seguiremos con el mismo escenario y modo de aparición pero con la estrategia el más cercano.
- **Experimento 16:** continuamos con el escenario de concentración pero cambiamos al modo de aparición de campana de Gauss. Lo ejecutaremos primero con la estrategia el primero libre
- **Experimento 17:** siguiendo el mismo contexto anterior, probaremos la estrategia aleatorio.
- **Experimento 18:** y seguiremos con la estrategia el más cercano en las mismas condiciones que el anterior.
- **Experimento 19:** en este experimento vamos a cambiar de escenario, pasaremos al escenario masivo, lo probaremos con el modo de aparición todos a la vez y con la estrategia el primero libre.
- **Experimento 20:** en este caso seguiremos con la combinación de escenario masivo y modo de aparición todos a la vez y usaremos la estrategia aleatorio.
- **Experimento 21:** al igual que el caso anterior pero con la estrategia el más cercano.
- **Experimento 22:** para el experimento 22 hemos combinado el escenario masivo con el modo de aparición constante y la estrategia el primero libre.
- **Experimento 23:** al igual que el anterior pero con la estrategia aleatorio.
- **Experimento 24:** para este la estrategia usada ha sido el más cercano.
- **Experimento 25:** en este caso se ha combinado el escenario masivo con el modo de aparición campana de Gauss y la estrategia el primero libre.
- **Experimento 26:** en este experimento se ha usado el contexto anterior pero con la estrategia aleatorio.
- **Experimento 27:** e igual que el anterior pero con la estrategia el más cercano.

La ejecución de las pruebas ha sido complicada, esto se debe a la limitación que tenemos a la hora de hacer peticiones a las APIs de rutas, en muchos casos, si coincidían peticiones en el mismo segundo algunas de ellas devolvían error por la limitaciones de peticiones por segundo, cuando esto pasaba, se tenía que detener la ejecución y volverla a empezar. No solo han sido complicadas por esta razón, si usábamos la API de Google limitaba las peticiones diarias por lo que teníamos un límite de pruebas al día.

Esto podría evitarse pagando una tarifa y teniendo muchas más peticiones disponibles y nos quitaría muchas limitaciones.

A pesar de todo, hemos podido realizar todas las pruebas y procederemos a analizar los resultados obtenidos en el siguiente capítulo.

CAPÍTULO 7
Análisis de los resultados

En este capítulo vamos a hacer un análisis de los resultados obtenidos en los experimentos planteados anteriormente. Dado que las pruebas tenían duraciones distintas muchas veces a causa de su modo de aparición, por ejemplo el modo todos a la vez siempre finalizaba antes que el modo constante, hemos recogido las medias de todos los tiempos y las hemos pasado a porcentajes para poder hacer comparaciones entre ellas más reales.

Para el cálculo de tiempos de los clientes, dado que el tiempo total que es la suma del tiempo de espera y del tiempo de trayecto, se ha calculado qué porcentaje de tiempo se ha dedicado a cada uno.

Por ejemplo, si un cliente ha estado activo durante 10 minutos, de los cuales 4 han sido de espera y 6 de trayecto, los porcentajes serían, un 40 % del tiempo invertido por el cliente estaría dedicado a la espera y el 60 % al trayecto.

Para el de los taxis, se ha calculado cuanto tiempo ha estado cada taxi parado y cuanto en movimiento a lo largo de la simulación, por lo que el porcentaje se ha calculado en base a la duración de la simulación.

Por ejemplo, si la duración de la simulación ha sido de 20 minutos y el taxis ha estado 15 en movimiento y 5 parados, es que ha estado el 75 % del tiempo en movimiento y el 25 % parado.

7.1 Resultados obtenidos

En esta sección vamos a exponer los resultados obtenidos y a hacer un análisis de los mismos.

Vamos a dividir la sección de experimentos realizados agrupados por combinaciones de escenario y modo de aparición, de este modo, se hará una comparación entre los tiempos obtenidos en un mismo contexto con las diferentes estrategias.

Como hemos comentado antes, la estrategia el más cercano se ha usado la versión de la distancia euclídea para poder ahorrar peticiones al servidor y así agilizar las pruebas, aún así, la aproximación entre la versión real y esta es muy buena.

7.1.1. Escenario Aleatorio y Modo de aparición Todos a la vez

En esta prueba hemos combinado el Escenario 1, aleatorio, con el Modo de aparición 1, todos a la vez y vamos a ejecutarlo con las tres estrategias implementadas.

En estas pruebas se estarían llevando a cabo los experimentos 1, 2 y 3.

Estrategia El primero libre

Estos resultados serían los del experimento 1, una combinación del escenario 1, con el modo de aparición 1 y la estrategia El primero libre.

Tiempo total de la simulación:	6.4 mins
Tiempo medio de taxis en movimiento:	76 %
Tiempo medio de taxis parados:	24 %
Tiempo medio de clientes en espera:	96 %
Tiempo medio de clientes en trayecto:	4 %
Tiempo medio de clientes activos:	2.6 mins

Como primera prueba ejecutada, lo que se puede ver es que los taxis elegidos más veces son los que tienen un identificador más bajo, esto es debido al tipo de estrategia que se ha usado que, como hemos dicho antes, era una situación un poco previsible.

Estrategia Aleatorio

Aquí están los resultados obtenidos del experimento 2, con la combinación escenario aleatorio, modo de aparición todos a la vez y con la estrategia aleatorio:

Tiempo total de la simulación:	5.5 mins
Tiempo medio de taxis en movimiento:	77 %
Tiempo medio de taxis parados:	23 %
Tiempo medio de clientes en espera:	95 %
Tiempo medio de clientes en trayecto:	5 %
Tiempo medio de clientes activos:	2.25 mins

Los resultados de esta prueba no son muy diferentes a los de la anterior pero dado que en ninguno de los dos se intenta optimizar nada, no es de extrañar que se parezcan tanto.

Una diferencia clara con el anterior es la distribución de los clientes, en el experimento anterior se juntaban muchos clientes para los primeros taxis y pocos o ninguno para los últimos, al contrario que antes, en este se han distribuido los clientes más uniformemente entre los taxis.

Estrategia El más cercano

En este experimento, el 3, se ha realizado la misma combinación de escenario y modo de aparición, usando esta vez la estrategia el más cercano.

Tiempo total de la simulación:	5.6 mins
Tiempo medio de taxis en movimiento:	79 %
Tiempo medio de taxis parados:	21 %
Tiempo medio de clientes en espera:	98 %
Tiempo medio de clientes en trayecto:	2 %
Tiempo medio de clientes activos:	2.25 mins

En esta estrategia se intenta optimizar el tiempo de espera del cliente para que sea lo menor posible aunque, en este caso el modo de aparición afecta mucho, al aparecer todos los clientes a la vez prácticamente todos los taxis estarán ocupados el 100 % del tiempo. Esto significa que habrá muchos clientes en cola, la cual se atiende en orden de llegada y se irá asignando el primer taxi que quede libre que sea o no el más óptimo. Sin embargo, si el tipo de cola no fuese *First In First Out (FIFO)*, es decir, el primero que entra sale, y recorriera la cola en busca del cliente más cercano a él es posible que mejorase. Por otro lado, si apareciese un cliente algo más alejado posiblemente su tiempo de espera aumentaría mucho y tampoco queremos sobrecargar el tiempo de espera de los clientes más aislados.

Resumen de los datos de los tres experimentos:

	Estrategia 1	**Estrategia 2**	**Estrategia 3**
Tiempo total de la simulación	6.4min	5.5min	5.6min
Tiempo medio de taxis en movimiento	76 %	77 %	79 %
Tiempo medio de taxis parados	24 %	23 %	21 %
Tiempo medio de clientes en espera	96 %	95 %	98 %
Tiempo medio de clientes en trayecto	4 %	5 %	2 %

Tabla 7.1: Tabla comparativa de los resultados obtenidos de los experimentos 1, 2 y 3

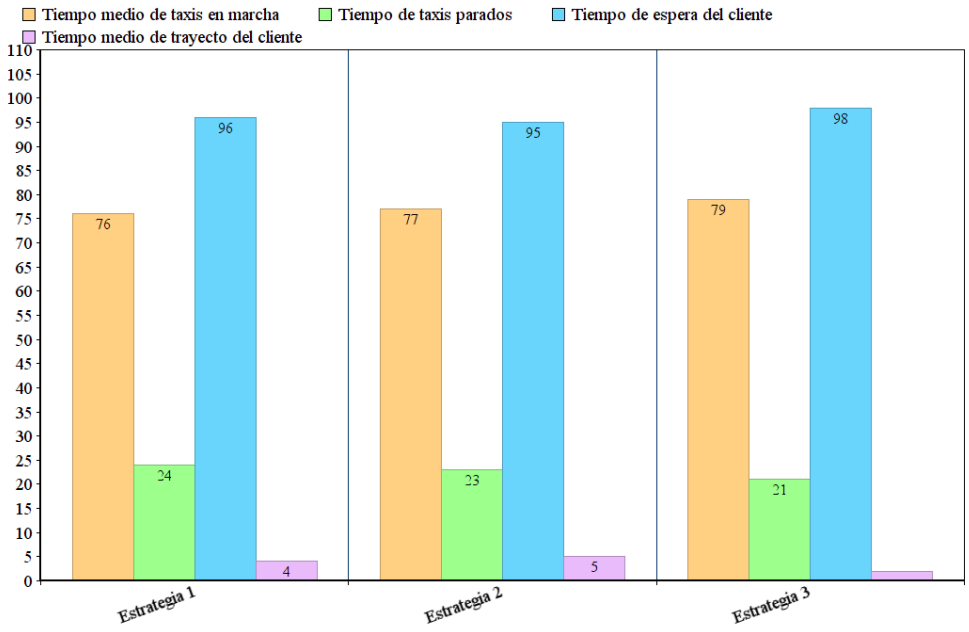

Figura 7.1: Representación gráfica de los resultados de los experimentos 1, 2 y 3

En esta combinación no se viene a notar mucho la diferencia de tiempo entre unas estrategias u otras. Esto se debe a que al aparecer todos los clientes a la vez, todos los taxis estarán ocupados enseguida por lo que llega un punto en el que hay tanto cliente esperando en la cola que el método de asignación será igual, el primer taxi libre recogerá al cliente que más tiempo lleve esperando.

7.1.2. Escenario Aleatorio y Modo de aparición Constante

En esta sección se van a analizar los experimentos 4, 5 y 6. Estos serán el escenario aleatorio con el modo de aparición constante y las estrategias el primero libre, aleatorio y el más cercano respectivamente.

Estrategia El primero libre

Esta prueba sería el experimento 4 explicado antes, los resultados de esta prueba son usando la estrategia el primero libre.

Tiempo total de la simulación:	9.6 mins
Tiempo medio de taxis en movimiento:	10 %
Tiempo medio de taxis parados:	90 %
Tiempo medio de clientes en espera:	54 %
Tiempo medio de clientes en trayecto:	46 %
Tiempo medio de clientes activos:	0.5 mins

En este modo los números son muy distintos al anterior. Esto es debido a que el tiempo entre cliente y cliente es mayor, esto hace que no esté el 100 % de los taxis todo el tiempo ocupados. Con la estrategia el primero libre, junta el 100 % de los cliente en el solamente el 10 % de los taxis (los de menor identificador), como consecuencia de esto tenemos un tiempo de simulación total de casi el doble y un porcentaje de tiempo sin conducción altísimo, debido a que el 90 % de los taxis no han tenido ningún cliente asignado.

El tiempo de espera de los clientes es menor, esto es debido a que, al aparecer periódicamente y no todos a la vez, los taxis tienen tiempo de ir a por ellos prácticamente nada más aparecen.

Estrategia Aleatorio

En este experimento, el 5, es la combinación del escenario aleatorio, el modo constante y la estrategia aleatorio.

Tiempo total de la simulación:	9.5 mins
Tiempo medio de taxis en movimiento:	10 %
Tiempo medio de taxis parados:	90 %
Tiempo medio de clientes en espera:	54 %
Tiempo medio de clientes en trayecto:	46 %
Tiempo medio de clientes activos:	0.28 mins

Al comparar esta prueba con la anterior vemos que apenas hay diferencia entre los resultados obtenidos. Esto es normal debido a que la asignación de taxis no sigue ningún criterio de optimización. La única diferencia que vamos a apreciar siempre entre esta estrategia y la anterior, será el reparto de clientes entre los taxis, que en esa estrategia será más equitativo.

Estrategia El más cercano

Aquí tenemos los resultados obtenidos del experimento 6, el escenario aleatorio, modo de aparición constante y la estrategia le más cercano.

Tiempo medio de taxis en movimiento: 6 %
Tiempo medio de taxis parados: 94 %
Tiempo medio de clientes en espera: 31 %
Tiempo medio de clientes en trayecto: 69 %
Tiempo medio de clientes activos: 0.20 mins
Tiempo total de la simulación: 9.3 mins

En estos tiempos si que se puede apreciar un descenso del tiempo de espera del cliente, esto se debe a que, en este caso, se está escogiendo al taxi más cercano a él. Al ser el modo de aparición constante, esto hace que la estrategia tenga más taxis donde elegir ya que rara vez estarán todos ocupados, por lo que si que se podrá elegir la mejor opción

El porcentaje tan alto de tiempo de taxis parados se debe a que para este escenario había demasiados taxis, es decir, un número menor de taxis habría sido suficiente para cubrir la demanda sin aumentar el tiempo de espera de los clientes.

Resumen de los datos de los tres experimentos:

	Estrategia 1	**Estrategia 2**	**Estrategia 3**
Tiempo total de la simulación	9.6min	9.5min	9.3min
Tiempo medio de taxis en movimiento	10 %	10 %	6 %
Tiempo medio de taxis parados	90 %	90 %	94 %
Tiempo medio de clientes en espera	54 %	54 %	31 %
Tiempo medio de clientes en trayecto	46 %	46 %	69 %

Tabla 7.2: Tabla comparativa de los resultados obtenidos de los experimentos 4, 5 y 6

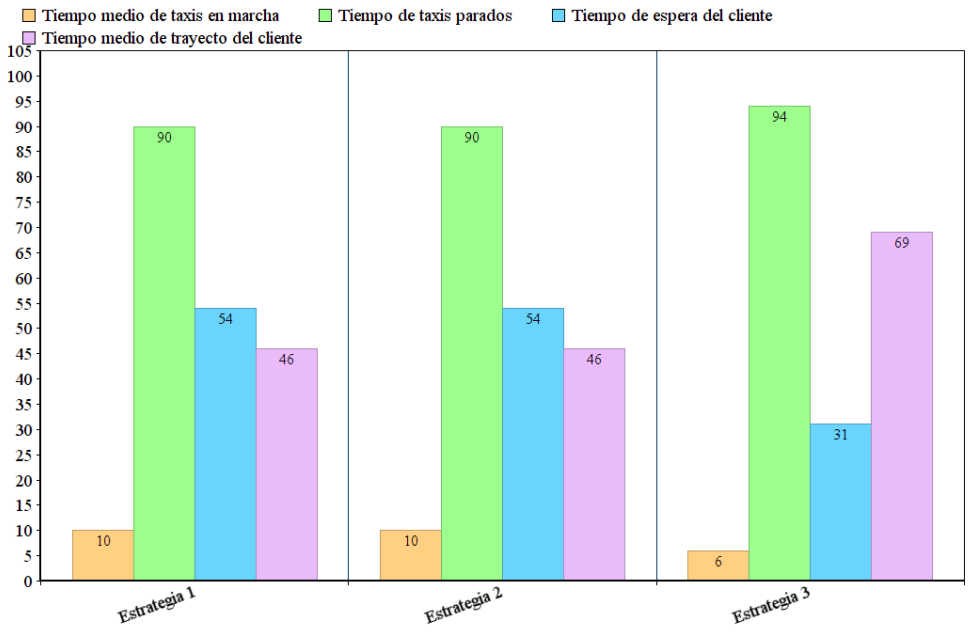

Figura 7.2: Representación gráfica de los resultados de los experimentos 4, 5 y 6

En las gráficas se puede apreciar fácilmente la diferencia de porcentaje de tiempo invertido por los clientes en esperar a que los recojan con una diferencia del 23 % menos.

con respecto al tiempo de trayecto de los taxis, es por lo general bastante menor debido a que van muy holgados, aún así se aprecia una reducción del 4 % del tiempo de trayecto en la estrategia 3, por lo que esta estrategia sería la más óptima en este contexto. Tanto para mejorar el servicio de cara a los clientes como para reducir el uso de carburantes.

7.1.3. Escenario Aleatorio y Modo de aparición Campana de Gauss

En este set de pruebas están los experimentos 7, 8, 9. La combinación de factores en este caso será, escenario aleatorio y modo de aparición campana de Gauss. y aquí están los resultados con las distintas estrategias.

Estrategia El primero libre

La combinación con esta estrategia es el experimento 7, en el que se han obtenido los siguientes resultados.

Tiempo total de la simulación:	13 mins
Tiempo medio de taxis en movimiento:	63 %
Tiempo medio de taxis parados:	37 %
Tiempo medio de clientes en espera:	70 %
Tiempo medio de clientes en trayecto:	30 %
Tiempo medio de clientes activos:	0.41 mins

En esta prueba volvemos a ver aumentar el tiempo de conducción, aunque no llega a ser tan alto como en el modo de aparición 1. Esto se debe a que en este modo de aparición empiezan apareciendo muy despacio, va aumentando la velocidad de aparición casi hasta que aparecen a la vez y luego vuelve a haber un aumento del tiempo de aparición entre clientes.

En este caso, vemos aumentar el tiempo de espera de los clientes con respecto al modo de aparición constante pero al contrario con el modo de aparición todos a la vez.

Esto se debe a que mientras los clientes aparecen poco a poco los taxis tienen tiempo de ir a por ellos sin que esperen mucho, sin embargo en el punto medio de la simulación, han aparecido muchos clientes en poco tiempo por lo que estos han ido a la cola, de ahí que el tiempo de espera sea un 16 % mayor que en el modo de aparición anterior.

Respecto a la estrategia, al igual que en el resto de casos, hay más clientes asignados a los primeros taxis del set.

También remarcar que la duración de la simulación es mayor, esto es debido a que en total, si sumamos los periodos entre apariciones de cliente de este modo y del modo constante, el resultado de este será mayor.

Estrategia Aleatorio

Los resultados del experimento 8, combinación con la estrategia aleatorio son los siguientes.

Tiempo total de la simulación:	12.7 mins
Tiempo medio de taxis en movimiento:	63 %
Tiempo medio de taxis parados:	37 %
Tiempo medio de clientes en espera:	72 %
Tiempo medio de clientes en trayecto:	28 %
Tiempo medio de clientes activos:	0.41 mins

Si comparamos los resultados con la prueba anterior vemos como los resultados son muy similares, esto se debe, al igual que en pruebas anteriores, a que el reparto de taxis no sigue ningún tipo de estrategia para minimizar tiempos.

Como ha pasado también en los casos anteriores, la única diferencia con la estrategia anterior es que en esta el reparto de clientes es más uniforme.

Estrategia El más cercano

En esta sección se presentan los resultados del experimento 9, combinación del escenario aleatorio, modo de aparición campana de Gauss y estrategia el más cercano.

Tiempo total de la simulación:	12.3 mins
Tiempo medio de taxis en movimiento:	59 %
Tiempo medio de taxis parados:	41 %
Tiempo medio de clientes en espera:	66 %
Tiempo medio de clientes en trayecto:	34 %
Tiempo medio de clientes activos:	0.33 mins

Con esta estrategia se ve reducirse el tiempo de circulación un 4 %, aunque no es una reducción muy significativa. También se aprecia un poco de descenso en el tiempo de

espera de los clientes pero al igual que antes, no es mucho. Esto se debe al contraste entre los clientes que han aparecido muy rápido, que contrarrestan el tiempo en el que han ido saliendo despacio.

Resumen de los datos de los tres experimentos:

	Estrategia 1	Estrategia 2	Estrategia 3
Tiempo total de la simulación	13min	12.7min	12.3min
Tiempo medio de taxis en movimiento	63 %	63 %	59 %
Tiempo medio de taxis parados	37 %	37 %	41 %
Tiempo medio de clientes en espera	70 %	72 %	66 %
Tiempo medio de clientes en trayecto	30 %	28 %	34 %

Tabla 7.3: Tabla comparativa de los resultados obtenido en los experimentos 7, 8 y 9

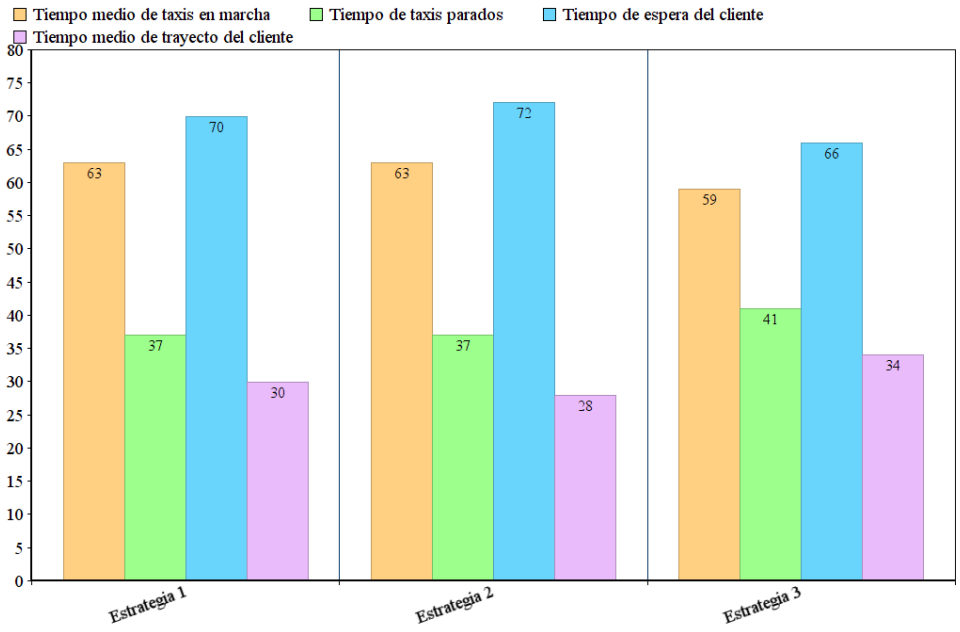

Figura 7.3: Representación gráfica de los resultados de los experimentos 7, 8 y 9

En este modo de aparición, las diferencias de tiempos entre las distintas estrategias no es muy grande. Sin embargo, a pesar de que es muy poco, la estrategia de el más cercano sigue ganando aunque sea por poco en tiempo de espera del cliente y en tiempo de taxis en marcha.

7.1.4. Escenario Concentración y Modo de aparición Todos a la vez

en este grupo de pruebas están los resultados de los experimentos 10, 11 y 12. En este caso, el tipo de escenario es concentración, es decir, la mayoría de los clientes aparecerán en una misma zona, esto lo vamos a combinar con el modo de aparición todos a la vez.

Estrategia El primero libre

Aquí tenemos la primera prueba, que sería el experimento 10, en el que se ha combinado el escenario y el modo de aparición con la estrategia el primero libre.

Tiempo total de la simulación:	6.3 mins
Tiempo medio de taxis en movimiento:	76 %
Tiempo medio de taxis parados:	24 %
Tiempo medio de clientes en espera:	96 %
Tiempo medio de clientes en trayecto:	4 %
Tiempo medio de clientes activos:	2.55 mins

Este caso es muy parecido al primero, con el escenario 1 y el modo 1, de hecho, en todos los casos en los que aparezcan todos a la vez a penas se verá diferencia entre los resultados. Debido a que todos los clientes van a acabar en la cola y serán atendidos por orden por el primer taxi disponible, ya que no habrá taxis con los que poder comparar nada, así que si solamente hay uno disponible, se asignará ese, aunque estén muy lejos.

En esta estrategia volvemos a ver otra vez la mala distribución entre los taxis y con que a su vez, esta se ve un poco suavizada por el modo de aparición, que fuerza a que todos los taxis tengan que recoger a algún cliente ya que el número de demandas es muy superior a la cantidad de taxis.

Si nos fijamos, el tiempo total de ejecución ha vuelto a reducirse, esto se debe a que los clientes aparecen todos a la vez, por lo que todo va más rápido y la ejecución termina antes.

Estrategia Aleatorio

Esto son los resultados del experimento 11, escenario concentración con el modo de aparición todos a la vez y con la estrategia aleatorio.

Tiempo total de la simulación:	5.8 mins
Tiempo medio de taxis en movimiento:	75 %
Tiempo medio de taxis parados:	25 %
Tiempo medio de clientes en espera:	96 %
Tiempo medio de clientes en trayecto:	4 %
Tiempo medio de clientes activos:	2.2mins

En este experimento, al igual que en casos anteriores, apenas hay diferencia entre esta estrategia y la anterior.

Al igual que antes vemos mucho tiempo de movimiento de los taxis debido a la sobrecarga que hay al mismo tiempo y a su vez el tiempo de espera de los clientes es altísimo, como antes, esto se debe a que al aparecer todos a la vez, hay mucho clientes esperando en la cola durante mucho tiempo.

Y como en casos anteriores, la distribución de los taxis mejora entre esta estrategia y la anterior.

Estrategia El más cercano

En este caso, el experimento 12, vemos la misma configuración que antes pero con la estrategia el más cercano.

Tiempo total de la simulación:	5.9 mins
Tiempo medio de taxis en movimiento:	75 %
Tiempo medio de taxis parados:	25 %
Tiempo medio de clientes en espera:	96 %
Tiempo medio de clientes en trayecto:	4 %
Tiempo medio de clientes activos:	2.3 mins

Los resultados de esta prueba son parecidos a los anteriores ya que, como hemos podido comprobar antes, en este modo de aparición la estrategia usada acaba siendo la misma dado que no hay taxis disponibles como para poder elegir el más cercano. No hay ninguna diferencia en los resultados entre esta prueba y las dos anteriores.

Resumen de los datos de los tres experimentos:

	Estrategia 1	**Estrategia 2**	**Estrategia 3**
Tiempo total de la simulación	6.3min	5.8min	5.9min
Tiempo medio de taxis en movimiento	76 %	75 %	75 %
Tiempo medio de taxis parados	24 %	25 %	25 %
Tiempo medio de clientes en espera	96 %	96 %	96 %
Tiempo medio de clientes en trayecto	4 %	4 %	4 %

Tabla 7.4: Tabla comparativa de los resultados obtenidos en los experimentos 10, 11 y 12

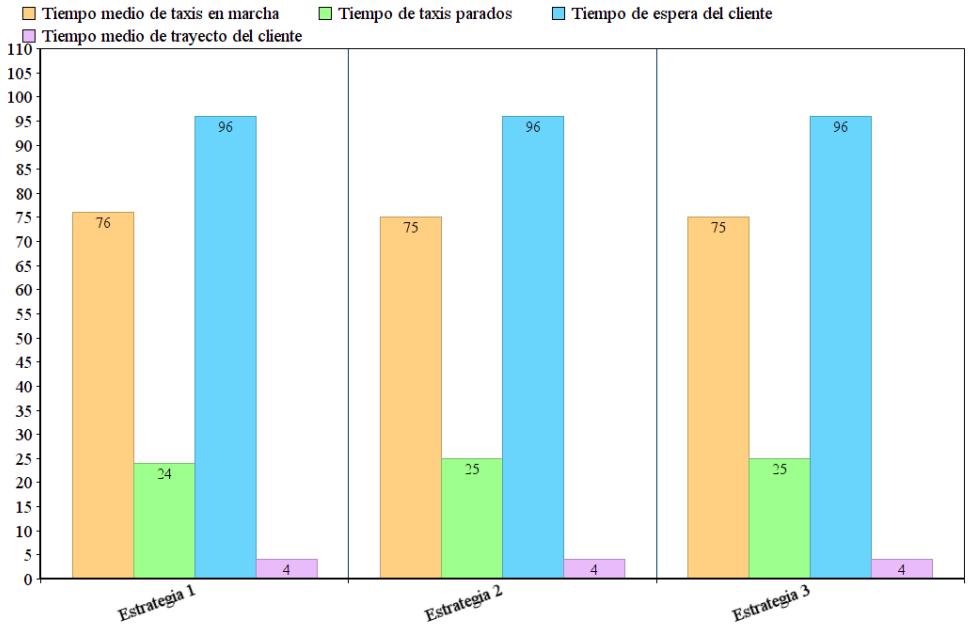

Figura 7.4: Representación gráfica de los resultados de los experimentos 10, 11 y 12

En este grupo de pruebas las diferencias entre los distintos tiempos son mínimas, es normal que en el modo de aparición todos a la vez, al final todas las estrategias acaben comportándose igual. Sin embargo, en el escenario aleatorio con la estrategia el más cercano, había un poco de diferencia, aunque fuese muy poca, esto puede deberse a que en este caso al estar casi todos los clientes en el mismo sitio, los taxis tengan que volver siempre al mismo sitio a por los clientes y haga que no mejore, y en el otro escenario puede ser que un taxi tenga que ir a por un cliente que tiene más cerca.

7.1.5. Escenario Concentración y Modo de aparición Constante

Este set de pruebas corresponden a los experimentos 13, 14 y 15. Con el escenario Concentración y el modo de aparición constante.

Estrategia El primero libre

La primera prueba de este set es el experimento 13. Con la estrategia de coordinación el primero libre.

Tiempo total de la simulación:	9.44 mins
Tiempo medio de taxis en movimiento:	7 %
Tiempo medio de taxis parados:	93 %
Tiempo medio de clientes en espera:	55 %
Tiempo medio de clientes en trayecto:	45 %
Tiempo medio de clientes activos:	0.22 mins

En los experimentos anteriores el tiempo de conducción era muy alto debido al contexto en el que estaban. En este modo de aparición se ve reducirse tanto el tiempo de

movimiento de los taxis como el de espera de los clientes. Como se ha explicado antes, esto se debe a que al ir apareciendo poco a poco los clientes, los taxis van más holgados. Y el tiempo de espera del cliente apenas es un poco más alto que el de trayecto, quedando más o menos proporcionado a la mitad del tiempo para cada parte.

Como en todos los casos anteriores con esta estrategia, los taxis que más clientes recogen son los que están primero en la lista de taxis.

Estrategia Aleatorio

Esta prueba es el experimento 13. Combinación del escenario concentración , modo de aparición constante y estrategia aleatorio.

Tiempo total de la simulación:	9.45 mins
Tiempo medio de taxis en movimiento:	7 %
Tiempo medio de taxis parados:	93 %
Tiempo medio de clientes en espera:	54 %
Tiempo medio de clientes en trayecto:	46 %
Tiempo medio de clientes activos:	0.22 mins

La comparación de estos datos con los obtenidos en la estrategia anterior son muy parecidos, esto se debe, como se ha explicado antes, a que las estrategias son parecidas con respecto al criterio de elegir un taxi para cada cliente.

Con este tipo de entorno, esta combinación de factores, hace que muchos taxis estén parados y aumente este tiempo. Esto se debe también a que hay demasiados taxis para esta prueba.

Estrategia El más cercano

Esta prueba correspondería al experimento 15. Combinando el escenario concentración y modo de aparición constante

Tiempo total de la simulación:	8.9 mins
Tiempo medio de taxis en movimiento:	6 %
Tiempo medio de taxis parados:	94 %
Tiempo medio de clientes en espera:	47 %
Tiempo medio de clientes en trayecto:	53 %
Tiempo medio de clientes activos:	0.18 mins

En los tiempos dados en esta estrategia vemos reducir el tiempo de espera un 7 %, a pesar de no ser muy grande la diferencia, se puede ver como el tiempo total de simulación es un poco menor, seguramente se debe a la reducción del tiempo de espera.

Es posible que el tiempo de espera no se reduzca mucho porque al estar todos los clientes concentrados en un mismo punto, la diferencia de distancias entre un taxi y otro no sea muy grande y , por lo tanto no afecte mucho al resultado final.

Resumen de los datos de los tres experimentos:

	Estrategia 1	Estrategia 2	Estrategia 3
Tiempo total de la simulación	9.44min	9.45min	8.9min
Tiempo medio de taxis en movimiento	7 %	7 %	6 %
Tiempo medio de taxis parados	93 %	93 %	94 %
Tiempo medio de clientes en espera	55 %	54 %	47 %
Tiempo medio de clientes en trayecto	45 %	46 %	53 %

Tabla 7.5: Tabla comparativa de los resultados obtenidos en los experimentos 13, 14 y 15

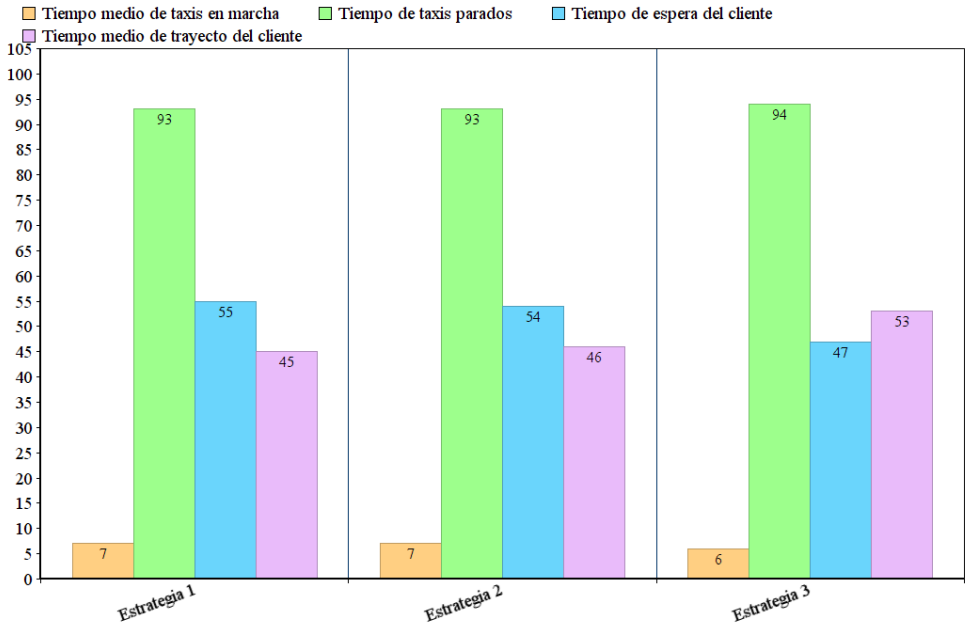

Figura 7.5: Representación gráfica de los resultados de los experimentos 13, 14 y 15

De este escenario, de las conclusiones que podemos sacar, la principal es que hay demasiados taxis, hay demasiado tiempo de taxis parados. Por un lado, que el tiempo de trayecto sea bajo, está bien porque significa que se estará usando menos carburante, sin embargo, si la diferencia es tan drástica es porque hay demasiados taxis.

7.1.6. Escenario Concentración y Modo de aparición Campana de Gauss

En esta sección veremos las pruebas con la configuración concentración y campana de Gauss. A esta combinación pertenecen los experimentos 16, 17 y 18.

Estrategia El primero libre

La combinación con la estrategia el primero libre es el experimento 16.

Tiempo total de la simulación:	9.7 mins
Tiempo medio de taxis en movimiento:	59 %
Tiempo medio de taxis parados:	41 %
Tiempo medio de clientes en espera:	71 %
Tiempo medio de clientes en trayecto:	29 %
Tiempo medio de clientes activos:	0.28 mins

En este modo de aparición vemos que el tiempo que pasan los taxis en movimiento aumenta con respecto a las pruebas anteriores. Esto se debe al periodo de este modo en el que que los clientes aparecen muy seguido. Motivo también de que el tiempo de espera de los clientes aumente.

Al aparecer tan seguidos muchos irán directos a la cola de clientes por lo que su tiempo de espera aumentará. Que haya clientes metidos en cola es porque no había taxis disponibles para llevarlos cuando han aparecido. Por esto es que se han visto aumentados los tiempos medios de los de movimiento de los taxis.

Estrategia Aleatorio

Esta prueba corresponde al experimento 17. Ejecución con la configuración escenario concentración, modo de aparición campana de Gauss y estrategia aleatorio.

Tiempo total de la simulación:	10.7 mins
Tiempo medio de taxis en movimiento:	59 %
Tiempo medio de taxis parados:	41 %
Tiempo medio de clientes en espera:	70 %
Tiempo medio de clientes en trayecto:	30 %
Tiempo medio de clientes activos:	0.3 mins

Los resultados son muy similares a los obtenidos anteriormente, prácticamente la única diferencia que hay con la anterior es que en la el primero libre al principio y al final se asignarán más clientes a los taxis con menor identificador y en el momento de aparición más alta de clientes los taxis estarán todos saturados.

Estrategia El más cercano

Este es el experimento 18, con la configuración escenario concentración, modo de aparición campana de Gauss y estrategia el más cercano.

Tiempo total de la simulación:	9.8 mins
Tiempo medio de taxis en movimiento:	58 %
Tiempo medio de taxis parados:	42 %
Tiempo medio de clientes en espera:	69 %
Tiempo medio de clientes en trayecto:	31 %
Tiempo medio de clientes activos:	0.27 mins

A pesar de haber usado esta estrategia los valores no han variado prácticamente nada, esto se debe a que muy pocos clientes se verán afectados por la mejora que supone el uso de esta estrategia. Esto es debido a que solamente los primeros y los últimos clientes se beneficiarán de minimizar su tiempo de espera ya que en el momento en el que los clientes empiezan a aparecer más rápido los taxis se verán sofocados y estaremos de

nuevo en la situación en la que los clientes estarán en cola esperando al primer taxi libre.

Resumen de los datos de los tres experimentos:

	Estrategia 1	Estrategia 2	Estrategia 3
Tiempo total de la simulación	9.7min	10.7min	9.8min
Tiempo medio de taxis en movimiento	59 %	59 %	58 %
Tiempo medio de taxis parados	41 %	41 %	42 %
Tiempo medio de clientes en espera	71 %	70 %	69 %
Tiempo medio de clientes en trayecto	29 %	30 %	31 %

Tabla 7.6: Tabla comparativa de los resultados obtenidos en los experimentos 16, 17 y 18

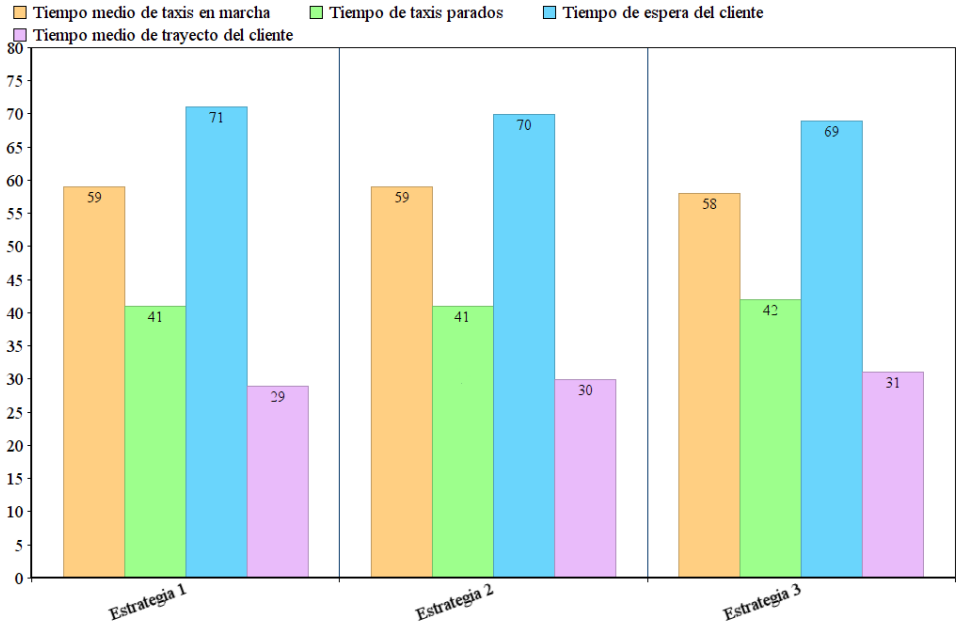

Figura 7.6: Representación gráfica de los resultados de los experimentos 16, 17 y 18

Como conclusión de este set de pruebas se puede decir que no se ven muy afectados los resultados aplicándoles las distintas estrategias. Ejecutando con la estrategia el más cercano, el tiempo de espera de los clientes apenas disminuye un 1-2 % de las otras estrategias y el tiempo de taxis en movimiento disminuye un 1 % con respecto a las otras dos estrategias

7.1.7. Escenario Masivo y Modo de aparición Todos a la vez

Este grupo de pruebas corresponden a los experimentos 19, 20 y 21. Con el escenario masivo y el modo de aparición todos a la vez.

Estrategia El primero libre

Aquí están presentados los resultados del experimento 19, escenario masivo, modo todos a la vez y estrategia el primero libre.

Tiempo total de la simulación:	23.4 mins
Tiempo medio de taxis en movimiento:	87 %
Tiempo medio de taxis parados:	13 %
Tiempo medio de clientes en espera:	96 %
Tiempo medio de clientes en trayecto:	4 %
Tiempo medio de clientes activos:	4.13 mins

El tiempo total de la simulación es mucho más alto que los de las pruebas anteriores, esto se debe a que la densidad de clientes es del doble que antes. Esta multiplicación de los clientes hace que los taxis estén saturados y estén casi el 100 % del tiempo de la simulación en movimiento.

A su vez, el tiempo de espera de los clientes es altísimo, como hemos visto antes, en el modo de aparición todos a la vez, este tiempo siempre es más alto, si además lo combinamos con este escenario que ya de por si va a tener tiempos de circulación de los taxis y de espera de los clientes muy altos, aparecen resultados como este.

De las estrategias no hace falta decir que, si en los escenarios anteriores con el modo de aparición todos a la vez acababan todas las estrategias dando el mismo resultado, en este no va a ser diferente.

En esta estrategia siempre se acumulan clientes en los primero taxis, sin embargo en esta ejecución van tan saturados que ni se nota.

Estrategia Aleatorio

Este experimento es el número 20, combinación del escenario masivo, modo de aparición todos a la vez y estrategia aleatorio.

Tiempo total de la simulación:	23.8 mins
Tiempo medio de taxis en movimiento:	87 %
Tiempo medio de taxis parados:	13 %
Tiempo medio de clientes en espera:	97 %
Tiempo medio de clientes en trayecto:	3 %
Tiempo medio de clientes activos:	4.5 mins

No es de extrañar que los resultados de esta prueba sean tan similares a la anterior ya que las condiciones en las que nos encontramos son las mismas. Por lo que las conclusiones que podemos sacar son las mismas que en el apartado anterior.

Estrategia El más cercano

Experimento 21, combinación del esenario masivo, con el modo de aparición todos a la vez y la estrategia el más cercano.

Tiempo total de la simulación: 22.4 mins
Tiempo medio de taxis en movimiento: 82 %
Tiempo medio de taxis parados: 18 %
Tiempo medio de clientes en espera: 96 %
Tiempo medio de clientes en trayecto: 4 %
Tiempo medio de clientes activos: 3.9 mins

Al igual que antes, los taxis van saturados, sin embargo, en este caso si que se puede apreciar, aunque es leve un poco de descenso del porcentaje de tiempo de espera de los clientes. Seguramente se deba a la última fase de la ejecución en la que ya no hay casi clientes y si que se pueda hacer una mejor elección del taxis elegido.

También se ha reducido un poco el tiempo de movimiento de los taxis, aunque también es muy poco.

Resumen de los datos de los tres experimentos:

	Estrategia 1	**Estrategia 2**	**Estrategia 3**
Tiempo total de la simulación	23.4min	23.8min	22.4min
Tiempo medio de taxis en movimiento	87 %	87 %	82 %
Tiempo medio de taxis parados	13 %	13 %	18 %
Tiempo medio de clientes en espera	96 %	97 %	96 %
Tiempo medio de clientes en trayecto	4 %	3 %	4 %

Tabla 7.7: Tabla comparativa de los resultados obtenidos en los experimentos 19, 20 y 21

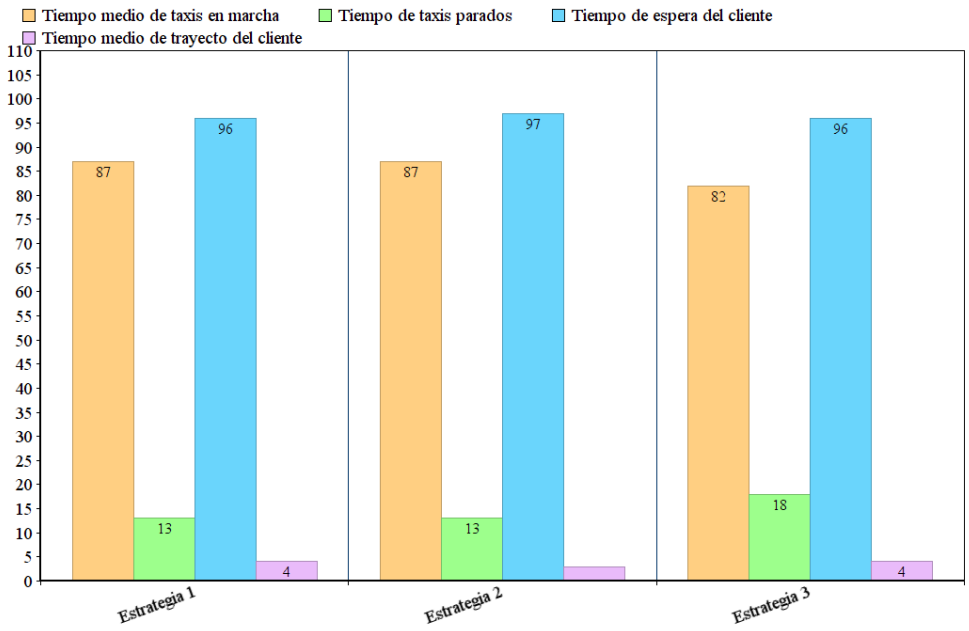

Figura 7.7: Representación gráfica de los resultados de los experimentos 19, 20 y 21

Lo primero que podemos decir de estas primeras ejecuciones con este escenarios es la clara falta de más taxis. Cuando se prevé que va a haber tantos clientes lo mejor sería aumentar el número de taxis.

Además, apenas viene a notarse la diferencia entre el uso de unas estrategias u otras, en el último caso si que se aprecia un poco de disminución de los tiempo de espera y de movimiento, por lo que nos decantaríamos por usar esa.

7.1.8. Escenario Masivo y Modo de aparición Constante

Este set de pruebas serán los experimentos 22, 23 y 24. Con el escenario masivo y el modo de aparición contante combinados cada uno con una estrategia.

Estrategia El primero libre

Este experimento es el 22, con el escenario masivo, modo de aparición constante y la estrategia el primero libre.

Tiempo total de la simulación:	22.4 mins
Tiempo medio de taxis en movimiento:	80 %
Tiempo medio de taxis parados:	20 %
Tiempo medio de clientes en espera:	94 %
Tiempo medio de clientes en trayecto:	6 %
Tiempo medio de clientes activos:	2.11 mins

En esta ejecución se aprecia reducirse un poco el tiempo de movimiento de los taxis, al ir apareciendo los clientes más despacio habrán momentos en los que habrá algún taxi que no tendrán clientes, pero en el momento en el que hayan aparecido tantos clientes como taxis hay, los siguientes irán yendo a la cola de clientes, de ahí que el tiempo de espera siga siendo tan alto.

También se ve una reducción del tiempo de cliente activo, es es porque aunque esperen mucho en la cola, la suma del tiempo total será menor. Aunque el tiempo de espera es demasiado alto.

Estrategia Aleatorio

Este experimento es el número 23, en este caso con la estrategia aleatorio.

Tiempo total de la simulación:	21.7 mins
Tiempo medio de taxis en movimiento:	84 %
Tiempo medio de taxis parados:	16 %
Tiempo medio de clientes en espera:	93 %
Tiempo medio de clientes en trayecto:	7 %
Tiempo medio de clientes activos:	1.9 mins

En esta prueba se ve un aumento del movimiento de los taxis, aunque yo esto lo atribuiría al factor aleatorio, es posible que al asignar aleatoriamente haya hecho un mejor reparto, sin embargo, esto es un poco cuestión de suerte. Si se hubiesen podido realizar pruebas más exhaustivas y poder sacar medias de los resultados obtenidos en todas, seguramente no se apreciaría prácticamente ninguna diferencia.

y lo que nos hemos ido encontrado entre esta estrategia y la anterior es que los clientes se suelen asignar mas equitativamente en esta estrategia que en la anterior. Incluso en estos escenarios en los que los taxis van tan saturados, en los momentos en los que ya hay más calma se llega a apreciar esa diferencia.

Estrategia El más cercano

Aquí se explicarán los resultados obtenidos del experimento 24. Escenario masivo, modo de aparición constante y estrategia el más cercano.

Tiempo total de la simulación:	19.5 mins
Tiempo medio de taxis en movimiento:	87 %
Tiempo medio de taxis parados:	13 %
Tiempo medio de clientes en espera:	85 %
Tiempo medio de clientes en trayecto:	15 %
Tiempo medio de clientes activos:	1.5 mins

Aquí si se aprecia un descenso del tiempo de espera del cliente de un 10 % aproximadamente, este porcentaje dado la situación que tenemos es bastante alto. Lo que mejora bastante la situación.

Puede deberse a que al ir apareciendo más lentamente los clientes, los taxis tanto al inicio como al final de la aplicación van más holgados y el algoritmo tiene taxis libres para elegir al más cercano.

Resumen de los datos de los tres experimentos:

	Estrategia 1	**Estrategia 2**	**Estrategia 3**
Tiempo total de la simulación	22.4min	21.7min	19.5min
Tiempo medio de taxis en movimiento	80 %	84 %	87 %
Tiempo medio de taxis parados	20 %	16 %	13 %
Tiempo medio de clientes en espera	94 %	93 %	85 %
Tiempo medio de clientes en trayecto	6 %	7 %	15 %

Tabla 7.8: Tabla comparativa de los resultado obtenidos en los experimentos 22, 23 y 24

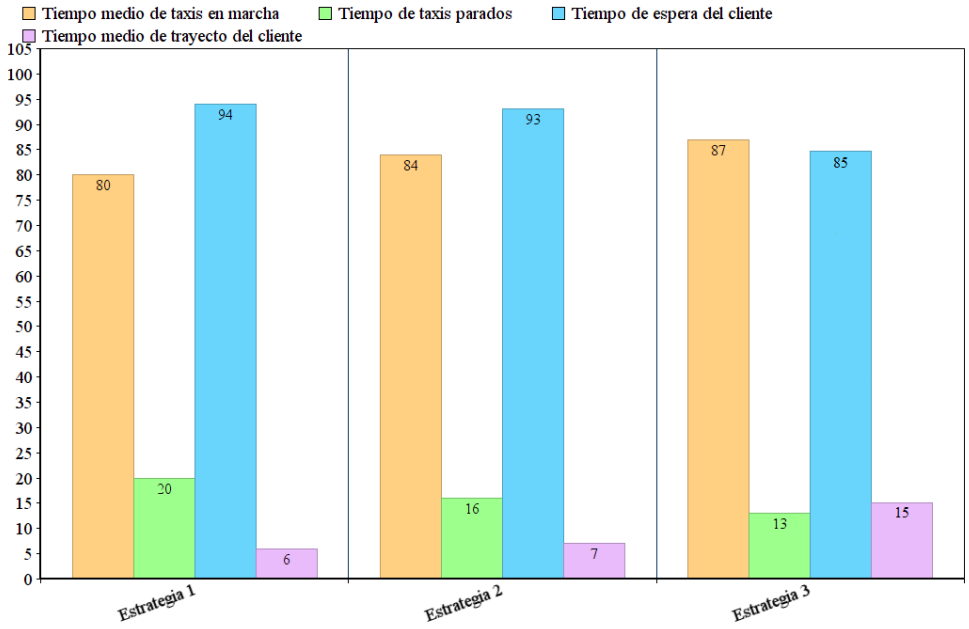

Figura 7.8: Representación gráfica de los resultados de los experimentos 22, 23 y 24

Como conclusión de este set de pruebas, podemos decir, que como en el anterior grupo, hacen falta más taxis, a pesar que en los resultados obtenidos con la estrategia 3 haya descendido el tiempo de espera, por el momento esta es la mejor estrategia y la que mejores resultados está dando, incluso en casos como este.

7.1.9. Escenario Masivo y Modo de aparición Campana de Gauss

Este conjunto de pruebas corresponden a los experimentos 25, 26 y 27. Combinación del escenario masivo con el modo de aparición campana de Gauss.

Estrategia El primero libre

Estos son los resultados obtenidos en el experimento 25, con la estrategia el primero libre:

Tiempo total de la simulación:	24.9 mins
Tiempo medio de taxis en movimiento:	68 %
Tiempo medio de taxis parados:	32 %
Tiempo medio de clientes en espera:	79 %
Tiempo medio de clientes en trayecto:	21 %
Tiempo medio de clientes activos:	2.1 mins

En este experimento el tiempo de taxis parados aumenta, el modo de aparición en campana de Gauss, hacer que tanto al principio como al final los clientes aparezcan muy lentos, por eso, durante esos periodos la mayoría de los taxis están parados.

Que el tiempo de espera de los clientes se reduzca también se debe a lo mismo, ya que mientras no haya muchos clientes habrá suficientes taxis como para abastecer a los clientes.

Durante esos periodos, debido a la estrategia vemos que los primeros taxis del set son los que van a por casi todos los clientes. En el momento en el que aparecen muchos clientes esto se suaviza debido a que van a estar saturados y todos los taxis tendrán que llevar a clientes.

En esta combinación el tiempo total de la simulación siempre aumenta un poco debido a que el tiempo que tardan en ir apareciendo los clientes es un poco mayor.

Estrategia Aleatorio

La combinación del escenario masivo, el modo de aparición campana de Gauss y la estrategia aleatorio es el experimento 26.

Tiempo total de la simulación:	26.7 mins
Tiempo medio de taxis en movimiento:	64 %
Tiempo medio de taxis parados:	36 %
Tiempo medio de clientes en espera:	80 %
Tiempo medio de clientes en trayecto:	20 %
Tiempo medio de clientes activos:	2.2 mins

Los resultados de este experimento son iguales al anterior, apenas se nota mucha diferencia, una vez vistos los casos anteriores, esto es normal, los resultados de estas dos estrategias son muy similares en todos los casos, solamente cambia en que esta estrategia distribuye mejor los clientes entre los taxis.

Estrategia El más cercano

Este es el experimento 27. Combinación del escenario masivo, modo de aparición campana de Gauss y estrategia el más cercano.

Tiempo total de la simulación:	24.4 mins
Tiempo medio de taxis en movimiento:	65 %
Tiempo medio de taxis parados:	35 %
Tiempo medio de clientes en espera:	79 %
Tiempo medio de clientes en trayecto:	21 %
Tiempo medio de clientes activos:	1.82 mins

Y como en las pruebas anteriores, en este escenario apenas afecta la elección de la estrategia debido a que hay demasiados pocos taxis y no da para que se pueda elegir entre varios taxis.

Sin embargo, el tiempo total invertido por los clientes, desde que aparecen hasta que llegan a su destino, disminuye un poco, aunque la proporción de tiempo de espera es igual que en las estrategias anteriores.

Resumen de los datos de los tres experimentos:

	Estrategia 1	Estrategia 2	Estrategia 3
Tiempo total de la simulación	24.9min	26.7min	24.4min
Tiempo medio de taxis en movimiento	68 %	64 %	65 %
Tiempo medio de taxis parados	32 %	36 %	35 %
Tiempo medio de clientes en espera	79 %	80 %	79 %
Tiempo medio de clientes en trayecto	21 %	20 %	21 %

Tabla 7.9: Tabla comparativa de los resultados obtenidos en los experimentos 25, 26 y 27

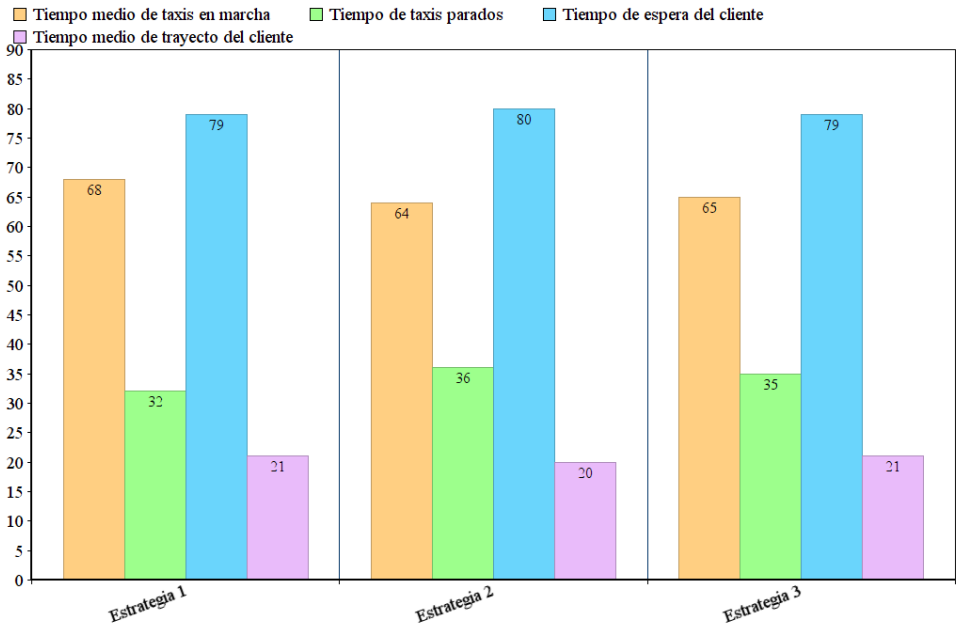

Figura 7.9: representación gráfica de los resultados de los experimentos 25, 26 y 27

De este grupo de pruebas, como conclusión podemos sacar que la elección de las estrategias no es un factor muy importante, no al menos con la cantidad de taxis que hay, seguramente si se hubiesen hecho las pruebas con más taxis se apreciaría más el uso de una estrategia u otra.

7.2 Resultados globales

En este apartado vamos a ver los resultados desde un punto de vista más general.

Hemos hecho las gráficas separadas por escenarios.

7.2.1. Leyenda

Esta es la leyenda de colores utilizadas en las gráficas:

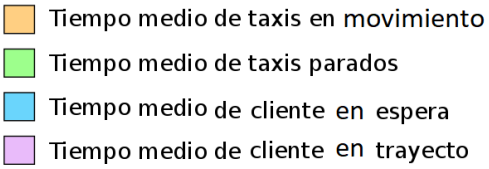

Figura 7.10: Leyenda

7.2.2. Escenario 1: Aleatorio

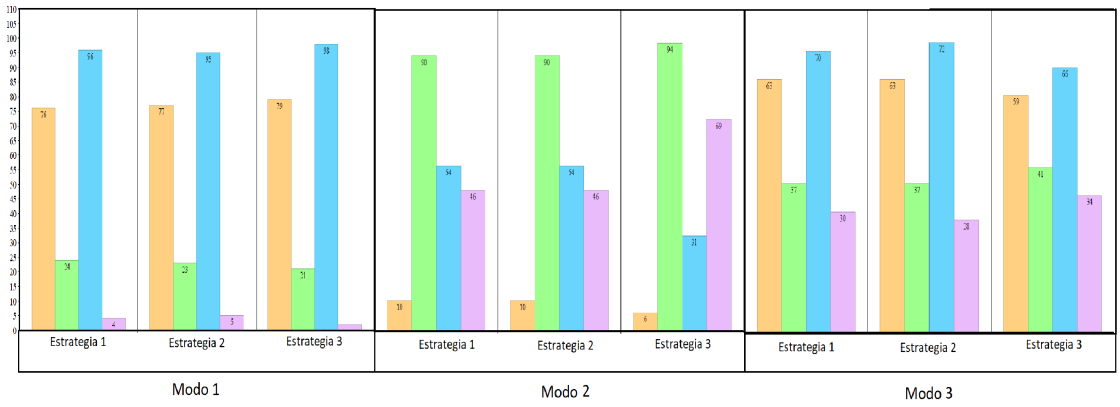

Figura 7.11: Representación gráfica de los experimentos con el escenario aleatorio

En este escenario las diferencias entre unos resultados y otros son muy grandes. En el modo de aparición 1, todos a la vez, no se aprecian diferencias muy grandes entre las distintas estrategias, esto se debe, como ya hemos podido ir viendo antes con los experimentos uno por uno, al ir a parar todos los clientes a la cola, no hay ninguna diferencia entre usar unas u otras estrategias. Ya que cada taxi que termine será el encargado de atender al primer cliente de la cola, esté donde esté.

En el modo 2, constante, si que se nota el cambio de estrategia, las dos primeras, el primero libre y aleatorio, los resultados son similares como ya hemos podido ver en los resultados, sin embargo, la estrategia el más cercano si qye se puede apreciar un descenso del tiempo de espera de los clientes. Dado que ahora los clientes van apareciendo más despacio, si que hay taxis para poder elegir cuando aparece un cliente.

También se puede ver que el tiempo de espera de taxis parados es muy alto. Que el tiempo de trayecto sea poco es bueno, pero que la diferencia sea tan grande nos dice que sobran taxis para este escenario con este modo de aparición.

Las últimas tres gráficas pertenecen al modo de aparición 3, campana de Gauss, en este caso, los resultados pueden verse como un punto intermedio entre los observados en el modo 1 y el modo 2. Esto se debe a que el inicio y el final en este modo van apareciendo despacio y por la mitad aparecen muy rápido, tanto como en el modo de aparición 1, por eso es normal que los resultados sean algo entre medio.

En este experimento la elección de la estrategia si que se nota, aunque la mejora no sea muy grande, si que se disminuyen un poco los tiempo de espera de los clientes con la estrategia 3 y el tiempo de taxis en movimiento se reduce también un poco.

7.2.3. Escenario 2: Concentración

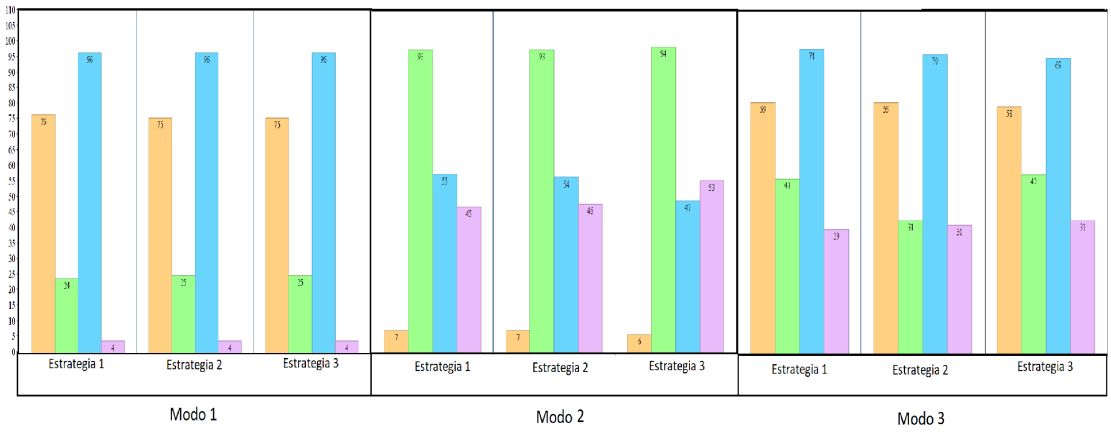

Figura 7.12: Representación gráfica de los experimentos con el escenario concentración

En este escenario, con el modo de aparición 1, nos encontramos en una situación bastante similar a la anterior. El hecho de que todos los clientes aparezcan al mismo tiempo hace que las estrategias acaben actuando del mismo modo ya que para cada cliente solamente habrá un taxi disponible.

En el modo de aparición constante, si que vemos diferencias al usar la estrategia el más cercano, se ve como el tiempo de espera del cliente disminuye con respecto a las otras estrategias. Por otro lado, los tiempos de circulación también se ven un poco reducidos, aunque no mucho.

Al igual que en el escenario anterior, con el modo de aparición 2, el tiempo de taxis parados aumenta mucho, esto se debe al exceso de taxis para manejar esta situación.

Volvemos a ver como el modo 3, está como entre el modo 1 y el modo 2. Con este modo en este escenario el tiempo de espera de los clientes es muy alto, incluso con la estrategia el más cercano.

Esto se debe a que, al estar todos lo clientes concentrados en un punto, las distancias entre los clientes y los taxis serán parecidas, de modo que el taxi más cercano, del segundo más cercano, la diferencia de distancia con el cliente debe ser muy poca.

Además, en el pico de apariciones de clientes, los taxis se ven saturados y se encolarán muchos clientes de modo qu el tiempo de espera aumentará.

7.2.4. Escenario 3: Masivo

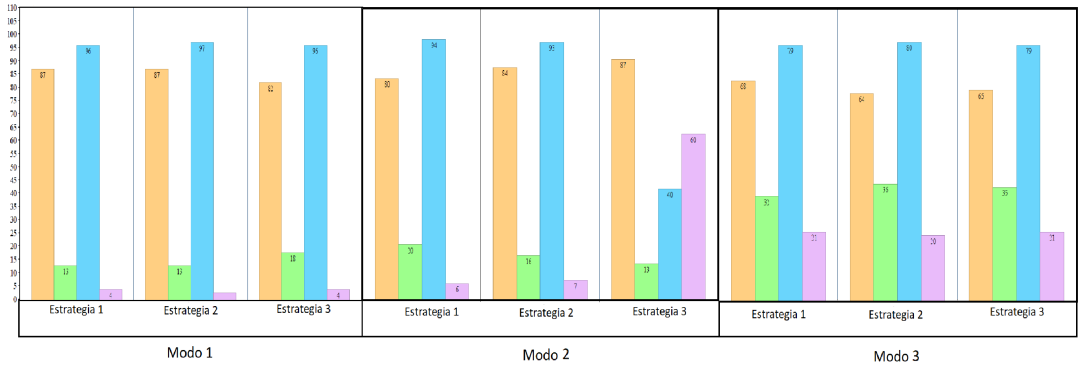

Figura 7.13: Representación gráfica de los experimentos con el escenario masivo

Con este escenario, el duplicar la cantidad de clientes que hay, hace que estén los taxis saturados durante toda la simulación, independientemente de como vayan apareciendo.

En el primer modo de aparición, todos a la vez, a penas se nota si se ha usado una estrategia u otra.

En la estrategia le más cercano, se aprecia que el tiempo de trayecto de los taxis disminuye un poco, aún así nada muy significante.

En el modo de aparición constante, si que se aprecia una pequeña disminución del tiempo de espera de los clientes.

A pesar de que no sea muy grande, dada la cantidad de clientes que hay, si que es una cantidad significativa. También se ve disminuir un poco el tiempo de movimiento de los taxis.

El modo de aparición 3, campana de Gauss, las tres gráficas son extremadamente parecidas, en este caso no hay diferencia entre estrategias en ninguno de los tiempos. Pero si comparas con los resultados con la gráfica del modo 1, si que se puede apreciar que, en este modo de aparición, los tiempos de espera de los clientes son mejores.

Como conclusiones de las pruebas, podemos decir que los resultados se acercan bastante a lo que nos esperábamos, los tiempos varían mucho según el modo de aparición y el escenario, unos más que otros pero en general, como hemos ido comentando antes, los resultados son bastante razonables.

En prácticamente todas las pruebas, la estrategia el más cercano mejoraba los tiempos, excepto en los casos más extremos, que simplemente daba resultados similares a los de las otras estrategias. Por ende, la estrategia con mejores resultados en todos los casos ha sido la estrategia 3.

Había pruebas en las que se necesitaban más taxis y otras que menos, la cantidad de taxis ha sido un valor fijo en nuestras pruebas, sin embargo, es un factor importante a tener en cuenta al planificar como va a funcionar una flote de vehículos.

Decisiones sobre donde poner inicialmente los taxis, se debe elegir mediante el análisis de un histórico de datos en el que aparezcan las zonas en las que hay más densidad de aparición de clientes, como puedan ser el aeropuerto, hospitales, estaciones de tren, etc.

Sin embargo, mediante este tipo de pruebas se puede hacer una previsión sobre si van a hacer faltar más taxis o van a sobrar.

CAPÍTULO 8
Conclusiones

En este capítulo vamos a realizar un repaso de todo lo que se ha hecho, de los problemas encontrados y de posibles mejoras.

8.1 Trabajo realizado

Lo primero que hicimos al empezar el proyecto fue fijar unos objetivos a los que llegar.

El objetivo general era crear un simulador del comportamiento de una flota de vehículos con el que poder ejecutar y probar situaciones y poder hacer análisis de los resultados obtenidos.

Y entre los objetivos específicos entraban crear una interfaz amigable desde la que se pudiese visualizar la ejecución y por donde mostrar los resultados.

También, una vez realizado el simulador, hacer pruebas de validación, para comprobar si funcionaba, una vez comprobado que funcionaba, crear casos que podrían darse e implementar estrategias de coordinación y hacer pruebas.

Una vez realizadas las pruebas hacer un análisis de los resultados obtenidos.

Como hemos podido ver a lo largo de la memoria, estos objetivos se han cumplido todos, lo que no significa que no hayan posibles mejoras para la aplicación.

8.2 Posibles mejoras

Como posibles mejoras de la herramienta se podrían hacer algunas modificaciones para que la aplicación pueda ser usada desde un terminal móvil.

Otra posible mejora que ya estoy llevando a cabo, es el uso de la librería Node.js, esta librería nos habría permitido una ejecución con una comunicación más sencilla, dinámica y, sobre todo, bidireccional.

También podría hacerse una interfaz más dinámica en la que se pudiese elegir el formato de los resultados o que se añadieran al formulario de elección de estrategias, por ejemplo, ahora mismo las opciones de elección de estrategia y de modo de aparición están puestas elemento a elemento de forma estática en la interfaz, en caso de añadir una nueva estrategia y querer poder elegirla desde la interfaz se debería añadir la opción en la vista.

Una mejor podría ser hacer este proceso automático para así facilitar más a los usuarios la adaptación de nuevas estrategias a la aplicación.

8.3 Problemas

El principal problema que hemos tenido ha sido durante las pruebas y es la limitación que da, a veces, usar librerías de terceros, en nuestro caso, el uso de los servicios de obtención de rutas que teníamos era limitado, tanto en peticiones mensuales como en peticiones por segundo, esto ha dado problemas para la ejecución ya que si una petición fallaba había que volver a empezar.

Esto ha provocado que no se hayan podido hacer pruebas con la estrategia el más cercano por distancia en lugar de la distancia euclídea ya que para saber la distancia y poder encontrar la más corta habría que hacer demasiadas peticiones al servidor de rutas.

A su vez, también se ha implementado la estrategia el más cercano con corrección, esto hace que si el cliente no ha subido al taxi que se le ha asignado, este pueda ser asignado a otro cliente si se considera que va a ser más óptimo. No se han podido hacer pruebas ya que se multiplican las peticiones a los servidores de rutas y se nos agotan antes de poder terminar la simulación.

Bibliografía

[1] Entrada de la wikipedia Consultado en https://es.wikipedia.org/wiki/Agente_inteligente_(inteligencia_artificial).

[2] Entrada de la página web de comunidad informática CMM Consultado en http://es.ccm.net/faq/4474-los-agentes-inteligentes.

[3] Artículo de la Revista internacional científica y profesional, escrito por Pedro Hípola y Benjamín Vargas-Quesada emitido en abril del 1999 Consultado en http://www.elprofesionaldelainformacion.com/contenidos/1999/abril/agentes_inteligentes_definicion_y_tipologia_los_agentes_de_informacion.html.

[4] Apuntes de la asignatura de Agentes Inteligentes de la Universidad Politécnica de Valencia, escrito por Carlos Carrascosa. Consultado en http://personales.upv.es/ccarrasc/doc/2003-2004/AI_Web/definicion.html

[5] Memoria del trabajo Aplicación de agentes cognitivos en vehículos autónomos inteligentes, Consultado en http://www.it.uc3m.es/jvillena/irc/practicas/09-10/13mem.pdf.

[6] Página oficial de NetLogo Consultado en https://ccl.northwestern.edu/netlogo/

[7] Página oficial de SimPy Consultado en https://simpy.readthedocs.io/en/latest/

[8] Página oficial de Anylogic Consultado en https://www.anylogic.com/

[9] Página oficial de Repast Consultado en https://repast.github.io/

[10] Página oficial de Leaflet Consultado en http://leafletjs.com/

[11] Página del proyecto de github MovingMarkers Consultado en https://github.com/ewoken/Leaflet.MovingMarker

[12] Página oficial de Flask Consultado en http://flask.pocoo.org/

[13] Página oficial de Google Maps API Consultado en https://developers.google.com/maps/documentation/directions/?hl=es-419

[14] Página oficial de MapQuest Consultado en https://www.mapquest.com/

[15] Página oficial de MapQuest Consultado en http://hello.mapquest.com

[16] Página oficial de Open Street Map Consultado en http://www.openstreetmap.org/

[17] Página oficial de docker Consultado en https://www.docker.com/

[18] Página oficial de Dockerfile Consultado en https://docs.docker.com/engine/reference/builder/

[19] Página oficial de docker-compose Consultado en https://docs.docker.com/compose/

APÉNDICE A
Manual de uso

El uso de la aplicación es muy sencillo.

Para ejecutar la simulación deberemos cargar los fichero de datos de clientes y de taxis que queremos usar, indicaremos el número de taxis y clientes que queremos que participen en la simulación. Se deberá seleccionar qué estrategia se quiere utilizar y el modo de aparición que se prefiera.

La aplicación tiene unos valores por defecto para, en caso de no rellenar los campos se ejecute de todos modos.

En la siguiente sección se explicará como crear los sets de datos y el formato que deben tener.

A.1 Sets de datos

Para la creación de sets de datos se debe tener en cuenta el formato de estos ya que sino, el servidor no los va a entender.

Vamos a ver como se han hecho los datos de los taxis y los clientes.

A.1.1. Taxis

En caso de querer meter tu propio set de taxis en el simulador, este debe contar con la siguiente estructura:

0,39.47778,-0.37718
1,39.47552,-0.3798
...

Cada una de las lineas representará un taxis, del cual, el primer número será su identificador, y los dos siguientes serán, respectivamente, la latitud y la longitud de su posición inicial.

El identificador no tiene porqué ser un número, puede coger el nombre que se quiera.

A.1.2. Clientes

Para los sets de clientes, si se quiere crear uno nuevo, se debe seguir la siguiente estructura:

0,39.47188,-0.37688,39.46429,-0.36684
1,39.47172,-0.37036,39.47543,-0.37611
...

En este caso, el cliente debe tener un origen y un destino, es decir, donde está y donde quiere ir. Al igual que los taxis, el primer valor es su identificador, y puede tener el valor que se quiera. Y los dos siguientes valores, son las coordenadas de su origen y las dos siguientes las de su destino.

Es muy importante separar los valores por comas (,) dado que el servidor va a ir a buscarlas para separar los valores y crear las estructuras de datos con las que trabajará él.

A.2 Servidor

En esta sección vamos a explicar todas las partes del servidor, en caso de querer sustituir alguna parte se debe seguir las especificaciones que hemos escrito aquí para que no haya fallos en la comunicación y todo siga siendo coherente.

A.2.1. Controlador

Para cambiar el controlador se deberá tener en cuenta la comunicación que tiene con el resto de servicios.

onPlay

Este método es llamado desde el cliente para iniciar la simulación.

Devuelve cargados todos los datos en forma de diccionario. Ese diccionario contendrá tanto los taxis como los clientes.

El diccionario contendrá las dos estructuras de datos siguientes:

```
Taxis {  0:  {
                position:     Tupla
                assigned:     Booleano
                taken:        Booleano
             }
          ...
}
```

```
Clientes { 0:  {
                origin:        Tupla
                destination:   Tupla
                ready:         Booleano
                going:         Booleano
                done:          Booleano
             }
          ...
}
```

Para recuperar el diccionario de taxis se puede acceder con la clave 'Taxis', es decir, esta instrucción: **respuesta['taxis']** devolverá todos los taxis con la misma forma de la imagen de arriba. para el caso de los clientes será igual pero la clave sera 'clientes'.

La forma de llamar a los métodos desde el cliente, es hacer una petición de tipo GET a la URL siguiente

http://*host*/simulator/onPlay/<numT>/<numC>/<chosenStrategie>

Donde *host* dependerá de donde hayas levantado el servidor, nosotros lo hemos ejecutado siempre desde local por lo que se sustituiría *host* por localhost. El parámetro numT y numC son las cantidades de taxis y clientes respectivamente que el usuario ha seleccionado desde la interfaz y la estrategia elegida también será la que se habrá elegido desde la interfaz.

newClient

Este método recibirá el identificador del cliente que ha aparecido en el mapa y que solicita un taxi.

Se hará un cálculo y responderá al cliente con una lista en la que el primer parámetro será un booleano. Este parámetro identificará si se le ha asignado algún taxi al cliente. En caso de ser verdadero irá seguido del índice del taxi, la ruta para recoger al cliente, la velocidad calculada para ese tramo, la ruta para llevar al cliente de su origen hasta su destino y la velocidad. En resumen,

[True, assignedTaxi, taxiToClient, taxiToClientVel, originToDestin, originToDestinVel].

En caso que no se le haya asignado ningún taxi se meterá el cliente en cola, de este modo la respuesta será una lista de un solo elemento, será un Booleano con el valor False.

La forma de la petición será la siguiente:

http://simulator/newClient/<positions>/<clientId>

Aquí el cliente le comunicará al servidor todas las posiciones actuales de los taxis en el mapa con una lista de listas con dos coordenadas, es decir, [[x,y],[...], ...] y el servidor actualiza las posiciones de los taxis. También se le pasará el identificador del cliente que ha aparecido para poder hacer el cálculo de la ruta.

newTaxi

En este caso, en la petición también de tipo GET se le mandará las posiciones, igual que el caso anterior, el taxiId será el identificador del taxi que acaba de terminar su trabajo y el clientId es el identificador del cliente al que el taxi acaba de dejar en su destino.

http://*host*/simulator/newTaxi/<positions>/<taxiId>/<clientId>

La forma en la que este método recibe y devuleve información es igual que en newClient solo que en este caso, se manda el índice del cliente que acaba de dejar ese taxi, ya que este método es el que se dispara cuando el taxi ha dejado al cliente en su destino.

Las respuetas de este método son iguales que en el caso anterior excepto por el hecho de que en este método se va a estar haciendo una llamada para comprobar si todos los clientes están ya en su destino, en caso de que así sea, se hará una llamada al logger que analiza el log generado por el servidor y en base a él, hace cálculos sobre los tiempos que se han ido dando a lo largo de la simulación por lo que se devolverá lo siguiente:

[“End”, drivingTimeAvg, stopedTimeAvg, waitingTimeAvg, trajectTimeAvg, totalTimeAvg, duration]

Esto es, del primer miembro indicará al cliente que la simulación ha terminado, el segundo indicará la media de tiempo que los taxis han estado en movimiento, el tercero lo mismo pero estando parados, el cuarto miembro será el tiempo medio de clientes en espera, el quinto es la media del tiempo de trayecto y el sexto el de duración total media de cliente, es decir, espera y trayecto. Por último la duración total de la simulación.

A.2.2. Estrategias

Las estrategias funcionan todas del mismo modo, se les pasará como parámetro los taxis, los clientes y el identificador del cliente al que se le quiera asignad un taxi.

La petición será un GET asi:

http://*host*/strategies/*strategie*/<taxis>/<clients>/<idClient>

En esta URL, el parámetro *strategie* dependerá de la estrategia elegida, el resultado de todas las estrategias será el identificador del taxi asignado.

A.2.3. Rutas

Para las rutas, la petición debe hacerse de la forma siguiente:

http://*host*/routes/<taxiPosition>/<clientOrigin>/<clientDestination>

De modo que se mandarán dos listas de coordenadas, una será la ruta de ida del taxi hasta el cliente y la otra desde donde está el cliente hasta su destino. Se mandará también las distancias de ambos tramos para el cálculo de la velocidad que debe tomar el taxi para que esta sea uniforme en todos los casos. De modo que la respuesta podría queda como sigue:

[route1, route2, distance1, distance2]

A.2.4. Logger

El logger es el servicio que se llamará al finalizar la simulación, este servicio lo que hará será recoger el fichero de texto en el que se ha escrito el log, o historial de acciones hechas por el simulador y analizar los tiempos que tienen, esto se ha implementado mediante expresiones regulares que siguen patrones, estos patrones son los mismo que se han seguido en la implementación del controlador del servidor y así se puede analizar de forma sencilla los tiempos que ha durado por ejemplo la espera de un cliente, en este caso lo que se imprime en el log es lo siguiente:

Client 0 appears:1502791952442
Client 0 taken:1502791963631:by 1
Client 0 done:1502791969768

De este modo, analizando las lineas y almacenando algunos datos podemos saber cuanto ha tardado en recogerlo algún taxi restando el tiempo *appears* al tiempo *taken* e igual para el trayecto y también sabemos quién ha sido el taxi que se ha hecho cargo de este cliente. Los tiempos están en milisegundos, para tenerlo en otra unidad habría que pasarlo, aunque nosotros por uniformidad y para poder hacer una relación más sencilla, lo hemos pasado a porcentaje.

Para la ejecución de todos los servicios basta con colocarse en la raíz del proyecto y ejecutar el comando:
docker-compose up

De este modo se levantarán todos los servicios a la vez.

A.3 Cliente

A.3.1. Vista

Si lo que te gustaría cambiar es la vista, lo que deberás hacer es tener en cuenta que desde el controlador se usan los identificadores de los elementos para asignarles un contenido, esto deberás seguirlo o modificar el controlador para que se adapte a los nombres de tus identificadores.

A.3.2. Controlador

Aquí en caso de querer cambiar solamente el controlador de la interfaz, si creas uno de cero deberás seguir el estándar de comunicación explicado arriba para que cuadren los datos enviados.

APÉNDICE B
Configuración del sistema

B.1 Herramientas y tecnologías necesarias:

Para poder ejecutar la aplicación hay que tener instalado para el servidor:

- **Python3**: para poder ejecutar Python debes descargarlo desde aquí: *https://www.python.org/downloads/*
- **Flask**: levantar el servicio en python, nos hemos ayudado del framework Flask, aqui dejamos su tutorial de instalación en inglés: *http://flask.pocoo.org/docs/0.12/installation/*
- **Leaflet**: aquí dejamos un tutorial bastante básico y fácil de comprender que ha hecho el equipo de leaflet aquí: *http://leafletjs.com/examples/quick-start/*
- **movingMarker**: esta es la librería en la que nos hemos respaldado para que los marcadores puedan moverse: *https://github.com/ewoken/Leaflet.MovingMarker*

B.2 Estructura de directorios:

Para docker la estructura de directorios es muy importante de modo que la estructura del proyecto debe ser como sigue:

Simulador:

- docker-compose.yml
- **Servidor**
  - Dockerfile
  - server.py
  - **templates**
    - index.html
  - **static**
    - **dataSets**
    - **js**
    - **movingMarker**
    - **styles**

Strategies

Dockerfile

strategies.py

Routes

Dockerfile

routes.py

Logger

Dockerfile

logger.py