UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Diseño y análisis de metaheurísticas basadas en simulación para la optimización de la localización estática de vehículos de emergencias sanitarias

Trabajo Fin de Grado

Grado en Ciencia de Datos

AUTOR/A: Gallego Andreu, Carlos

Tutor/a: Sánchez Anguix, Víctor

Cotutor/a: Alberola Oltra, Juan Miguel

Cotutor/a externo: DE SANTIS, MARIANNA

CURSO ACADÉMICO: 2022/2023

Resumen

Los vehículos de emergencia sanitaria por ejemplo, ambulancias, son un recurso escaso y costoso cuyo uso debe ser optimizado para garantizar la mejor calidad de servicio. En términos sanitarios, la calidad del servicio en el transporte urgente está relacionada con el tiempo de respuesta de la ambulancia. Es decir, el tiempo comprendido entre que se recibe la llamada por parte de los servicios de emergencia y la llegada de la ambulancia al lugar de la incidencia. El problema de la localización estática consiste en determinar la estación de inicio y reposo de un conjunto de ambulancias entre un conjunto de posibles localizaciones. Esta configuración inicial es importante puesto que determina aquellos sectores de la población que quedan cubiertos, así como los tiempos de respuesta una vez que el sistema comienza a funcionar y se comienzan a recibir llamadas. En este proyecto nos apoyamos en el uso de algoritmos metaheurísticos para la optimización de la localización estática de las ambulancias en el área metropolitana de Valencia, haciendo uso de simulación basada en datos del año 2019, para la correcta evaluación de la solución propuesta por las metaheurísticas para el problema de la localización estática.

Palabras clave: Metaheurísticos, Optimización, Simulación, Salud, Ambulancias

Resum

Els vehicles d'emergència sanitària per exemple, ambulàncies, són recursos escassos i costosos, l'ús dels quals ha de ser optimitzat per a garantir la millor qualitat de servei. En termes sanitaris, la qualitat del servei en el transport urgent està relacionada amb el temps de resposta de l'ambulància. És a dir, el temps comprés entre que es rep la telefonada per part dels serveis d'emergència i l'arribada de l'ambulància al lloc de la incidència. El problema de la localització estàtica consisteix a determinar l'estació d'inici i repòs d'un conjunt d'ambulàncies, entre un altre conjunt de possibles localitzacions. Aquesta configuració inicial és important perquè determina aquells sectors de la població que queden coberts, així com els temps de resposta una vegada que el sistema comença a funcionar i es comencen a rebre cridades. En aquest projecte ens recolzem en l'ús d'algorismes metaheuristics per a l'optimització de la localització estàtica de les ambulàncies en l'àrea metropolitana de València, fent ús de simulació basada en dades de l'any 2019, per a la correcta avaluació de la solució proposada per les metaheuristiques per al problema de la localització estàtica.

Paraules clau: Metaheuristics, Optimització, Simulació, Salut, Ambulàncies

Abstract

Emergency medical vehicles (i.e., ambulances) are a scarce and costly resource whose use must be optimized to ensure the best quality of service. In healthcare terms, the quality of service in emergency transport is related to the ambulance response time. That is, the time from which the call is received by the emergency services to the arrival of the ambulance at the scene of the incident. The static location problem consists of determining the starting and resting station of a set of ambulances from a set of possible locations. This initial configuration is key, since it determines which sectors of the population are covered, as well as the response times once the system starts to operate and calls are received. In this project we rely on the use of metaheuristic algorithms for the optimization of the static location of ambulances in the metropolitan area of Valencia, making use of simulation based on data from the year 2019, for the correct evaluation of the solution proposed by the metaheuristics for the static location problem.

Keywords: Metaheuristics, Optimization, Simulation, Health, Ambulances

Contenido

Índice de Ilustraciones

Capítulo 1. Introducción

España podría considerarse como uno de los países con mejor sanidad del mundo (Intergeneracional, 2018), por cosas como la alta esperanza de vida y las bajas tasas de mortalidad por causas evitables, todo ello a pesar de ser uno de los países europeos que menos gasta en sanidad pública en cuanto a % del PIB. La sanidad y el afán por salvar vidas además es vocación de muchos españoles, siendo las carreras relacionadas con la enfermería y la medicina unas de las más demandadas. Dentro de la sanidad privada, algunos de nuestros médicos o cardiólogos son reconocidos a nivel mundial (ConSalud, 2020), sin embargo, la gran mayoría de españoles utiliza los servicios públicos sanitarios, que son suficiente para garantizar el acceso equitativo a servicios médicos de calidad para todos los ciudadanos, independientemente de su situación económica o social.

La Sanidad Pública española se podría dividir en 2 grandes grupos; la atención primaria, donde se garantiza que cada paciente tenga respuesta a los problemas de salud más comunes, a través de un médico de cabecera, y la atención especializada, cuando los problemas de salud son más complejos y requieren de un médico especialista en un ámbito de la salud concreto. Dentro de estos grupos, podría resultar que necesitemos acudir en forma de urgencia, donde el tiempo de respuesta es clave si se trata de un problema muy grave. Si bien hay muchas urgencias donde afortunadamente podemos desplazarnos por nosotros mismos hasta el centro de atención de urgencias más cercano, hay muchas otras situaciones donde será necesario que sean los servicios médicos los que vengan hacia nosotros. Esta pequeña pero importante parte del sistema sanitario se llama Servicio de Emergencias Sanitarias (SES) y es la que abordaremos en este Trabajo Fin de Grado.

El SES es responsable de brindar atención médica de emergencia a personas que sufren lesiones o enfermedades agudas que ponen en peligro su vida o su salud. Algunas de las tareas de esta entidad son la atención de llamadas para evaluar la gravedad de la situación y determinar qué respuesta es necesaria, la coordinación de ambulancias y el transporte seguro y rápido de pacientes. Su objetivo principal es alcanzar a los pacientes, atenderles y estabilizarlos con los recursos médicos correctos, y si lo necesitaran, trasladarlos a un entorno hospitalario para que reciban atención médica más especializada.

En el área metropolitana de la ciudad de Valencia, el SES tiene implantado un sistema estático de vehículos de emergencia. Esto quiere decir las ambulancias tienen una base asignada y acaban volviendo a ella cuando no hay más pacientes que atender, al contrario que en otros sistemas en los cuales los vehículos pueden alterar su estación a lo largo del día o mes. Distribuir estas ambulancias de forma estratégica es de vital importancia para poder afrontar situaciones críticas, ya que de ello depende la capacidad de respuesta y el tiempo de llegada a los pacientes. Gestionar bien dónde colocamos los vehículos no solo permite salvar vidas, sino que también contribuye a evitar la saturación de los servicios en áreas de mucha demanda, así como a ahorrar en términos económicos, al reducir costos asociados al desplazamiento de ambulancias y al mejorar la utilización del personal sanitario.

Si lo vemos de esta forma, sería ideal poder distribuir los vehículos de emergencia donde más lo necesite la población, pero puede ser difícil elegir el método ¿Deberíamos priorizar las

áreas con mayor densidad de población, o aquellas con una proporción más alta de población adulta propensa a problemas de salud? ¿Qué tal si nos guiamos por las llamadas al servicio de emergencias, colocando las ambulancias donde más demanda haya? En este TFG trataremos de encontrar la mejor asignación estática de ambulancias para el área metropolitana de la ciudad de Valencia, aprovechando información sobre los vehículos de emergencia de los que dispone el SES, las posibles bases que pueden considerarse como estaciones y llamadas de emergencia. Concretamente, se trata de un proyecto de optimización, donde buscaremos minimizar el tiempo medio de respuesta que tardan las ambulancias en llegar al lugar de las emergencias.

Para afrontar este proyecto, hay que tener en cuenta la magnitud y complejidad del problema al que nos enfrentamos. Se ilustrará más adelante en detalle, pero debido a la cantidad de asignaciones de ambulancia posibles, es necesario utilizar métodos aproximados que nos den una buena solución, próxima a la óptima, en un tiempo razonable. Las técnicas que nos permiten lograr esto se denominan metaheurísticas, y el TFG se centrará en desarrollar modelos aprovechando sus principios, para encontrar la solución al problema de asignación de ambulancias.

Por último, este trabajo forma parte del ecosistema "iReves" *Innovación en Reubicación de Vehículos de Emergencias Sanitarias,* un proyecto de investigación valenciano cuyo propósito consiste en crear herramientas inteligentes capaces de brindar, de manera instantánea, información precisa a los encargados de los vehículos de emergencia sanitaria acerca de la ubicación óptima de los mismos en todo momento. En concreto, este proyecto se desarrolla dentro de la herramienta Reves Static desarrollada por Reves Mind. Reves Static aborda el desafío de la ubicación estática de los vehículos de emergencia sanitaria, donde cada vehículo tiene una base de partida y regreso fija durante sus turnos de trabajo. Sin embargo, pueden surgir contratiempos, como la falta de disponibilidad de vehículos o el aumento de emergencias, que requieren modificaciones temporales en estas ubicaciones estáticas, aunque puede no ser la forma más optima de gestionar el problema desde un punto de vista teórico, en la práctica es mucho más realista que las ambulancias formen parte de una estación y no se muevan de ella.

Reves Mind cuenta con tres herramientas más: Reves Dynamic, Reves Map y Reves Simula, las cuales se utilizan según la decisión que se deba tomar. Podemos ver un esquema de la organización de las herramientas de iReves en la Ilustración 1.

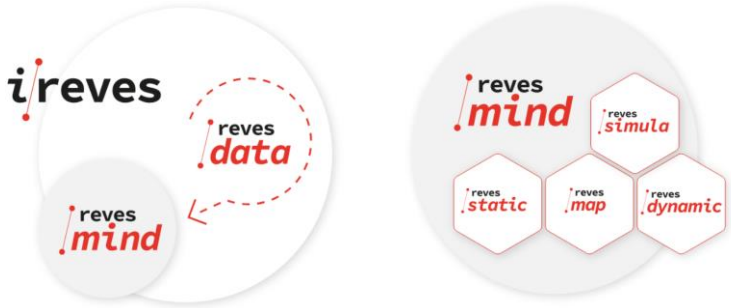

Ilustración 1. Herramientas dentro del proyecto iReves. Imagen obtenida de la página de iReves[1].

[1] https://ireves.webs.upv.es/

1.1. Motivación

Una de las grandes motivaciones de realizar este trabajo es la posibilidad de ayudar a la población de forma directa, en un tema tan importante como la sanidad. Si rebajamos los tiempos de respuesta a emergencias, podríamos llegar a tiempo a atender emergencias muy críticas, llegando incluso a salvar más vidas. También podríamos proporcionar una mayor cobertura y accesibilidad a los servicios de emergencia, si asignamos ambulancias en lugares que puedan estar más abandonados. Si lo visualizamos desde un punto de vista financiero, mejorar el sistema de ambulancias también sería interesante para el SES, ya que ahorraría en costos al tener que desplazarse menos distancia o al menos de forma más eficaz a los pacientes, pudiendo invertir ese dinero en mejorar otras facetas de la sanidad.

Además de los beneficios directos para la población y el sistema de salud, este trabajo también puede tener implicaciones socioeconómicas más amplias. Una mejor atención médica de emergencia puede tener un impacto positivo en la calidad de vida de las personas, en la confianza de la comunidad en el sistema de salud y en la percepción de bienestar general. También puede influir en la atracción de inversiones, turismo y desarrollo económico de la región, al ofrecer un sistema de emergencias sanitarias eficiente y confiable.

Por otra parte, los métodos que utilizaremos para optimizar la distribución de las ambulancias podrían tener aplicabilidad y utilidad en otras ciudades y contextos. Las poblaciones que busquen mejorar su sistema de emergencias sanitario, especialmente aquellas con similitudes demográficas con la ciudad de Valencia, podrían aprovechar los conceptos de nuestras estrategias y adaptarlos a sus necesidades. Cada ciudad tiene sus restricciones y sus recursos, pero las conclusiones a las que lleguemos podrían servir a diversas localidades como punto de partida para optimizar sus propios sistemas de servicios de emergencias.

1.2. Objetivos e Impacto Esperado.

El objetivo principal de esta investigación es diseñar y desarrollar modelos de optimización con técnicas metaheurísticas que sean capaces de obtener una distribución de ambulancias óptima o cercana a la óptima con respecto al tiempo de respuesta promedio a las emergencias, dentro del área metropolitana de Valencia.

Otro de los objetivos de este proyecto es proponer estos modelos de optimización como contribución al Estado del Arte, en el campo de la distribución de ambulancias. Dado el creciente interés y la necesidad de mejorar la eficiencia de los servicios de emergencia en las ciudades, el objetivo es compartir y difundir las técnicas desarrolladas en este trabajo, para que otras ciudades puedan adaptar y probar estas técnicas fácilmente.

En cuanto al impacto esperado, el mejor resultado posible sería crear algún modelo metaheurístico que tuviera un impacto real dentro del sistema de emergencias sanitarias de la comunidad Valenciana, ya fuera consiguiendo una distribución de los recursos directamente mejor que la actual, o proponiendo algún cambio que más tarde fuera aprobado. Los resultados obtenidos en este proyecto se podrían emplear para mejorar las estrategias aplicadas por el sistema valenciano de salud.

1.3. Metodología

Para poder obtener el tiempo medio de respuesta, nos aprovecharemos de un simulador diseñado para emergencias sanitarias. El funcionamiento se explicará más adelante en detalle, pero dado un conjunto de ambulancias y llamadas, y gracias también a información sobre las carreteras, el simulador es capaz de emular los movimientos de idas y venidas de los vehículos en función del orden y lugar de las llamadas. Esto permite obtener varias métricas al final del proceso, entre ellas el tiempo medio de respuesta.

La dificultad reside entonces en la elección de las distribuciones de ambulancias a probar, y es en este punto donde se centra nuestro proyecto. Desarrollaremos modelos metaheurísticos que actuarán como algoritmos de optimización y tendrán como función objetivo a optimizar la salida del simulador, minimizando el tiempo medio de respuesta.

El simulador desempeña un papel fundamental en este proyecto, ya que permitirá evaluar con precisión la calidad de las asignaciones propuestas. Sin embargo, es importante destacar que ya está creado y nosotros solo lo aprovecharemos. Nuestra contribución principal radica en los modelos metaheurísticos, que contendrán la estrategia y las reglas para proponer asignaciones de ambulancias, basándose en los resultados obtenidos a través del simulador.

Una vez hayamos diseñado los modelos y adaptado el simulador a nuestro problema, realizaremos experimentos con ellos, así como con otros de comparación, que nos servirán como base para evaluar los resultados. Un modelo puede ser mejor que otro si en promedio consigue mejores distribuciones de ambulancias, o si las encuentra en menos tiempo.

1.4. Estructura

El contenido del presente trabajo se organiza de la siguiente forma:

Este primer capítulo contiene la introducción del Trabajo Fin de Grado, la motivación, los objetivos que se persiguen y la metodología empleada para lograrlos, así como el impacto esperado del proyecto.

El capítulo 2 habla sobre el estado del arte. Concretamente, analizarán los sistemas de emergencias sanitarias y su optimización, abordando los sistemas estáticos, métodos de optimización, métricas, metaheurísticas y sistemas basados en simulaciones. También obtendremos una comprensión profunda de las estrategias clave para mejorar los sistemas de emergencias sanitarias.

En el capítulo 3 nos enfocaremos en el problema específico de optimización de emergencias sanitarias en el área metropolitana de Valencia. Analizaremos los datos de los que disponemos y los recursos del Servicio de Emergencias Sanitarias en Valencia y sus alrededores.

El capítulo 4 agrupa las herramientas informáticas que se han utilizado, tanto de software como de hardware. Aquí se explicarán en detalle estas herramientas. El hardware será especialmente importante para la experimentación, mientras que el software es fundamental para diseñar los modelos.

En el capítulo 5 se hablará en detalle del simulador, una de las partes clave de nuestro modelo, aunque su creación no es parte de este TFG.

El capítulo 6 consiste en el análisis del marco legal y ético del proyecto, comentaremos qué restricciones tenemos con nuestros datos. También de si es ético tomar decisiones sobre la salud algorítmicamente.

En el capítulo 7 nos centraremos en la modelización del problema, y presentaremos los modelos metaheurísticos que intentarán mejorar la asignación actual de ambulancias en el área metropolitana de Valencia. Después en el capítulo 8 continuaremos con la experimentación de estos modelos y los compararemos para averiguar si alguno tiene ventaja sobre otro.

Por último, el capítulo 9 contiene las conclusiones, el legado del proyecto, la relación con los estudios cursados, y las líneas de trabajo futuras que se podrían investigar para mejorar el proyecto.

Capítulo 2. Marco teórico

En este apartado, nos adentraremos en un análisis detallado del funcionamiento de los sistemas de emergencias sanitarias, centrándonos posteriormente en los sistemas estáticos. Además, exploraremos cómo se han aplicado métodos de optimización con el objetivo de mejorar estos sistemas, empleando diversas métricas como referencia. Por último, examinaremos las metaheurísticas y los sistemas de optimización basados en simulaciones, destacando su relevancia en este contexto. A lo largo de esta sección, obtendremos una comprensión más profunda de los elementos clave y las estrategias utilizadas para optimizar los sistemas de emergencias sanitarias, sentando las bases para el desarrollo del enfoque que queremos plantear.

2.1. Información sobre los sistemas de emergencias sanitarias

El sistema de emergencias sanitarias es una red compleja de órganos y recursos diseñados para brindar una respuesta rápida y efectiva ante situaciones críticas que ponen en peligro la vida y la salud de las personas. En esta red, diversos actores trabajan en conjunto para garantizar una atención médica oportuna y de calidad.

Uno de los órganos clave en este sistema es el centro de llamadas de emergencia, el punto de contacto inicial cuando se requiere asistencia médica urgente. Como referencia, en la Comunidad Valenciana las llamadas al 112 las gestionaría este órgano. Los operadores capacitados atienden las llamadas, recopilando información crucial sobre la situación y coordinando la respuesta adecuada. Estos centros son fundamentales para una gestión eficiente de los recursos y para asegurar que la ayuda llegue a tiempo. Los centros de llamadas están coordinados con los conductores de ambulancias, para avisar del lugar al que tienen que acudir y así tardar lo menos posible.

Para el transporte rápido y seguro de los pacientes, las ambulancias son recursos esenciales en el sistema de emergencias sanitarias. Estos vehículos están equipados con tecnología médica avanzada y son operados por personal capacitado en atención prehospitalaria. Además de su función de transporte, las ambulancias también pueden brindar atención médica en ruta, lo que es crucial para mantener la estabilidad del paciente durante el traslado. Las ambulancias normalmente son de varios tipos, de forma que unas responden a emergencias más graves y otras a las más leves, de esta forma, los recursos médicos se pueden dividir.

También hay otros vehículos de carretera con capacidad para responder a emergencias, pero sin capacidad de transporte de los pacientes. Estos vehículos actúan como intervención rápida a emergencias normalmente no tan graves, que principalmente pueden ser respondidas al momento, sin necesidad de acudir a un centro de salud. Además del transporte por carretera, hay sistemas de emergencias más complejos que también cuentan con otros vehículos, como helicópteros de rescate o barcos, dependiendo de la topología de la ciudad o las necesidades de la zona en la que esté desplegada el sistema de emergencias sanitarias. La Ilustración 2 muestra algunos de estos vehículos dentro de la Comunidad Valenciana.

Ilustración 2. Otros vehículos de emergencias sanitarias menos conocidos. A la izquierda, una aeronave de emergencias. A la derecha un vehículo de intervención rápida. Ambos vehículos son parte del sistema de emergencias de la Comunidad Valenciana. Fuente: Servicio de Emergencias Sanitarias de la Comunidad Valenciana (SESCV).

2.2. Problemas de optimización en sistemas de emergencias sanitarias

Uno de los aspectos fundamentales para optimizar un sistema de emergencias sanitarias es tomar decisiones acerca de la estructura que debe seguir. En general, estos problemas se pueden dividir en dos grupos principales.

En primer lugar, encontramos los sistemas estáticos, en los cuales las ambulancias tienen una base fija asignada de la cual obtienen y reponen recursos, y a la cual regresan después de atender a un paciente si no hay otro caso urgente por atender. La principal ventaja de estos sistemas radica en su facilidad de despliegue y configuración, ya que no necesitan ser redistribuidos en tiempo real a otras ubicaciones.

Por otro lado, están los sistemas dinámicos, en los cuales las ambulancias tienen una base inicial desde la cual parten, pero después de atender a un paciente pueden dirigirse a una estación diferente de la que salieron para esperar nuevas llamadas. Este enfoque también tiene sus ventajas, ya que cuando las ambulancias están ocupadas atendiendo a pacientes, pueden dejar zonas descubiertas al alejarse demasiado de su base. Así, las ambulancias pueden redistribuirse estratégicamente a lo largo del día, maximizando la cobertura y los tiempos de respuesta.

Además, los sistemas dinámicos también pueden plantearse como sistemas estáticos que se adaptan a lo largo de un mes o un año, es decir, tienen diferentes distribuciones dependiendo del momento. Por ejemplo, es posible que la distribución de la población varíe en épocas de verano, lo que requeriría una configuración distinta de las ambulancias en comparación con el invierno u otoño. Esta adaptabilidad permite ajustar el sistema a las necesidades cambiantes de la comunidad y garantizar una respuesta eficiente en todo momento. Al final los sistemas dinámicos son más complejos y teóricamente siempre serían mejor o al menos igual que un sistema estático, sin embargo, en la realidad son mucho más difíciles de llevar a cabo ya que requieren de comunicación a todas horas con las ambulancias, y también existe cierta resistencia a ser implementados por los propios trabajadores del servicio de emergencias, puesto que requiere que se muevan por diferentes estaciones dependiendo de la franja del día, el día, o el mes.

En el área metropolitana de Valencia está instaurado un sistema estático de emergencias sanitarias, es por ello que en este TFG nos centraremos más en este tipo de arquitectura.

2.3. Contribuciones en localización estática

Dentro de los problemas de localización estática, se puede buscar optimizar diferentes métricas que valoren cómo de buena es una distribución. Una de las primeras formas que surgieron para evaluar el sistema sanitario de emergencias fue intentar minimizar el número de ambulancias necesarias para cubrir unos puntos de demanda mínimos, (Constantine Toregas, 1971) a través de un concepto conocido como el recubrimiento. En el recubrimiento a cada ambulancia se le asigna una distancia o área, normalmente en forma de círculo, que llega a alcanzar, considerando esa zona como cubierta por el sistema de emergencias. Esta misma información también se puede utilizar para buscar la maximización del espacio total cubierto de un área dado una cantidad fija de ambulancias (ReVelle, 1974) . Fue a partir de estas investigaciones que se empezó a explotar la optimización en los servicios de emergencias, el uso del recubrimiento fue mejorando llegando a incluir aproximaciones sobre el tráfico para hacerlo más realista (John F. Repede, 1994) y a día de hoy con conceptos más modernos como las isócronas, que miden específicamente la distancia y las zonas que puede alcanzar un vehículo en un tiempo determinado, utilizando información sobre la red de carreteras (García Vecina, 2022) propuesto por investigadores de nuestra propia universidad, como vemos gracias al avance de la tecnología, podemos aproximar con mucha precisión el alcance de los vehículos.

Más recientemente, con la disponibilidad de datos geoespaciales y más específicos como la velocidad de los vehículos en cada carretera y las propias redes de carreteras (distancias exactas entre lugares) podemos minimizar el tiempo medio de respuesta de las ambulancias a las llamadas de emergencia directamente, una métrica mucho más interesante que el propio recubrimiento de zonas específicas, ya que se ha demostrado que el tiempo de respuesta es directamente proporcional con el ratio de supervivencia de los pacientes (Valerie J. De Maio, 2003).

Evaluar el tiempo de respuesta y el recubrimiento en el problema de asignación de ambulancias a estaciones son dos enfoques completamente diferentes. En el caso del recubrimiento, se puede determinar rápidamente la calidad de la distribución de las ambulancias. Sin embargo, al evaluar el tiempo de respuesta, se requiere simular los movimientos de las ambulancias hacia y desde las estaciones, comprender el sistema de toma de decisiones que siguen para responder a las llamadas de emergencia y, en muchos casos, contar con datos sobre la ubicación y el momento de dichas llamadas. No obstante, también existen modelos estocásticos que pueden tener en cuenta el tiempo de respuesta sin depender exclusivamente de datos precisos sobre la ubicación y el momento de las llamadas, considerando factores como la densidad de población, la disponibilidad de ambulancias y las características geográficas para modelar la dinámica del sistema y generar resultados aproximados del tiempo que tardaría la ambulancia en llegar. Algunos autores en Shanghái (Lu Zhen, 2014) o en Turquía, (Nuşin Uncu, 2022) han optimizado el tiempo medio de respuesta, utilizando aproximaciones matemáticas, consiguiendo mejorarlo en este último caso en más del 40% en zonas urbanas y rurales.

En otros estudios también se ha medido con el tiempo máximo de viaje, es decir, en lugar de minimizar la media, se intenta minimizar que el paciente más lejano esté lo más cerca posible, (Mohammad Maleki, 2014) aunque esta forma de abordar el problema puede no ser muy robusta y estar influenciada por datos anómalos.

Otro de los problemas del tiempo medio de respuesta es que es difícil comparar resultados entre ciudades, lo que se considera rápido en un lugar puede ser lento en otro, especialmente si tenemos en cuenta sólo área metropolitana o toda la provincia. En Eskisehir, Turquía, una ciudad con 570k habitantes, el tiempo de respuesta era de 6m30s y consiguió ser rebajado a 4m (Tugba Sarac, 2014), mientras que en Londres el objetivo es intentar bajar de los 12 minutos (Richard McCormack, 2015)

2.4. Algoritmos metaheurísticos

Los metaheurísticos son métodos aproximados que guían el proceso de búsqueda de soluciones con el objetivo de explorar las posibilidades eficientemente. A diferencia de los algoritmos exactos, que encuentran la solución óptima si se dispone del tiempo y recursos suficientes, los algoritmos metaheurísticos se enfocan en encontrar soluciones satisfactorias en un tiempo razonable, siendo especialmente útiles cuando el espacio de búsqueda es muy grande.

Una de las ventajas de los algoritmos metaheurísticos es su capacidad para escapar de óptimos locales y explorar diferentes áreas del espacio de búsqueda. Utilizan estrategias heurísticas que permiten saltar entre soluciones y realizar cambios en la búsqueda con el fin de encontrar mejores resultados. Esto los hace flexibles y adaptativos a diferentes problemas y condiciones.

En los sistemas de emergencia, el espacio de búsqueda de soluciones puede llegar a ser realmente complejo. Por ejemplo, puede darse el caso que moviendo tres ambulancias de lugar al mismo tiempo se mejore significativamente el tiempo de respuesta, pero si movemos sólo una de ellas, empeore. En este sentido los algoritmos metaheurísticos pueden llegar a ser la clave para optimizar sistemas de emergencia.

Dentro de los algoritmos metaheurísticos, dos de los más utilizados para optimizar distribuciones de ambulancias son los algoritmos genéticos y la búsqueda tabú. Cada uno cuenta con características que permiten abordar eficazmente los problemas a los que se enfrenta la optimización de sistemas de emergencias.

2.4.1. Algoritmos genéticos

Los algoritmos genéticos (Holland, 1975) fueron introducidos por primera vez hace casi 50 años. Estos algoritmos se basan en la teoría de la evolución y los principios genéticos de selección natural y reproducción, y se utilizan para abordar problemas de optimización, siguiendo un proceso esquemático en el cual una población de individuos evoluciona con el tiempo, manteniendo los individuos más aptos.

El proceso de los algoritmos genéticos comienza generando una población inicial compuesta por un número determinado de individuos, cada uno representando una posible solución (por ejemplo, diferentes distribuciones de ambulancias). Estas soluciones son evaluadas en función del objetivo de optimización, como el tiempo de respuesta promedio para cada distribución de ambulancias. La bondad de una solución se denomina "fitness" de la solución, un nombre que se utiliza específicamente en estos algoritmos.

A continuación, los individuos mejor evaluados se reproducen mediante la operación de cruce, que combina los genes o los valores de dos individuos para generar descendencia. Esta operación se realiza con la premisa de que los buenos individuos deberían ser buenos padres, transmitiendo sus características favorables a las generaciones futuras. Por otra parte, a los individuos ya sean padres o hijos resultantes, se les puede aplicar la operación de mutación, que introduce cambios aleatorios en los individuos para explorar nuevas soluciones y evitar el estancamiento en óptimos locales.

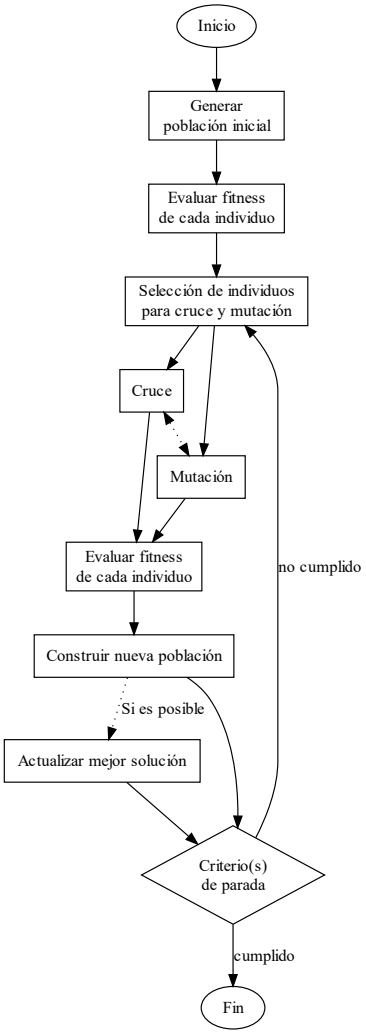

Ilustración 3. Esquema de un algoritmo genético general. El cruce y la mutación se pueden dar al mismo tiempo, o uno tras otro. Elaboración propia con Graphviz[2].

[2] Software de código abierto para representación de diagramas de flujo https://graphviz.org/

Después de la reproducción y la mutación, la población resultante se evalúa nuevamente y solo se conservan los individuos más aptos, en este sentido, nos quedaríamos con las mejores distribuciones de ambulancias, según nuestra métrica. Este proceso de selección, reproducción y mutación se repite durante varias generaciones, permitiendo que la población evolucione hacia soluciones óptimas. La Ilustración 3 muestra el esquema que por lo general siguen estos algoritmos, aunque pueden llegar a ser mucho más complejos.

Los algoritmos genéticos brindan varias ventajas en la optimización de sistemas de emergencias, ya que su naturaleza iterativa y su capacidad para explorar un amplio espacio de soluciones permiten encontrar resultados satisfactorios. Además, su flexibilidad y aplicabilidad a diferentes problemas en diversos campos los convierten en una herramienta versátil.

La estrategia de los algoritmos genéticos ha sido utilizada muchas veces para optimizar sistemas de emergencias debido a su facilidad para implementar información ad-hoc y en general por su versatilidad, se ha utilizado para maximizar el espacio cubierto por ambulancias (Haldun Aytug, 2002), en un conjunto de datos controlado donde se conocía la asignación de ambulancias ideal, demostrando su utilidad en problemas de recubrimiento y capacidad para encontrar soluciones casi óptimas incluso la óptima, en un corto período de tiempo. En la propia ciudad de Delhi, (Zaheeruddin, 2021) también se ha utilizado, demostrando que el tiempo medio de respuesta puede reducirse en un 4.35% empleando la misma cantidad de ambulancias. En otro caso un algoritmo genético se utilizó para sugerir la cantidad de ambulancias que se debían utilizar, llegando a la conclusión de que era necesario añadir 9 ambulancias; y para encontrar mejoras en el sistema de emergencias sanitario, teniendo en cuenta los costes que supondría y buscando un balance en varias carreteras de Brasil. (Ana Paula Iannoni, 2008).

2.4.2. Búsqueda tabú

La búsqueda tabú fue introducida en 1986 (Glover, Future Paths for Integer Programming and Links to Artificial Intelligence, 1986), como una forma de superar las limitaciones de otros métodos de búsqueda tradicionales, a la vez que se mencionaba por primera vez el término metaheurística. Más tarde el propio autor escribiría un libro para que sirviera de referencia de los principios de la búsqueda tabú (Glover, Tabu Search, 1997)

La idea principal detrás de la búsqueda tabú es explorar el espacio de soluciones de manera inteligente, evitando quedar atrapado en óptimos locales y buscando activamente soluciones más prometedoras. Para lograr esto, la búsqueda tabú mantiene un conjunto de soluciones visitadas previamente en una lista llamada "lista tabú". Estas soluciones visitadas se consideran "tabú" y se evita volver a ellas durante un cierto período de tiempo o número de iteraciones. El proceso de búsqueda tabú comienza con una solución inicial, que puede ser generada de forma aleatoria o mediante otro algoritmo. A partir de esta solución inicial, se exploran los vecinos, que son soluciones similares a la solución actual con pequeñas modificaciones. Estos vecinos se generan aplicando movimientos específicos al estado actual, como intercambios, inserciones o eliminaciones. Por ejemplo, un vecino de una cierta asignación de ambulancias podría ser otra asignación, donde simplemente una ambulancia estuviera cambiada de posición.

La búsqueda tabú tiene una característica distintiva en comparación con otros enfoques de búsqueda local: permite movimientos que pueden empeorar la solución actual a corto plazo, pero que tienen el potencial de conducir a una solución mejor en el futuro. Estos movimientos "tabú" se utilizan para evitar quedarse estancado en óptimos locales y explorar diferentes regiones del espacio de búsqueda. La capacidad que tiene para escapar de óptimos locales y explorar de manera inteligente el espacio de soluciones lo convierte en una herramienta valiosa para abordar problemas difíciles y mejorar los resultados obtenidos con otros enfoques de búsqueda, lo cual es especialmente importante en el problema de la optimización de ambulancias.

La Ilustración 4 muestra cómo funciona por dentro un modelo de búsqueda tabú. Generalmente son menos complejos que un algoritmo genético, pero pueden complicarse dependiendo del problema que queramos resolver.

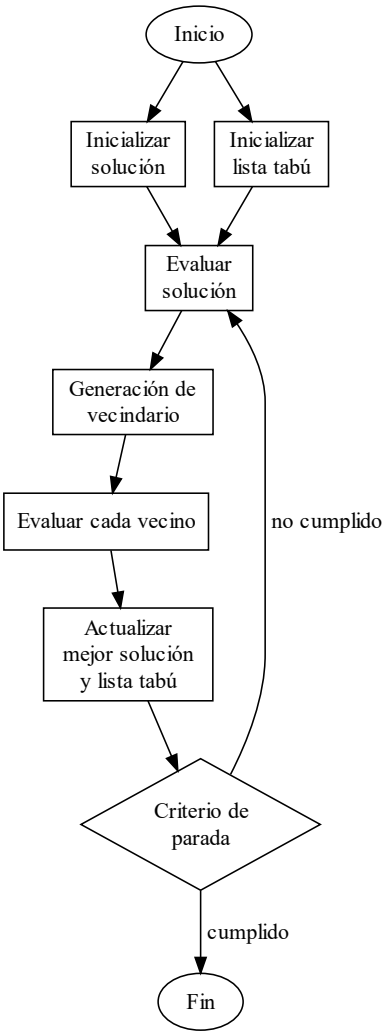

Ilustración 4. Esquema general de una búsqueda tabú. Elaboración propia con Graphviz.

Podemos encontrar un ejemplo de búsqueda tabú en problemas de emergencias sanitarias en (Joseph Tassone, 2020) donde se utilizó para optimizar la asignación de un conjunto de ambulancias aéreas, obteniendo mejores resultados en el mismo tiempo de ejecución que con un modelo sencillo de búsqueda local, esto es, sin utilizar la lista tabú. También se ha utilizado en

problemas de recubrimiento, con el objetivo de minimizar la cantidad de ambulancias necesarias para cubrir al menos 2 veces ciertas zonas de demanda (Michel Gendreau, 1997).

2.5. Software de simulación de emergencias sanitarias.

Una de las mejores formas de incorporar datos precisos sobre el problema en los modelos es utilizando simulaciones, ya que aprovechan la capacidad de modelar diferentes escenarios y variables. Por ejemplo, dada una cierta distribución de ambulancias y un conjunto de llamadas, un simulador; que contaría con un método de "*dispatching*" o decisión de envío de ambulancias, y un mapa de carreteras como el que vemos en Google Maps, sería capaz de calcular exactamente el tiempo de respuesta promedio a las emergencias.

Las simulaciones pueden ser deterministas, si no incluye efectos aleatorios, es decir que cada vez que lanzamos una simulación con los mismos datos de entrada el resultado será el mismo, o estocástica, si dejamos efectos aleatorios. Esto último puede ser útil en los servicios de emergencia sanitarios, ya que podríamos aleatorizar el tráfico o el origen de las llamadas, factores que muchas veces son difíciles de determinar, sin embargo, esto quiere decir que para evaluar cómo de buena es una distribución, deberíamos hacer varias simulaciones con los mismos datos de entrada, dependiendo de cantidad de distribuciones posible y la velocidad de simulación, el coste temporal de la optimización en general puede escalar muy rápido.

Los simuladores se suelen utilizar como la función a optimizar, es decir, dadas las variables que podamos incorporar (distribución de ambulancias, origen de las llamadas, tráfico, mapa de carreteras…) el simulador devuelve el tiempo de respuesta promedio a las emergencias. Esto permite combinarlos con los métodos de resolución vistos anteriormente. Los algoritmos genéticos y la búsqueda tabú actúan como métodos inteligentes de exploración de soluciones, mientras que el simulador proporciona el método por el que evaluamos cada solución.

Muchos simuladores se han utilizado para evaluar la bondad de una distribución de ambulancias, algunos son BartSim, utilizado en Auckland (Henderson, 2000), para respaldar la decisión de abrir una nueva base de ambulancias en el norte de la ciudad, así como proponer nuevos cambios en la ubicación de las bases y la cantidad de médicos en cada estación; Tifar Modelling en Ámsterdam (Van Buuren, 2012), utilizado para simular escenarios realistas a partir de parámetros obtenidos de datos históricos y probar distintas configuraciones y políticas de reubicación, o EMSSim (Moon, 2015), proporcionando una simulación completa de los servicios médicos de urgencia durante algunas catástrofes, desde el rescate de las víctimas en el lugar de lo ocurrido hasta su tratamiento definitivo en los hospitales. Recientemente, se ha diseñado JEMSS (Julia package for Emergency Medical Services Simulation) (Samuel Ridler, 2022), un simulador determinista pensado para emergencias sanitarias. La clave de JEMSS es que ha sido diseñado observando los otros simuladores, centrándose en la facilidad de uso y la velocidad. JEMSS además está escrito en Julia, un lenguaje de programación ideal para realizar las simulaciones ya que puede llegar a ser tan rápido[3] como C++, a la vez que mantiene la

[3] https://github.com/niklas-heer/speed-comparison

facilidad de uso de Python, el lenguaje principal que utilizamos en el Grado en Ciencia de Datos.

Este simulador es el que utilizaremos para la experimentación, y que enlazaremos con nuestros métodos metaheurísticos. Se necesitarán algunas modificaciones para poder hacer la conexión que se estudiarán en el Capítulo 5. El simulador. Las razones que han llevado a la decisión de utilizar este emulador se han explicado anteriormente, es nuevo, de código abierto, y su facilidad de uso y rapidez, características clave ya que vamos a necesitar simular cientos de miles de distribuciones.

Capítulo 3. Descripción del Problema

En este capítulo analizaremos nuestro problema concreto de optimización de emergencias sanitarias, centrándonos específicamente en el área metropolitana de Valencia. Veremos los datos de los que disponemos, así como los recursos con los que cuenta el Servicio de Emergencias Sanitarias en Valencia y alrededores.

3.1. Sistema de Emergencias del área metropolitana de Valencia

3.1.1. Tipos de ambulancias

En el área metropolitana de Valencia, desde 1992 están implantados los vehículos de Servicio de Atención Médica Urgente (SAMU), y los de Soporte Vital Básico (SVB) entre otros. Los recursos que lleva cada vehículo los podemos encontrar en su página web[4], y los que interesan para este proyecto los resumimos en la Tabla 1.

Tabla 1.Vehículos de respuesta a las emergencias en el SESCV. Información obtenida de la web del sistema de emergencias sanitarias de la Comunidad Valenciana.

VEHÍCULO	DOTACIÓN
SAMU	Médico/a SAMU, Enfermería SAMU, Técnico/a en Emergencias Sanitarias
Soporte Vital Básico (SVB)	Dos Técnicos/as en Emergencias Sanitarias
Soporte Vital Avanzado (SVA)	Enfermero/a SAMU y dos Técnicos/as en Emergencias Sanitarias

En el área metropolitana de Valencia, tanto los vehículos SAMU como los SVA brindan atención a las mismas emergencias. Por lo tanto, en este Trabajo Fin de Grado se considerarán equivalentes, denominándolos ambos SVA de acuerdo con el modelo valenciano. Los SVB, encargados de las emergencias más leves, también se incluirán en el modelo, y al conjunto de todos los vehículos los llamaremos Vehículos de Emergencia Sanitaria o VES en adelante. Podemos ver cómo son estos vehículos en la Ilustración 5.

Ilustración 5. Vehículos de emergencia activos en el área metropolitana de Valencia. A la izquierda, una ambulancia de soporte vital básico. A la derecha una ambulancia de soporte vital avanzado. Fuente: Servicio de Emergencias Sanitarias de la Comunidad Valenciana (SESCV).

[4] https://ses.san.gva.es/es/descripcio-de-recursos

Dentro del área de la comunidad valenciana que vamos a tratar nosotros, disponemos de 10 ambulancias SVA y 19 ambulancias SVB. Los dos tipos de ambulancias están en constante comunicación con el servicio de emergencias, de manera que una vez han tratado con el paciente y están volviendo a la base, pueden volver a ser llamados por otra emergencia directamente, agilizando la respuesta.

En el área metropolitana de valencia, las ambulancias actualmente están distribuidas según laIlustración 6.

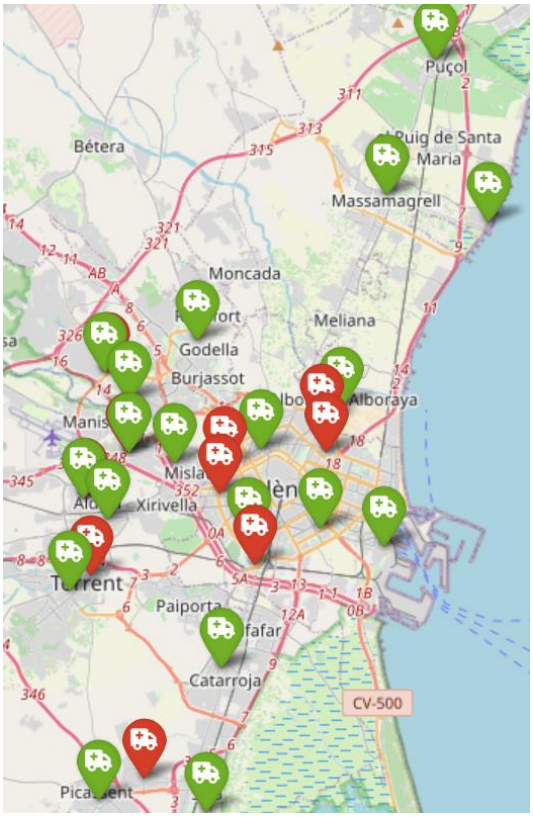

Ilustración 6. Distribución actual de las ambulancias SVA (rojo) y SVB (verde) en el área metropolitana. Elaboración propia con Python (folium).

Las ambulancias están bastante distribuidas a lo largo del área metropolitana, existiendo algunas estaciones con 2 ambulancias en Aldaia, Quart y Paterna. Sorprende la falta de SVA hacia el norte de Valencia, las SVB parecen estar más dispersadas.

3.1.2. Estaciones sanitarias

Las estaciones sanitarias engloban todas las zonas desde los que pueden salir y regresar ambulancias en el área metropolitana de valencia, habiendo en total 277. Estos datos han sido proporcionados por el sistema de emergencias sanitarias, pero realmente son datos abiertos, ya que está formado por todos los Centros de Salud, Centros de Rehabilitación, Consultorios auxiliares u Hospitales de la zona. La Ilustración 7 muestra algunos ejemplos de centros sanitarios.

De cada estación tenemos información sobre su nombre, dirección y coordenadas, que se utilizarán para situarlas en la red de carreteras de forma precisa. Para este proyecto, hemos decido poner como límite que las estaciones sanitaras pueden albergar como máximo 2

ambulancias, que pueden ser en forma de 2 del mismo tipo o una de cada. Aunque hay lugares, como hospitales, donde teóricamente se podrían situar muchas más ambulancias, poner 2 como límite permite no saturar un lugar con demasiados recursos. Igualmente, debido a la gran cantidad de estaciones que hay en comparación con las ambulancias, es prácticamente innecesario subir el límite. Podemos visualizar todas las estaciones posibles en el mapa de la Ilustración 8.

Ilustración 7. El Hospital Universitario y Politécnico "La Fe" de Valencia (izquierda), y el centro de salud de Valencia Tres Forques (derecha), son algunos de los lugares considerados como estaciones. Imágenes extraídas de los periódicos El Meridiano/ El Levante.

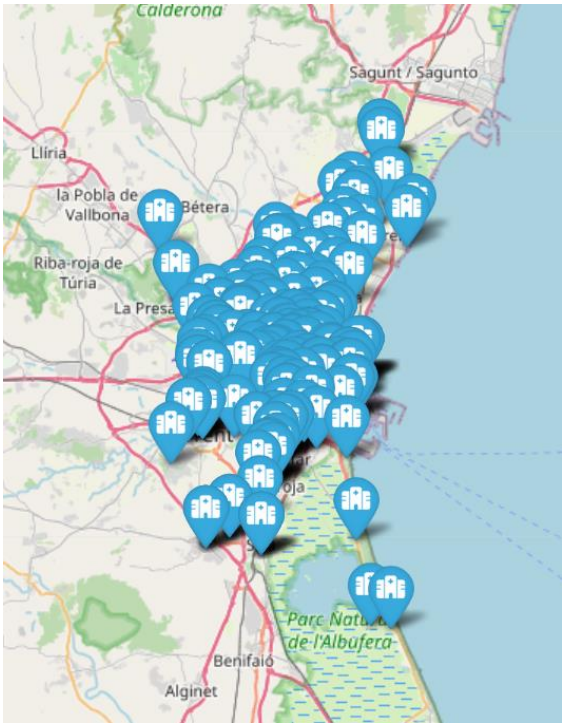

Ilustración 8. Representación de las posiciones de las estaciones en el área metropolitana de valencia. Elaboración propia con Python (folium).

Hay estaciones en prácticamente todos los barrios y ciudades de valencia y la zona metropolitana, llegando a los limites en la Albufera, Silla, Torrent, La Presa y Puçol.

3.1.3. Red de carreteras

La red de carreteras no es específica para los problemas de emergencias sanitarias, pero permiten incluir la información precisa sobre las distancias entre puntos de la ciudad y la

velocidad a la que puede ir cada vehículo, si utilizamos un simulador para la optimización. La red de carreteras de toda el área metropolitana de valencia se ha obtenido a partir de ORS[5] (Open Route Service). Este servicio es de código abierto y cualquiera puede visitar la página y obtener un mapa de carreteras de una zona de cualquier parte del mundo.

3.1.4. Llamadas a los servicios de emergencia.

El CICU (Centro de información y coordinación de urgencias) es el que recibe las llamadas del 112 y decide qué vehículo y recursos enviar en base al problema del paciente. En nuestro caso, el conjunto de datos que vamos a utilizar ha sido generado en base a conocimiento experto, dentro del proyecto iReves para trabajos de investigación, mediante datos abiertos como la densidad de población en los municipios del área metropolitana, o la edad de sus ciudadanos. En concreto, el conjunto de datos está generado con datos desde el 1 de enero de 2019 hasta el 31 de diciembre de ese mismo año.

De cada llamada tenemos información sobre el momento exacto del día en el que se hizo y las coordenadas desde las que se hizo, además de otros datos como cuánto tarda en salir la ambulancia desde que se recibe la llamada (*dispatch delay*), que en este caso siempre es alrededor de 50s y cuánto tiempo estuvo la ambulancia donde ocurrió la emergencia (*onSceneDuration*), representado en la Ilustración 9. Esta información nos permite saber con gran precisión cómo se ha gestionado cada emergencia. También contamos con qué tipo de ambulancia se encargó de la emergencia.

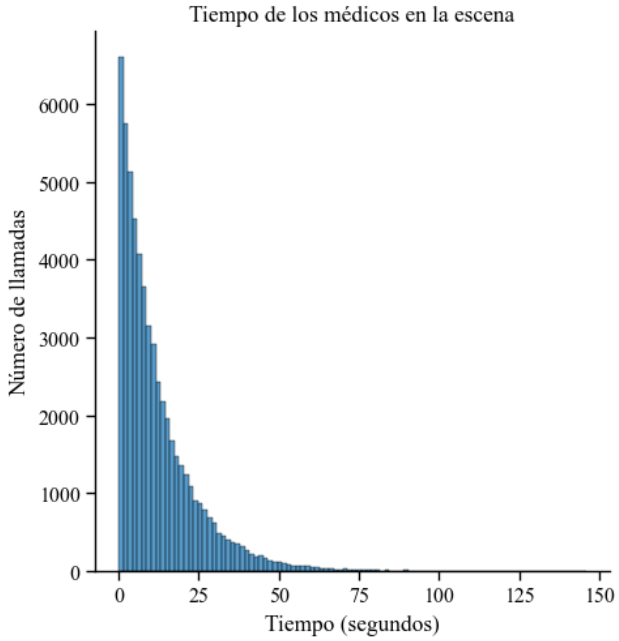

Ilustración 9. Distribución del tiempo que pasan las ambulancias en el lugar del paciente. Hay algunos que necesitan incluso más de 50 minutos de atención. Elaboración propia con Python (seaborn).

En total se generaron 57599 llamadas a emergencias que han ocurrido en 2019. Siendo noviembre el mes con más llamadas, y agosto el mes con menos llamadas. Curiosamente, en verano es cuando menos llamadas hay al 112, lo cual puede deberse a que muchos ciudadanos

[5] https://openrouteservice.org/

aprovechan para salir e irse de vacaciones. Aunque también es cierto que valencia tiene una gran cantidad de turismo en esta época. La distribución de llamadas por mes puede visualizarse en la Ilustración 10.

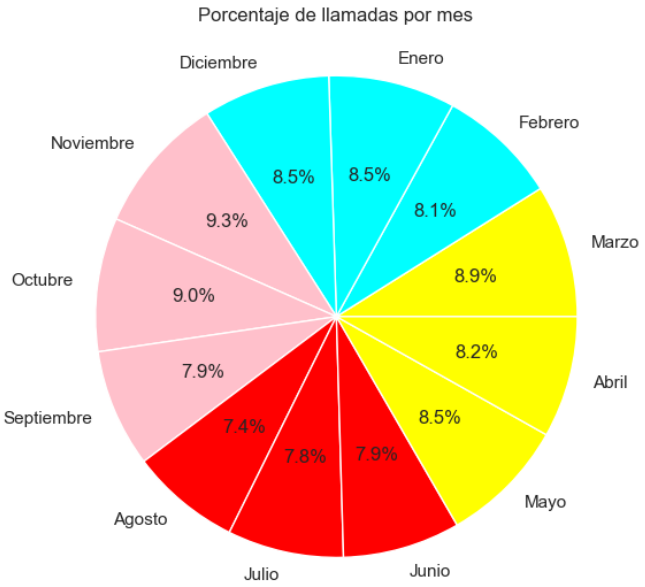

Ilustración 10. Porcentaje de las llamadas por mes, coloreado por trimestre (invierno, primavera, verano, otoño). Elaboración propia con Python (seaborn).

A cada llamada se le asigna un nivel de prioridad, siendo 1 "alta prioridad" y 3 "baja prioridad" dada las necesidades del paciente. La mayoría de ellos eventualmente necesitan ser transportados por alguna urgencia, y sorprende un poco ver cómo hay más porcentaje de personas de baja prioridad que necesitan ser transportadas al hospital, pero esto puede deberse a que en las ambulancias de tipo 1 (SVA) hay un médico dentro del vehículo, que muchas veces evita tener que desplazar al individuo. Esta información se puede ver resumida en la Ilustración 11.

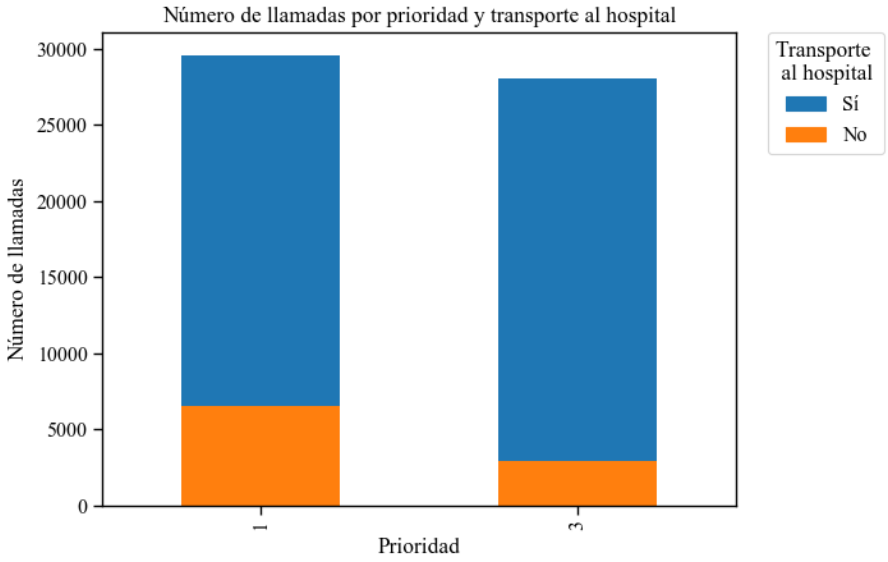

Ilustración 11. Necesidad de transportar al paciente al hospital, disgregado por tipo de ambulancia. Elaboración propia con Python (seaborn).

Las llamadas se distribuyen de forma similar a lo largo de toda el área metropolitana, centrando núcleos tanto en la propia ciudad de Valencia como en Torrent, como se puede ver en los mapas de calor de la Ilustración 12.

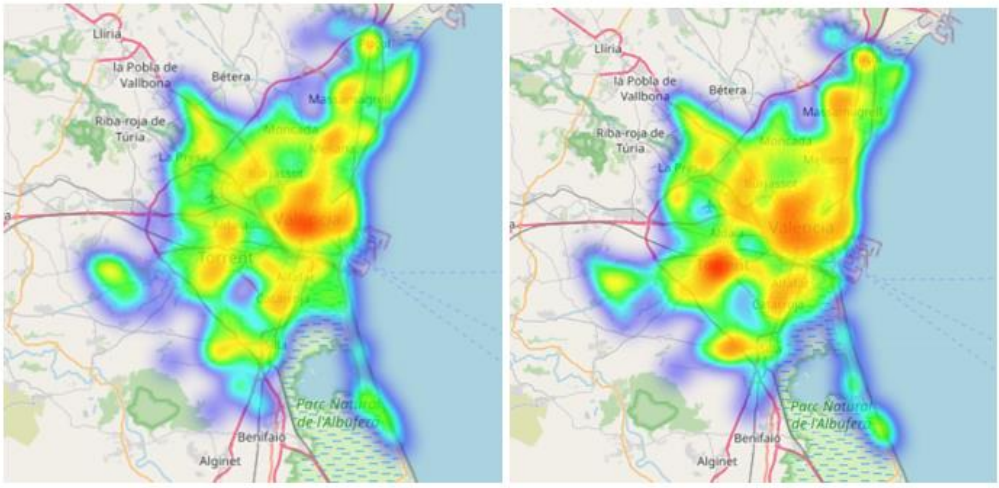

Ilustración 12. Distribución de llamadas en la provincia. Llamadas de tipo 1 (emergencias graves) // Llamadas de tipo 3 (emergencias leves). Valencia centro y Torrent son los 2 grandes núcleos de llamadas. Elaboración propia con Python (folium).

3.2. Análisis de la complejidad del problema

Es inevitable sentir curiosidad por la cantidad de formas que hay de distribuir las ambulancias a lo largo del área metropolitana de Valencia. Con 29 ambulancias para colocar y 277 posibles estaciones donde caben 2 ambulancias como máximo, el número debe de ser muy grande. Si bien el problema es complejo, y no se puede solucionar utilizando combinatoria directamente debido a las restricciones que plantea, podemos calcular una aproximación si suponemos que las ambulancias son del mismo tipo, utilizando una función generatriz.

Una función generatriz es una forma de codificar una secuencia infinita de números tratándolos como los coeficientes de una serie de potencias. Se trata de un concepto poderoso en combinatoria, el estudio del recuento y la ordenación de objetos. Los conceptos de las funciones generatrices fueron introducidos por Euler en el siglo XVIII, por lo que no podemos adjuntar un artículo moderno como referencia, pero hay libros (Wilf, 2005) centrados en la aplicación de estas funciones.

En este caso, podemos utilizarla para contar el número de formas de distribuir "k" objetos idénticos (ambulancias) entre "n" grupos distintos (estaciones) con un máximo de "m" artículos por grupo (ambulancias en estaciones). La función generadora de este problema viene dada por:

$$(1 + x\ +\ x^2 \ldots x^m)^n = (1 + \mathrm{x} + \mathrm{x}^2)^{277}$$

Ecuación 1. Función generatriz (definición) y aplicada en el caso de las 277 estaciones.

Podemos visualizarlo mejor si tuviéramos 5 estaciones en las que distribuir 3 ambulancias con un máximo de 2 en cada estación, en cual caso la función sería $(1 + x + x^2)^5$. La cantidad

de distribuciones sería el valor del coeficiente "k" que acompañe x^3 al expandir esta serie, que se puede ver desglosada en la Ecuación 2.

$$x^{10} + 5x^9 + 15x^8 + 30x^7 + 45x^6 + 51x^5 + 45x^4 + \mathbf{30x^3} + 15x^2 + 5x + 1$$

Ecuación 2. Función generatriz expandida en un caso de 5 estaciones y 2 ambulancias como máximo por estación.

Por lo que habría 30 distribuciones.

Volviendo a nuestro caso, deberíamos encontrar el coeficiente que acompañe a x^{29}, ya que tenemos 29 ambulancias en total, en la expansión de $(1 + x + x^2)^{277}$. Este desglose es muy complejo y para resolverlo se puede utilizar un programa matemático como WolframAlpha[6]. El resumen del resultado se puede encontrar en la Ecuación 3.

$$x^{554} + 277x^{553} + \cdots + \mathbf{25374202921332972801167894442376908532786}x^{29} + \cdots + 38503x^2 + 277x + 1$$

Ecuación 3. Expansión de la Ecuación 1 (caso de 277 estaciones)

Un valor aproximado a $2.537 * 10^{40}$. ¿Cómo de grande es este número? Es difícil de visualizar. El número estimado de granos de arena en la Tierra es de unos $7.5 * 10^{18}$, si cada grano de arena representara una Tierra entera, con su propio conjunto de granos de arena, necesitaríamos más de mil millones de estas "Tierras de arena" para aproximarnos a la cantidad de distribuciones posibles en Valencia.

[6] https://www.wolframalpha.com/input?i=expand+%281%2Bx%2Bx%5E2%29%5E277+&lang=es

Capítulo 4. Recursos utilizados

Para llevar a cabo este TFG y a lo largo de él ha sido necesario utilizar herramientas informáticas específicas, tanto de software como de hardware, y dedicaremos un capítulo un poco más técnico para explicarlas. Algunos de los conocimientos necesarios para el manejo de estos recursos fueron obtenidos a lo largo del Grado en Ciencia de Datos, y otros se tuvieron que investigar y aprender para la realización del proyecto.

4.1. Software

Es fundamental comprender que esto es un problema de optimización, y estamos minimizando el resultado de un simulador, que, dada una distribución de ambulancias y un conjunto de llamadas, devuelve el tiempo de respuesta promedio a las emergencias. Este simulador está escrito y desarrollado en Julia (Jeff Bezanson, 2012), un lenguaje de programación que surgió en 2012 con características como facilidad de uso similar a Python, pero con la velocidad y eficiencia algorítmica de C/C++. Un lenguaje de uso general pero pensado para análisis matemático y ciencia computacional, en concreto métodos algorítmicos e investigación operativa. Estas características hacen a Julia un lenguaje de programación ideal para diseñar no solo simuladores de eventos, si no nuestros modelos metaheurísticos, que además podrán aprovechar el *multithreading* de Julia, fácil de implementar.

La versión de Julia que se ha utilizado es la 1.6.7, por su compatibilidad con JEMSS, la librería del simulador. La mayoría del código se ha escrito utilizando las funciones bases de Julia, y solo se han necesitado algunas librerías:

- FileIO / DelimitedFiles para acceder a los ficheros CSV de las llamadas
- Random para aleatorizar varios procesos, como la selección de días de llamadas dentro de las metaheurísticas (lo veremos más adelante)
- JEMSS para lanzar las simulaciones
- TreeParzen[7] para la optimización de los hiperparámetros

Para que el simulador funcione, es necesario además instalar el solver Gurobi[8] desde su página web, el propio JEMSS te dirigirá a ella si intentas instalarlo sin este solver, y también te indicará que versiones son compatibles. Aunque no se profundizará en este tema, Gurobi es uno de los *solvers* (herramientas diseñadas para encontrar soluciones óptimas a problemas de optimización) más utilizados en investigación operativa ya que cuenta con modelos "estado del arte" y funcionan de forma muy eficiente y con conjuntos de datos muy grandes, además de su compatibilidad con lenguajes de programación. En este caso, Gurobi es gratis para investigadores, alumnos y profesores.

Además de Julia, hemos utilizado Python para otras tareas secundarias, como realizar el análisis exploratorio de los datos, visualizar los resultados y comunicarnos con la API de Open Route Service. Aunque Julia también es capaz de llevar a cabo estas funciones, en nuestro caso,

[7] https://github.com/IQVIA-ML/TreeParzen.jl
[8] https://www.gurobi.com/documentation/quickstart.html

hemos recibido una formación más extensa en Python a lo largo de nuestra carrera, y en para este caso específico, podríamos decir que es mejor lenguaje ya que tiene muchas librerías que hacen fácil lo que queremos conseguir, al igual que Julia lo era para realizar la optimización.

La versión de Python que se ha utilizado es la 3.9.13, también por su compatibilidad con las librerías que necesitábamos utilizar, y que en concreto han sido estas:

- Pandas[9] en el manejo de CSVs como tablas de datos
- Numpy[10] para algunas funciones matemáticas
- Folium[11]: Una librería para hacer visualizar el mapa de valencia, con mapas de calor de las llamadas o distribuciones de ambulancias.
- Matplotlib/seaborn[12] para la realización de gráficos
- Openrouteservice[13]: Librería para conectar con la API de ORS
- Scipy[14] para cálculos estadísticos en los resultados

En el desarrollo de este proyecto, hemos utilizado Git[15] en combinación con GitLab[16] para mantener el código del proyecto en un repositorio y facilitar la conexión con la máquina virtual donde se realizaron los experimentos. Git es una herramienta de control de versiones que permite a múltiples colaboradores mantenerse sincronizados en un proyecto, mientras cada uno tiene una copia local del trabajo. En nuestro caso, hemos utilizado GitLab como plataforma para crear repositorios privados de código.

Para escribir y editar el código en el entorno local, hemos utilizado Visual Studio Code[17], que ofrece una fácil integración con repositorios de GitLab. Una de las ventajas adicionales de Visual Studio es su capacidad para localizar rápidamente el origen de las funciones, lo que resulta especialmente útil al trabajar con una biblioteca extensa como JEMSS ya que hemos necesitado editar y comprender partes del código fuente. En general utilizar VSCode como editor de código ha agilizado mucho el trabajo.

Otro de los softwares que han ayudado en la realización de este proyecto ha sido Lucidchart[18], una página para construir diagramas de flujo y representar información de forma sencilla, aplicación que ya vimos en algunas asignaturas de la carrera; y Overleaf[19] un editor de LaTeX online para escribir algunos algoritmos en pseudocódigo, de forma que estén resumidos en lenguaje natural.

[9] https://pandas.pydata.org/
[10] https://numpy.org/doc/stable/
[11] https://python-visualization.github.io/folium/
[12] https://seaborn.pydata.org/
[13] https://openrouteservice-py.readthedocs.io/en/latest/
[14] https://docs.scipy.org/doc/scipy/
[15] https://git-scm.com/
[16] https://about.gitlab.com/
[17] https://code.visualstudio.com/
[18] https://lucid.app/documents
[19] https://es.overleaf.com/

4.2. Hardware

Utilizar tantos datos para conseguir simulaciones realistas tiene un coste y era mayor de el que esperábamos, el ordenador donde desarrollamos el código era capaz de lanzar simulaciones, pero neutralizaba el PC durante alrededor de media hora, utilizando todos los recursos y necesitando escribir datos en disco. Era posible, pero inviable, es por ello que necesitamos utilizar una máquina virtual para realizar toda la parte de la experimentación, y esta máquina fue cedida por el DSIC (Departamento de Sistemas Informáticos y Computación de la UPV) exclusivamente para la realización de este proyecto.

Es importante tener en cuenta los recursos de los que dispone esta máquina virtual, en caso de que alguien quiera replicar los experimentos. Para dejar constancia de todas las herramientas de las que ha dispuesto el proyecto, y del nivel de recursos que ha necesitado, se listará aquí las características de la máquina virtual:

- Sistema Operativo Linux bigmem 4.15.0-20-generic #21-Ubuntu SMP 2018 x86_64 x86_64 x86_64 GNU/Linux
- Procesadores: 16 cores Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz
- Memoria: 31 GiB de RAM

Capítulo 5. El simulador

Como hemos visto, vamos a utilizar JEMSS como simulador en este TFG, una librería de Julia que está preparada para simular emergencias sanitarias con alto nivel de detalle. Se trata de la primera librería de código abierto para emergencias sanitarias, desde el paquete EMS[20] escrito en Python, aunque JEMSS es superior a este, incorporando la red de carreteras (en lugar de viajes en línea recta), gran capacidad de visualización de la simulación y velocidad por el lenguaje de programación.

Según JEMSS podemos realizar simulaciones de 100 días en menos de 1 segundo, nosotros hemos lanzado experimentos con varias cantidades de días para entender cuánto le cuesta con nuestros datos. Los resultados se pueden ver en la Ilustración 13.

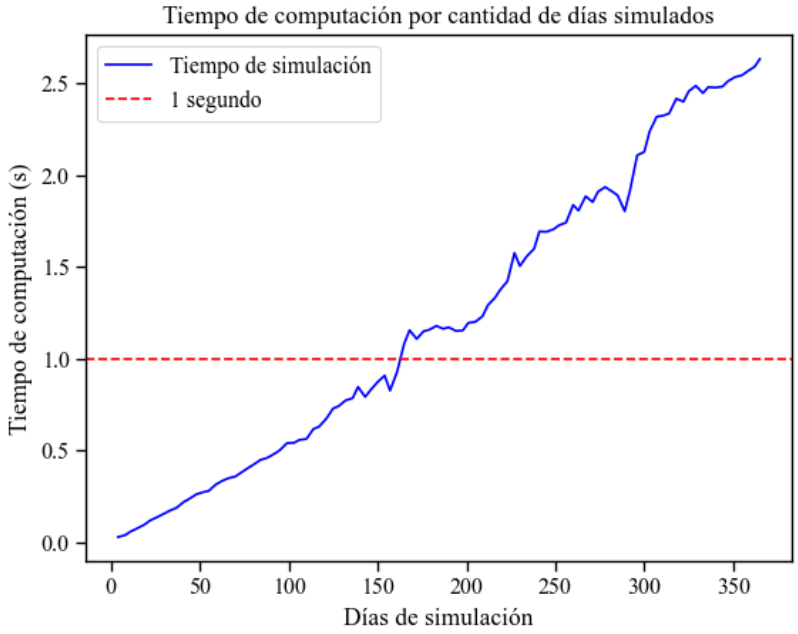

Ilustración 13. Evolución del tiempo de cómputo de una simulación en el área metropolitana de Valencia por días simulados. A mayor cantidad de días, más lento es el proceso de simulación. Elaboración propia con Python (seaborn).

Debido a que quizá Valencia no es una ciudad con tantas emergencias como puede ser una capital como Madrid o Paris, es capaz de simular más de 100 días en 1 segundo, ya que realmente depende de la cantidad de llamadas que se realicen. Concretamente en 1 segundo es capaz de realizar un poco más de 150 días, por otra parte, simular todo el año cuesta alrededor de 2.6 segundos. Cada cantidad de días fue simulada 30 veces y promediada, para que los resultados fueran un poco robustos.

El simulador situó el tiempo medio de respuesta, de la asignación actual de ambulancias en el área metropolitana de Valencia, en 4.07 minutos. Este valor es importante porque servirá de comparativa para los modelos y los resultados que obtengamos más adelante.

[20] https://github.com/eotles/EMS

El sistema de envío o “dispatching” que sigue tanto JEMSS como el sistema de emergencias del área metropolitana es el mismo. La clave del funcionamiento es, que una vez la ambulancia finaliza de atender a un paciente, ya sea tras llevarlo al hospital o no, se envía directamente a por otro. Esto puede ocurrir directamente si hay llamadas de emergencia en la cola, o si ocurre una mientras la ambulancia vuelve a su base. La Ilustración 14 muestra este sistema de dispatching de las ambulancias.

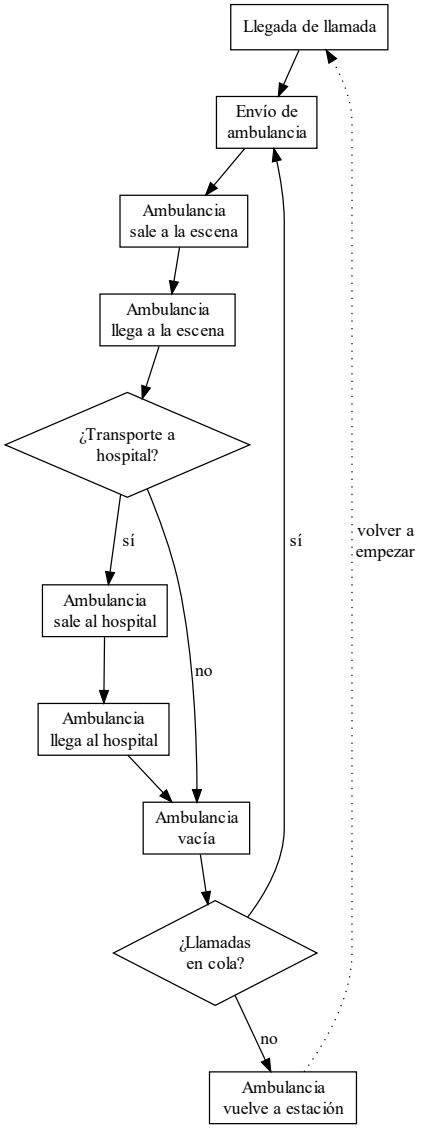

Ilustración 14. Sistema de dispatching de ambulancias de JEMSS y del sistema de emergencias implantado en el área metropolitana de Valencia. Elaboración propia con Graphviz.

5.1. Funcionamiento del simulador

El proceso de simulación podría dividirse en 2 partes:

1. **La carga de los datos**, consistiendo en volcar la red de carreteras, nodos y arcos, las llamadas, las estaciones y las ambulancias, consiguiéndolo a partir de un fichero que contiene las referencias a cada uno de los archivos que necesita. A partir de una función se crea un objeto de tipo “simulación”. El objeto simulación ahora se queda en estado

"cargado" es decir, listo para simular y calcular las métricas, como el tiempo medio de respuesta. Podemos ver una representación de este proceso en la Ilustración 15.

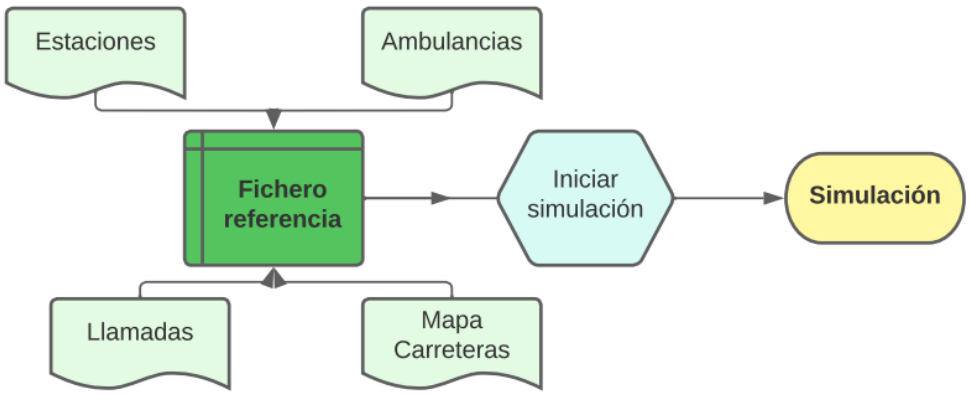

Ilustración 15. Carga de un objeto simulación en JEMSS. Elaboración propia con Lucidchart.

2. **La simulación**, los objetos simulación "cargados" se pueden simular para directamente obtener los resultados, o se puede ejecutar otra función para visualizar una animación de los resultados, de forma que se abre un navegador y se muestra gráficamente cómo se mueven las ambulancias a lo largo de la simulación por la ciudad, tiene mucho detalle y se puede apreciar como las ambulancias de tipo 1 son más rápidas que las de tipo 2, o cómo algunas carreteras son más rápidas que otras. También se puede ver dónde se encuentra el paciente en cada momento. Una vez se ejecuta la simulación se queda en estado "descargada", es decir, no puede volver a simularse. Si queremos volver a hacerlo debemos reiniciar la simulación o crear otra desde 0. El proceso de simulación se puede visualizar en la Ilustración 16.

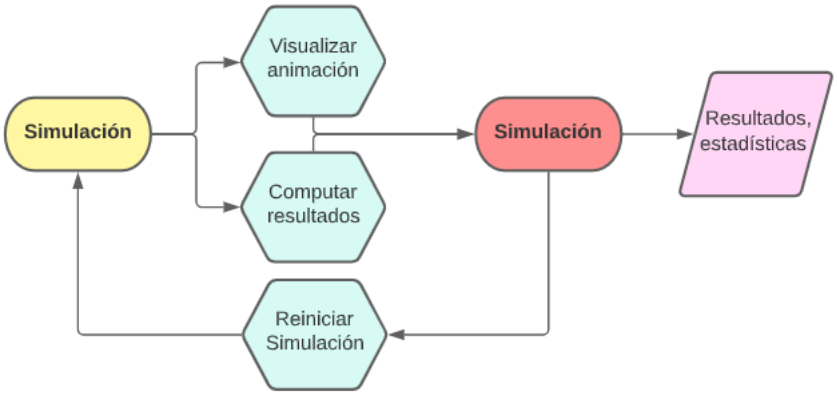

Ilustración 16. Descarga de un objeto simulación. Al finalizar se pueden obtener diferentes datos y estadísticas. Elaboración propia con Lucidchart.

La visualización de los resultados es muy interesante, aunque en este trabajo solo utilizaremos la función para obtener los resultados ya que es más rápida y nos devuelve lo que necesitamos igualmente.

Es importante diferenciar ambas partes, la carga de datos, en nuestro caso y dada la red de carreteras, tarda alrededor de 74s. El proceso de simulación depende de cuántos días de

llamadas tengamos en cuenta, pero como vimos en la Ilustración 13, alrededor de 2.6s para todo el año, y 0.007s para sólo 1 día.

5.2. Cambios en el simulador

Es aquí donde vemos la necesidad de incluir cambios en el simulador, ya que, si queremos construir un algoritmo genético o utilizar otras metaheurísticas, necesitamos simular miles de combinaciones de ambulancias. Una de las claves que hará esto posible será la creación de una nueva función, que a partir de una simulación en estado "cargado", una lista de ambulancias y de llamadas, permitirá crear nuevos objetos simulación instantáneamente, ya que no necesitarán volver a cargar el fichero de la red de carreteras, el principal causante de que sea tan lento, ya que toma el 97.5% del tiempo de la carga de los datos total. Esta función debe escribirse dentro del módulo *run_config* de JEMSS, y se puede encontrar en el Anexo, Ilustración 43, por si alguien quisiera emplearla en sus experimentos.

Además, dentro del propio simulador será necesario exportar otras funciones que de normal están ocultas para el usuario, y que crean internamente las ambulancias y el conjunto de llamadas, para después ser pasadas a la nueva función de simulación. Estas funciones se utilizarán en los métodos metaheurísticos para acelerar el proceso y eliminar la dependencia de ficheros csv o xml. Las funciones en concreto han sido: *initAmbulance!*,*addEvent!* e *initSim2* (nueva función de simulación).

Con los cambios que hemos incluido permitimos que, a partir de una simulación de referencia, con las carreteras y las estaciones ya incluidas, pueda crear múltiples simulaciones sin tener que reiniciarse ni empezar de 0, esto agiliza el proceso en unos 73.99 segundos, ya que generar simulaciones pasa a tardar alrededor de 0.01s, aunque el proceso de simulación en sí sigue tardando el mismo tiempo de antes. De esta forma, el nuevo modelo nos permite realizar simulaciones siguiendo el esquema de la Ilustración 17.

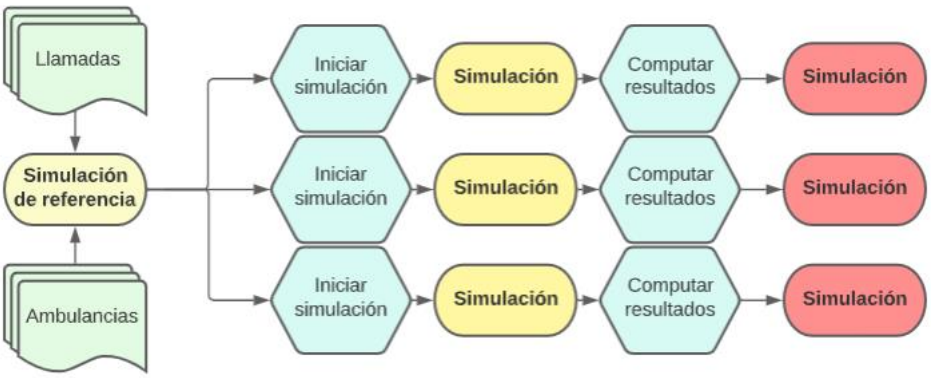

Ilustración 17. Proceso de emulación actualizado tras los cambios en el simulador. El nuevo método nos permite crear simulaciones con diferentes ambulancias y llamadas instantáneamente. Elaboración propia con Lucidchart.

Más adelante en el apartado 7.1. Modelización del problema, veremos cómo transformamos los datos para que el enlace del proceso de simulación a las metaheurísticas fluya sin problemas.

Capítulo 6. Análisis del Problema

6.1. Análisis del marco legal y ético

En este trabajo se utilizan datos abiertos del Servicio de Emergencias Sanitarias de la Comunidad Valenciana (SESCV), así como de otras fuentes. El tratamiento de estos datos se lleva a cabo cumpliendo con el marco legal establecido, en particular, con el Reglamento General de Protección de Datos (RGPD) de la Unión Europea.

De acuerdo con el RGPD, se deben tomar medidas para proteger los datos personales y garantizar la privacidad de los individuos. En el caso de las ambulancias, la información sobre las SVAs y SVB es de carácter público y se puede encontrar en la página web del SESCV. En cuanto a la divulgación de la posición exacta de las ambulancias, esta información no presenta ningún problema, ya que no permite identificar a ningún individuo.

En cuanto a los datos de las llamadas, es importante destacar que se han generado utilizando conocimiento experto y datos públicos, como la densidad de población. Esto significa que no estamos utilizando datos privados sensibles. Desde el inicio del proyecto, se han implementado medidas de anonimización para proteger la identidad de los pacientes y cumplir con los principios de minimización de información establecidos en el artículo 5.1.c del RGPD.

Gracias a que las llamadas no son reales, no contamos con información sensible, como el número de teléfono o la duración de la llamada. Además, en este proyecto en particular, se suprimió información del conjunto de datos original, como la razón de la llamada, dado que no es relevante para la investigación. Los datos médicos son además uno de los principales identificadores indirectos de las personas y es por ello que se considera información sensible bajo el artículo 9 del RGPD.

A pesar de que las coordenadas de las llamadas han sido generadas aleatoriamente, se han aplicado técnicas de desplazamiento aleatorio para dificultar una posible identificación de la persona que realizó la llamada, ya que los datos sociodemográficos también están consideradores posibles identificadores indirectos. De esta forma, ni siquiera nosotros sabremos exactamente los orígenes de las llamadas.

Por otro lado, la red de carreteras de la Comunidad Valenciana utilizada en este proyecto se basa en información abierta obtenida de Open Route Service, lo que significa que no se trata de datos personales y no entra en el ámbito de aplicación del RGPD.

¿Es ético tomar este tipo de decisiones algorítmicamente? Al final uno de los objetivos de este proyecto es proponer una nueva distribución de ambulancias para el área metropolitana de valencia, y lo queremos hacer en base a lo que nos dicen los datos. Aunque estamos utilizando mucha información para hacer los experimentos lo más realistas posibles, sería imprudente dejar las vidas de las personas en manos de lo que diga un algoritmo, al menos directamente. Estamos seguros de que este proyecto puede ayudarnos a minimizar el tiempo de respuesta a las emergencias, pero deberíamos combinar los resultados que obtengamos con conocimiento de expertos: desde alcaldes de los ayuntamientos de los pueblos y ciudades hasta los propios

médicos. Por ejemplo, puede ser que “idealmente” necesitemos colocar 3 ambulancias en Torrent, pero puede que no cuente con suficientes médicos que vivan cerca, y además esto implica que, si la ambulancia estaba en otra ciudad, el médico se quede sin ese trabajo.

Antes de aplicar cualquier cambio sobre el sistema de ambulancias, se debería comprobar que se pudiera llevar a cabo en la realidad, además, se debería probar durante un periodo de tiempo y posiblemente comparar con los resultados que tenemos ahora para ver si hay mejoras. Es un proceso que puede empezar desde la teoría y las matemáticas como es este TFG, pero es solo eso, nosotros podemos proponer ideas y otro grupo de expertos debería evaluarlas.

6.2. Identificación y análisis de soluciones posibles

Dada la amplia cantidad de estaciones posibles en la zona metropolitana y la limitada cantidad de ambulancias disponibles para su distribución, el número de combinaciones posibles de soluciones es prácticamente infinito. Por lo tanto, en este Trabajo de Fin de Grado (TFG), se opta por utilizar técnicas metaheurísticas que se acerquen lo máximo posible a la solución ideal. En la literatura, una técnica ampliamente utilizada para abordar este tipo de problemas es la búsqueda tabú, pero esta tiene un problema, y es que depende en gran medida de la solución inicial. Debido al alto número de vecinos que tiene cada una de nuestras soluciones, proponemos en este trabajo también el uso de un algoritmo genético. Esta elección nos permite aprovechar el conocimiento ad-hoc sobre el problema, adaptarlo específicamente a nuestro caso y reducir la dependencia del grupo de soluciones inicial.

La búsqueda tabú y los algoritmos genéticos serán las técnicas metaheurísticas utilizadas para resolver nuestro problema, además con el objetivo es evaluar y comparar la efectividad de cada método, también se probará la búsqueda aleatoria. Aspiramos a proponer un enfoque que sea altamente eficaz para abordar el desafiante problema de la gestión de ambulancias, y que pueda ser considerado un "estado del arte" en este campo. Para respaldar esta propuesta, aprovecharemos el simulador JEMSS, el cual busca establecerse como una referencia para abordar la optimización de la distribución de ambulancias.

Capítulo 7. Soluciones implementadas

En este capítulo, abordaremos la descripción del proceso de modelización del problema, la implementación de los datos en los métodos metaheurísticos y la construcción de dichos modelos.

7.1. Modelización del problema

Una de las cosas que es consistente en nuestras soluciones es el modo de representación de una solución. En la literatura se suelen utilizar representaciones vectoriales donde hay un vector de longitud N, y x[n] representa la cantidad de ambulancias en la estación n, normalmente 1 o 0. Pocas veces hay más de 1 tipo de ambulancia, en ese caso se pueden utilizar tuplas en lugar de valores numéricos en cada posición del vector.

Si repasamos las restricciones generales:

- (R1) La cantidad de ambulancias de tipo 1 (SVA) debe ser 10
- (R2) La cantidad de ambulancias de tipo 2 (SVB) debe ser 19
- (R3) No puede haber más de 2 ambulancias en total en una misma estación

Vemos que tenemos tanto restricciones propias de cada tipo como conjuntas. Julia maneja operaciones vectoriales de forma muy eficiente, por lo que en lugar de utilizar tuplas, decidimos implementar 2 vectores de longitud 277, que en Julia podemos representar como una matriz M(2, 277), de 2 filas y 277 columnas donde:

$$\text{(R1)} \sum_{i=1}^{277} M[1,i] == 10$$

$$\text{(R2)} \sum_{i=1}^{277} M[2,i] == 19$$

$$\text{(R3)} \sum_{i=1}^{2} M[i,j] == 2, \forall j \in \{1 \dots 277\}$$

Esta es la representación que se utilizará para trabajar con las ambulancias durante los algoritmos, sin embargo, a la hora de calcular la bondad de la solución, deberemos transformar cada ambulancia a su objeto *Ambulance*. A continuación, hablaremos de cómo hemos modelizado los datos para incluirlos en el simulador y en los métodos metaheurísticos, en forma de código.

7.1.1. Ambulancias

Podemos representar un individuo (distribución de ambulancias) como 2 listas de longitud 10 y longitud 19, respectivamente, donde cada elemento sea el número de la estación donde se coloca la ambulancia de ese tipo, en este caso el orden de las estaciones no influye ya que todas

las ambulancias de cada tipo son iguales. Esta representación hace muy fácil generar individuos aleatorios:

$$Individual = [(x_1, x_2, \dots, x_{10}), (y_1, y_2, \dots, y_{19})] \; / \; \forall i \in \{1,2, \dots 19\}, x_i, y_i \in \{1,2, \dots ,277\}$$

Este conjunto de números en forma de tupla nos servirá para crear las ambulancias más adelante, ya que lo único que necesita JEMSS de cada ambulancia es el tipo y la estación donde debe colocarse. Sin embargo, para las técnicas metaheurísticas utilizaremos la representación matricial como hemos visto antes, ya que permite comprobar las restricciones, obtener vecindarios y realizar las operaciones de los algoritmos genéticos más fácilmente.

Para mostrar ejemplos de distribuciones de ambulancias y con el objetivo de simplificar el problema para que sea más sencillo de entender, supondremos que hay 5 estaciones donde queremos colocar 3 ambulancias SVA y 3 ambulancias SVB, esto se mantendrá a lo largo del TFG, ya que no es factible realizar ejemplos con 277 estaciones.

La Ilustración 18 nos ayuda a entender mejor la representación matricial que se utilizará en los modelos de optimización. Hemos ilustrado un ejemplo, si hubiera una asignación de ambulancias SVA en las estaciones [1,2,5] y ambulancias SVB en [4,3,5], la matriz sería [1 1 0 0 1]; [0 0 1 1 1].

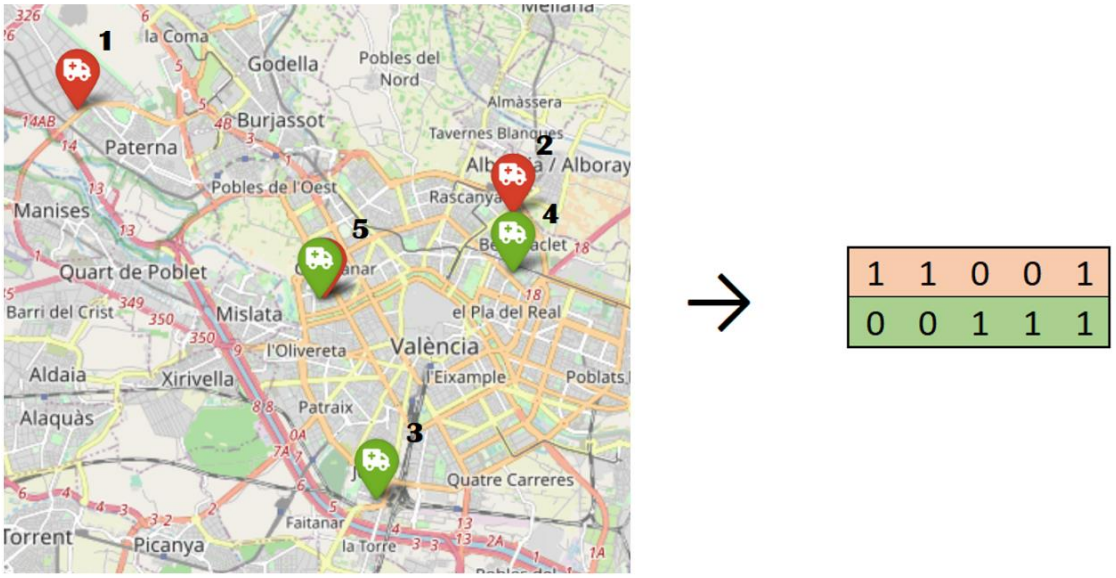

Ilustración 18. Transformación de la posición natural de las ambulancias a la representación matricial. Elaboración propia con Python (folium), y Excel.

Ahora es fácil asegurarse de que el individuo es factible comprobando las 3 restricciones iniciales. Después internamente, se utiliza una función para convertir la representación matricial en objetos ambulancia para la simulación la cual, ha sido extraída del código fuente de JEMSS.

7.1.2. Llamadas

Las llamadas se organizan en archivos, existiendo un fichero por cada día, un fichero por cada mes, y un fichero con todas las llamadas del año. Para crear nuestros objetos llamada haremos 2 grupos, por una parte, un grupo de llamadas "final" que contendrá todas las llamadas y que servirá para evaluar definitivamente la bondad de un individuo, por otra parte, un grupo

con el conjunto de llamadas individuales de cada día del año, la combinación de varios días se utilizará como fitness / tiempo de respuesta "intermedio" de un individuo, como veremos más adelante. Esto se hace para incrementar la velocidad de las simulaciones, ya que hemos visto antes que 1 año tardaba 2.6s en simular.

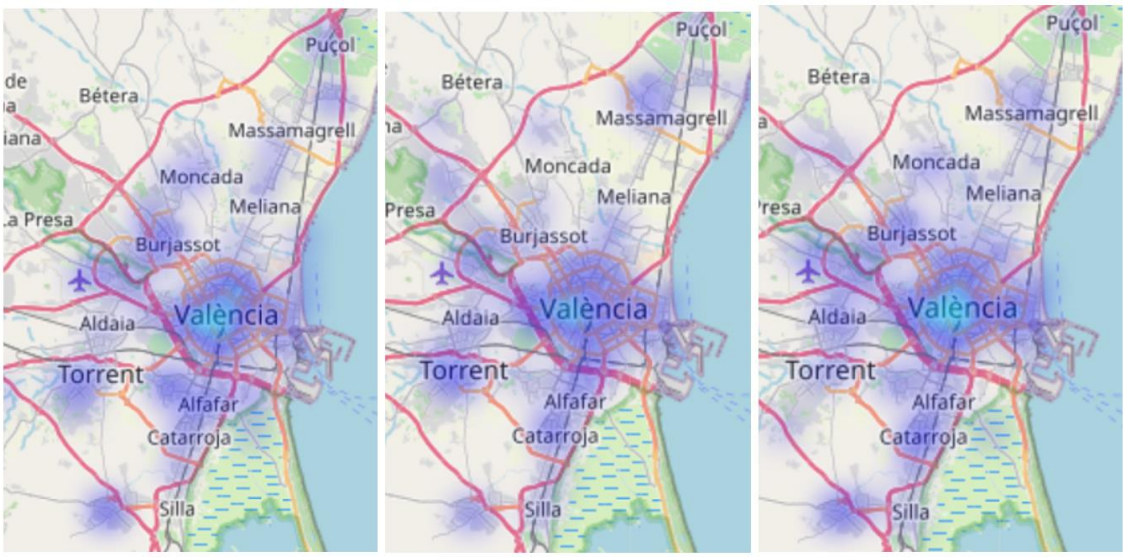

Ilustración 19. Distribución de las llamadas de los días 2 de junio (izquierda), 10 de enero (centro) y 28 de octubre (derecha) de 2019 en el área metropolitana de Valencia. Elaboración propia con Python (folium).

Creemos que es inteligente utilizar un conjunto aleatorio de días en lugar de todo el año para acelerar el proceso de simulación. Además, se puede observar en la Ilustración 19 que la distribución de las llamadas es similar para días aleatorios.

Al igual que con las ambulancias, las llamadas se incluyen en la simulación dentro de nuestro propio código, para no tener que hacerlo desde un fichero cada vez. Para esto hemos leído el fichero con todas las llamadas 1 vez y el de cada día también 1 vez, nos hemos guardado las llamadas en una lista y las vamos reutilizando.

7.1.3. Tiempo medio de respuesta y fitness

El "fitness" o bondad de la solución se utiliza en algoritmos genéticos y se podría definir como lo bien que responde un individuo/solución al problema a optimizar. En otras metaheurísticas como la búsqueda tabú también está el concepto de función objetivo a minimizar/maximizar, que sería algo similar. En nuestro caso, el fitness siempre se va a referir al tiempo medio de respuesta de las ambulancias a un conjunto de llamadas.

La evaluación de nuestro fitness se podría simplificar como la suma del tiempo de respuesta promedio de cada día, dividido entre el número de días totales que se empleen.

$$AvgRespTime = \frac{RespTime}{numCalls}$$

Como vimos en el esquema del nuevo proceso de simulación, una vez se tienen las llamadas y la distribución que se va a utilizar, se genera la nueva simulación a partir de la original, se calcula el tiempo de respuesta promedio a partir de las estadísticas de las llamadas y

se devuelve la media de todos los días que se hayan simulado. En este proceso se emplea *multithreading*, ya que cada día se puede simular independientemente de los demás.

7.2. Solución Algoritmo Genético "Estado-del-arte"

Encontrar una solución "estado del arte" es subjetivo, cada grupo de investigadores plantea una forma de abordar el problema dependiendo de los datos que tenga, la cantidad de ambulancias y estaciones, o cómo se quiere que sea el sistema de gestión de ambulancias que se quiera implementar. Los algoritmos genéticos además son muy personalizables y es difícil encontrar uno óptimo que pueda funcionar de forma general.

En esta parte nuestro objetivo era encontrar un algoritmo genético que haya funcionado en alguna ciudad y probarla nosotros mismos, debido al nivel de la revista en el que está publicado, por ser relativamente nuevo (2014), y por la similitud con nuestro trabajo, decidimos implementar este. En este caso, los investigadores también generaron la posición y el momento de las llamadas a través de datos públicos, en concreto con una distribución de poisson, una técnica común cuando no se cuenta con estos datos como vimos en el estado del arte.

Propiedades de este algoritmo genético:

- Inicialización de la población aleatoria de 20 individuos
- Los 15% (3) mejores individuos pasan a la siguiente iteración directamente
- Los 70% (14) siguientes mejores individuos reciben "1-point crossover", creándose 7 parejas aleatorias
- Los 15% (3) peores individuos reciben mutación
- "Normalización": Después del cruce es posible que el cromosoma no sea factible, en este caso se ajustan los valores para que lo sea, disminuyendo o aumentando los valores.
- El algoritmo termina cuando no mejora en 100 iteraciones sucesivas.

7.2.1. Función de mutación

La función de mutación es una operación específica de los algoritmos evolutivos para introducir diversidad genética en la población, que consiste en alterar aleatoriamente la información genética o las características de un individuo. La implementación exacta de esta función y como todas las del algoritmo genético depende de lo que se quiera conseguir y cómo sea el propio fenotipo, generalmente mediante una pequeña modificación aleatoria en el código genético.

En este problema, los autores deciden implementar una función de mutación que simplemente varía la posición de una ambulancia, es decir, dada una ambulancia de un tipo aleatorio, se coloca en una estación diferente en la que estaba. El individuo resultante podría considerarse un "vecino" del otro dentro del espacio de búsqueda, ya que es prácticamente idéntico.

Para comprender mejor el funcionamiento, hemos diseñado un ejemplo con la Ilustración 20. Dado un fenotipo P = [1,0,2,0,0]; [1,0,0,0,2], el resultado de aplicar mutación, suponiendo que el valor aleatorio "k" que se escoge es 6 (1º de las ambulancias SVB), y la estación para moverla "j" resulta ser la 10, sería P* = [1,0,2,0,0]; [0,0,0,0,3].

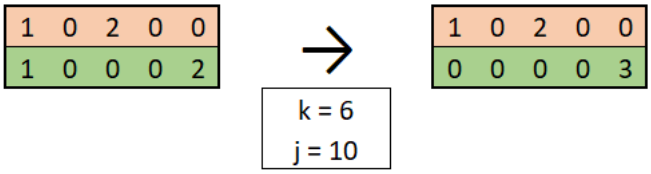

Ilustración 20. Ejemplo de mutación del algoritmo genético estado del arte. Elaboración propia con Excel.

Después de la mutación es posible que el fenotipo ya no cumpla con las restricciones planteadas inicialmente, por ejemplo, en este caso la estación 5 tiene 3 estaciones del mismo tipo, cuando el límite que hay es 2. Este problema se resolverá más adelante con la función de normalización.

Esta función de mutación se aplica a los 3 peores individuos de la población, es posible que simplemente cambiando una ambulancia de lugar la distribución mejore considerablemente, por ejemplo, porque se mueve a alguna zona donde no había ninguna de ese tipo cerca.

7.2.3. Función de cruce

El cruce o "*crossover*" es otro de los métodos propios de los algoritmos evolutivos, que consiste en una combinación de los genes de 2 (o más) de los fenotipos para crear otro individuo. Este proceso intenta simular el proceso de reproducción que sigue la naturaleza, bajo la teoría de que buenos individuos deberían resultar a su vez en buenos hijos para la población.

Una de las técnicas de cruce más comunes se llama "1-point crossover" y es la que utilizan los autores en este modelo. El cruce por un punto consiste en dividir cada padre en 2 a partir de un punto aleatorio, combinando la primera mitad del padre 1 con la segunda mitad del padre 2 y viceversa. Es muy fácil de implementar y además hace que la estructura y el orden relativo de los genes se conserve, lo cual puede ser bueno en problemas donde el orden de los genes es importante, como en este. Para la comprensión de su funcionamiento se ha diseñado un ejemplo, con la Ilustración 21.

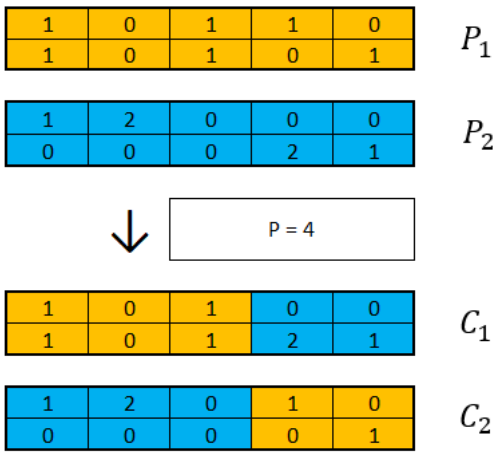

Ilustración 21. Ejemplo de cruce por un punto. Elaboración propia con Excel.

Si imaginamos un padre P1 = [1 0 1 1 0]; [1 0 1 0 1] y P2 = [1 2 0 0 0]; [0 0 0 2 1] y se escoge aleatoriamente el punto P = 4, el primer hijo resultante será C1 = [1 0 1 0 0]; [1 0 1 2 1] y el segundo, C2 = [1 2 0 1 0]; [0 0 0 0 1].

Después de la operación de cruce al igual que de la mutación, es probable que alguno de los hijos no sea factible ya que tiene más o le sobran ambulancias. Esta operación se ejecuta desde el 4º mejor individuo hasta el 17º, eligiendo pares aleatoriamente y formando en total 7 parejas.

7.2.2. Función de normalización

La función de normalización o *"fixing"* (arreglo/corrección) debe ser un método del algoritmo genético que dado un individuo no factible lo convierta en factible. Esto se hace con el objetivo de no mantener individuos no factibles dentro de la población, aunque es posible dejarlos siempre que decidamos cómo evaluar su fitness, por ejemplo, con alguna función de penalización, y siempre que el individuo final (en nuestro caso la distribución de ambulancias final) sea factible. Los autores que implementaron este algoritmo genético decidieron utilizar una función de normalización.

Los individuos no factibles pueden aparecer dentro de la población después de realizar cruce o mutación, si no se garantiza la factibilidad dentro de las propias funciones.

En el problema de asignación de ambulancias esto dependerá de cómo esté construido el fenotipo y de las restricciones que tenga. Concretamente, para nosotros un individuo puede ser no factible por varios motivos:

- Tiene demasiadas ambulancias de algún tipo
- Tiene muy pocas ambulancias de algún tipo
- Tiene más de las ambulancias permitidas en alguna estación.

Restricciones que además son independientes y pueden ocurrir todas al mismo tiempo. Hay que pensar muy bien como solucionamos el fenotipo para no perjudicar más a un tipo de ambulancias que al otro, principalmente por la última restricción, que será la primera en comprobarse.

El esquema seguido por la función de normalización para solucionar la restricción de las estaciones podemos encontrarlo en la Ilustración 22.

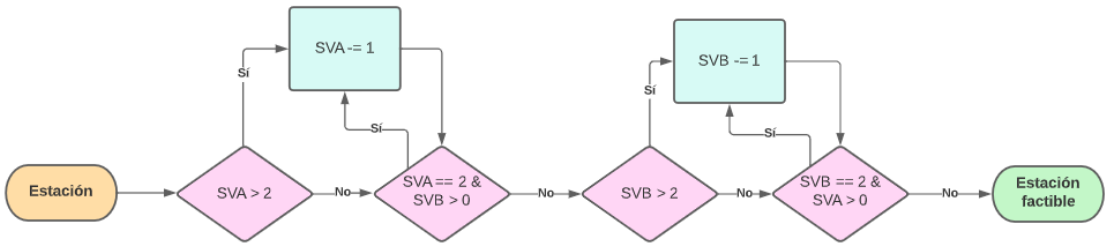

Ilustración 22. Esquema de corrección de las estaciones (3º restricción). Elaboración propia con Lucidchart.

Algunos ejemplos:

- 3 SVA + 2 SVB -> 2 SVA + 2SVB -> 1 SVA + 2SVB -> 1 SVA + 1 SVB -> Estación Factible
- 0 SVA + 3 SVB -> 0 SVA + 2 SVB -> Estación Factible

Es importante tener en cuenta que por cómo son las funciones de cruce y mutación, como máximo podremos tener 3 ambulancias del mismo tipo en la misma estación (si por la mutación se le añade una ambulancia a una estación que ya tenía 2). Por cruce nunca sobrepasarán el valor de de 2 ya que no aumenta ambulancias. El código implementado para solucionar este apartado, por lo tanto, está planteado pensando en las posibles situaciones en las que el fenotipo no será factible. Es decir, que a pesar de poder contener individuos no factibles, siempre podremos controlarlos.

Una vez está solventado el primer problema, es posible que igualmente acabemos con más ambulancias en total, de alguno de los tipos. Para solucionar este segundo problema, eliminaremos vehículos de estaciones aleatorias, de entre las que tienen al menos 1 ambulancia, hasta que el total de ambulancias sea igual a 10 o 19, dependiendo de si es SVA o SVB. Este apartado de la corrección se puede visualizar en la Ilustración 23.

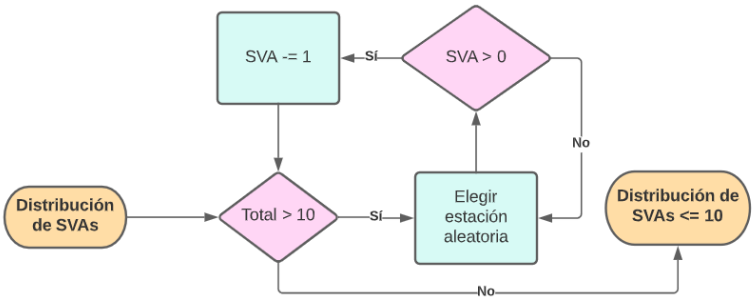

Ilustración 23. 1º parte del proceso de corrección de las restricciones 1 y 2 (para SVAs). Si lo necesitamos eliminamos ambulancias hasta que haya un máximo en total. Elaboración propia con Lucidchart.

Por último, se comprueba si hay menos ambulancias de las que deberían de algún tipo, y aleatoriamente se van insertando. En este paso además hay que comprobar que al insertar no volvamos a infringir la primera restricción, y para ello hay que comprobar si SVA + SVB < 2, en la estación que queramos añadir una más. Este proceso se muestra en el diagrama de la Ilustración 24.

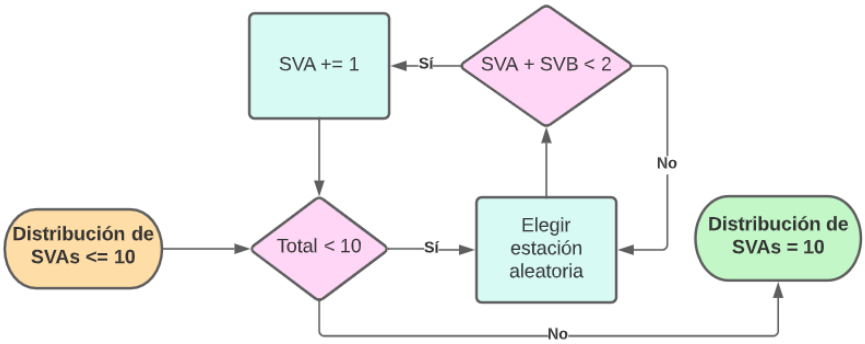

Ilustración 24. 2º parte del proceso de corrección de las restricciones 1 y 2 (para SVAs). Si lo necesitamos añadimos ambulancias hasta que haya exactamente las necesarias. Elaboración propia con Lucidchart.

De esta forma después de las operaciones de cruce y mutación nos aseguramos de que las distribuciones queden con exactamente 10 SVAs y 19 SVBs, y podamos crear las simulaciones.

7.2.4. Resumen del algoritmo

Algoritmo 1. Resumen del algoritmo genético estado del arte (pseudocódigo). Elaboración propia en LaTeX con Overleaf.

Algorithm 1 Algoritmo Genético "Estado del arte"

```
Inicializar población
Evaluar fitness de cada individuo

Mientras tiempo total < 600s
    Guardar mejores 3 individuos
    Seleccionar individuos para cruce y mutación
    Realizar cruce por 1 punto y guardar hijos
    Realizar mutación y guardar individuos
    Evaluar fitness de la población resultante

Evaluar la población sobre todas las llamadas
Devolver mejor individuo como solución
```

El criterio de parada de 600s lo hemos añadido nosotros para que la experimentación sea justa entre los diferentes métodos. Esto se verá mejor en el Capítulo 8.

7.3. Solución Algoritmo Genético propio

A partir del algoritmo genético estado del arte, buscamos mejorarlo aprovechando nuestro conocimiento sobre el problema y que pueda servir de referencia para otras ciudades con una distribución o situación similar de VES y estaciones.

7.3.1. Definición de la población

Uno de los principales problemas a los que se puede enfrentar el algoritmo genético es converger en un mínimo local. Es posible que haya una combinación de ambulancias perfecta pero muy alejada en el espacio de búsqueda de buenas combinaciones de ambulancias, es por ello que nos gustaría implementar la inclusión explícita de diversidad en la población, es decir, en lugar de mantener solamente una población de buenos individuos (por su fitness), los cuales pueden estar muy cerca entre ellos dentro del espacio de búsqueda (por ejemplo, solo alterando 1 o 2 ambulancias de posición), dividimos la población en 2 grupos:

- C1: Individuos con un alto fitness
- C2: Individuos con alta diversidad respecto al grupo C1.

La diversidad la valoramos como disimilitud. Concretamente dados 2 individuos, la disimilitud de ellos es básicamente la cantidad de ambulancias que tienen en lugares diferentes.

$$computeDissimilarity(ind_1, ind_2) = \sum_{j=1}^{nC1} |ind_1[i,j] - ind_2[i,j]|\,, \qquad \forall_i \in \{1,2\}$$

La Ilustración 25 muestra un ejemplo del cálculo de disimilitud entre dos individuos. Dados ind_1= [1 0 0 1 1]; [1 1 1 0 0] e ind_2= [0 0 2 0 1]; [1 1 0 0 1] su disimilitud será (1 + 0 + 2 + 1 + 0) + (0 + 0 + 1 + 0 + 1) = 6.

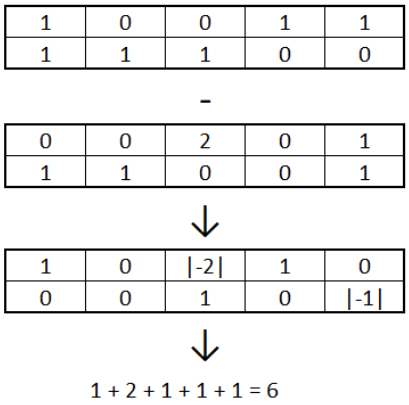

Ilustración 25. Cálculo de la disimilitud entre 2 individuos. Elaboración propia en Excel.

La cantidad de individuos de estos grupos "nC1" y "nC2" serán hiperparámetros a optimizar. Igualmente, se plantea la opción de que nC2 = 0, que querrá decir que esta idea no es beneficiosa para el algoritmo.

Los individuos de C1 y C2 se deciden al crear la población, y luego se actualizan al final de cada iteración del algoritmo. Los nC1 mejores se convierten en el nuevo grupo C1 y sobre el resto se calcula su disimilitud con respecto al grupo C1, y los nC2 con valor más alto se quedan en C2.

$$Dissimilarity(C_2[i]) = \sum_{j=1}^{nC1} computeDissimilarity(C_1[j], C_2[i]), \forall_i \in \{1 \ldots nC1\}$$

Llegado el momento de escoger los padres para los hijos, es posible que coincidan pares del mismo grupo o mixtos entre C1 y C2. Tanto a los padres como a los hijos se les calcula su nuevo fitness en función de un conjunto de llamadas aleatorias, *n_calls*, al igual que en el algoritmo genético original.

7.3.2. Estrategia de selección de padres

En este nuevo enfoque del algoritmo genético, hemos implementado una estrategia de selección por torneo para elegir a los padres que participarán en las operaciones de cruce y mutación. A diferencia del modelo anterior, donde toda la población que sobrevivía actuaba de padre, ahora realizamos una competencia entre varios individuos para determinar quiénes serán los seleccionados como padres. Consideramos necesario una forma de selección de padres que no utilice a toda la población, ya que el tiempo de cómputo es uno de los recursos que más frena a los algoritmos genéticos, y en nuestro caso, necesitando simular y comprobar tantos individuos diferentes, todavía más. La selección por torneo es un método comúnmente utilizado en la literatura cuando tenemos este tipo de problemas.

Este mecanismo de selección simula una competencia entre individuos de la población. Se elige un número fijo de participantes al azar, es decir el tamaño del torneo (por ejemplo, *k*

individuos) y se compara su valor de fitness, el mejor individuo tiene una cierta probabilidad, *p*, de ser seleccionado, y este proceso se lleva a cabo para tantos padres, *L*, como queramos considerar. Estas 3 variables serán hiperparámetros del algoritmo genético, aunque guiaremos su valor indicando límites superiores e inferiores a no traspasar.

Para el tamaño del torneo, *"tournamentPart"* se considerarán valores entre 2 individuos (torneo binario) hasta 10. Un valor mayor de "*k"* promoverá una mayor diversidad genética y una exploración más amplia del espacio de soluciones, pero también puede aumentar la presión selectiva y disminuir la exploración local.

La probabilidad de selección del mejor individuo, *"bestProb"*, se dejará entre 0.7 y 1.0. Generalmente serán los mejores individuos los que actúen de padres, pero dejando este valor por debajo de 1 damos más posibilidades a que individuos del grupo C2 tengan hijos, que es una de las innovaciones principales.

La cantidad de padres a seleccionar se va a plantear como un porcentaje de la población, *"tournamentSel",* ya que el tamaño de la propia población (nC1 + nC2) depende de otros hiperparámetros, por lo que en cada modelo cambiará. Para el porcentaje se probarán valores entre 10% a 100%, aunque se redondeará hacia arriba la cantidad para que siempre haya un número par de padres. La Ilustración 26 muestra un ejemplo del proceso de selección por torneo, que podría darse con nuestros datos.

Chromosome	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Fitness	3.3	3.6	3.8	3.02	4.2	4.5	3.42	3.82	4.01	3.23

Tournament size = 3

Chromosome	C2	C6	C9
Fitness	3.6	4.5	4.01
↙		↘	
bestProb		1-bestProb	
C2		C6	C9
100%		50%	50%

Ilustración 26. Ejemplo de selección por torneo con 10 individuos. Elaboración propia en Excel.

7.3.3. Función de cruce

Una de las principales mejoras que se nos ocurrieron fue la de aprovechar la información de la ubicación de las estaciones para mejorar la función de cruce. Debido a la gran cantidad de estaciones que tenemos y la proximidad unas con otras, en vez de intercambiar una ambulancia de una estación con otra, intercambiar vecindarios de estaciones. Podemos calcular cuánto tiempo hay de una estación a otra y construir una matriz de distancias gracias a Open Route Service, la misma API que utilizamos para construir la red de carreteras.

Para esto solamente necesitamos la posición exacta de las estaciones, que es una variable disponible en forma de latitud/longitud en el fichero con la información de las estaciones.

Hay que tener en cuenta que la distancia que hay de una estación la otra no es la misma que al revés, ya que depende de cómo sean las carreteras. Por otra parte, ORS tiene un límite de llamadas a su API al minuto (al menos en la versión gratuita) por lo que cada 40 llamadas necesitaremos dejar dormido el bucle para que no nos corte la conexión.

Esta matriz de distancias la podemos convertir en una vector de listas que nos diga para cada estación las que están a una cierta distancia de ella. Después de varios experimentos con los tutores, decimos considerar que una estación es cercana a otra si se puede llegar en menos de 300 segundos, de esta forma dada una estación cualquiera, su vecindario son aquellas estaciones que estén a 5 minutos de ella.

Con esta nueva información proponemos una nueva función de cruce, donde se elige una estación aleatoria y todas las estaciones vecinas se intercambian entre los padres. Un ejemplo de este cruce puede visualizarse en la Ilustración 27. Si contamos con unos padres p1 = [1 0 1 1 0]; [1 0 1 0 1] y p2 = [1 2 0 0 0]; [0 0 0 2 1] y aleatoriamente se escoge la estación 2, que tiene de vecindario [2,4,5], el primer hijo será c1 = [1 2 1 0 0]; [1 0 1 2 1] y el segundo hijo será c2 = [1 0 0 1 0]; [0 0 0 0 1].

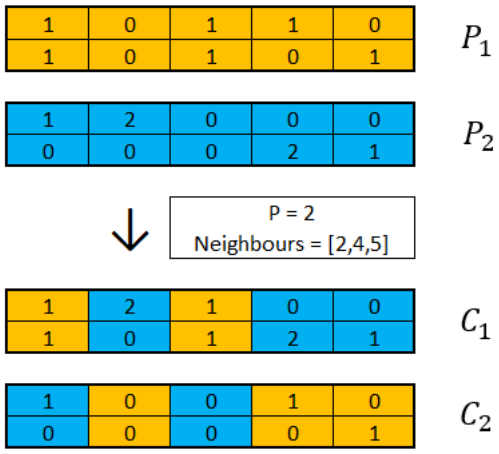

Ilustración 27. Ejemplo de cruce por vecinos. Elaboración propia en Excel.

Además, en lugar de limitarnos a seleccionar un único grupo de vecinos para el intercambio, y dado el gran número de estaciones involucradas en nuestro problema, hemos incluido el número de grupos de vecinos, *"n_groups"*, como una variable ajustable en el algoritmo genético. Esto quiere decir que se seleccionarán varios puntos aleatorios y se obtendrán sus estaciones vecinas, y todas las estaciones resultantes serán las que se intercambiarán para crear los hijos, lo que nos permite intercambiar múltiples conjuntos de vecindarios y explorar una mayor diversidad de soluciones potenciales.

Como se puede ver en el ejemplo, después de la operación de cruce, es posible que los hijos generados no sean factibles en términos del problema, por lo que será necesario aplicar nuestra función de corrección (fixing) para garantizar la viabilidad de los hijos.

Asimismo, hemos introducido otro hiperparámetro llamado *"n_children"*, que determina cuántas parejas de hijos generaremos a partir de cada par de padres. Esta decisión se basa en la premisa de que generar un mayor número de hijos en cada iteración puede acelerar la

convergencia del algoritmo, y dado el alto grado de aleatoriedad introducido por el cruce, cada hijo resultante será prácticamente único y enriquecerá la exploración del espacio de soluciones.

7.3.4. Función de mutación

La función de mutación se va a mantener igual que la del anterior algoritmo genético, pero en este caso serán los hijos fruto del cruce los que reciban la mutación, con una cierta probabilidad (*p_mut*) que será otro hiperparámetro a optimizar. En el anterior caso había un grupo de la población que recibía cruce y el otro, mutación.

Para visualizar entonces cómo queda la población en comparación con el anterior modelo, suponiendo una población de 20 individuos ordenado de mejor a peor fitness, podemos visualizar la Ilustración 28. En el primer modelo el color naranja indica individuos que directamente pasan a la siguiente población, en color azul individuos que reciben cruce y en color verde individuos que reciben mutación. En el segundo modelo los colores significan lo mismo, excepto para el verde claro, que serán los individuos que reciben cruce y luego sus hijos mutación, en este caso suponiendo una selección por torneo con *tournamentSel* = 50%, *bestProb* = 80% (aproximadamente) y *p_mut* = 20%.

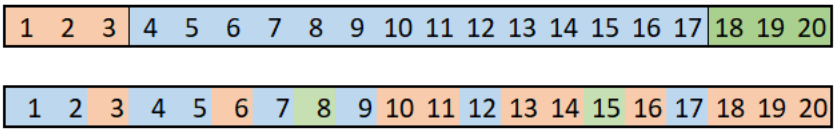

Ilustración 28. Comparación de una población de 20 individuos en el algoritmo genético estado del arte (arriba) y el nuestro (abajo). Elaboración propia con Excel.

Dependiendo de los hiperparámetros, un gran porcentaje de la población se puede quedar sin hacer nada durante este proceso, pero esto no es un problema ya que igualmente tanto los hijos como los padres pasarán a evaluar de nuevo su fitness, y si es necesario su disimilitud, para ver si sobreviven o no. Si el individuo es bueno para la población debería sobrevivir de nuevo, ya que si no puede ser porque tuviera suerte con la selección de las llamadas de la anterior iteración. En cada iteración estamos utilizando unos pocos días de llamadas cada vez, para acelerar el algoritmo.

7.3.5. Resumen del algoritmo

Algoritmo 2. Resumen de nuestra propuesta de algoritmo genético (pseudocódigo). Elaboración propia en LaTeX con Overleaf.

Algorithm 2 Algoritmo Genético mejorado

```
Inicializar población y evaluar fitness de cada individuo
    Crear grupo C1 de alto fitness
    Crear grupo C2 de alta disimilitud respecto a C1

Mientras tiempo total < 600s
    Seleccionar individuos para cruce por torneo
    Realizar cruce por vecindarios y guardar hijos
    Realizar mutación sobre hijos con cierta probabilidad
    Evaluar fitness de toda la población resultante y la anterior
        Actualizar grupo C1
        Actualizar grupo C2

Evaluar C1 sobre todas las llamadas
Devolver mejor individuo como solución
```

En conclusión, con estos cambios queremos incrementar la velocidad de convergencia del algoritmo, al mismo tiempo que mantenemos diversidad en la población para no caer en un mínimo local. También hemos querido dirigir un poco la búsqueda, asumiendo que estaciones cercanas forman un núcleo y responderán a emergencias similares.

7.4. Solución Búsqueda Tabú

La búsqueda tabú también es una de las técnicas que pueden resultar efectivas en nuestro problema si sabemos cómo aprovecharla. Se podría definir la búsqueda tabú como una búsqueda local dirigida, donde a partir de un individuo inicial se busca en su vecindario uno que sea mejor que él, sin embargo, si ese movimiento implica crear un bucle en la búsqueda, se cancela, concretamente nos vamos guardando una lista (tabú) de movimientos realizados, de forma que no puedan ser repetidos.

La lista tabú es precisamente una colección de los movimientos que va realizando el algoritmo, y que no podrá repetirse mientras ese movimiento está en la lista. Esta lista tendrá un tamaño máximo (N), de forma que eventualmente un movimiento si es necesario se puede repetir, pero no antes de haber realizado N en otra dirección.

7.4.1. El vecindario

La definición de lo que se considera un vecino para un individuo la definimos nosotros, por ejemplo, si una distribución de ambulancias es [1 0 2 0 0]; [1 1 0 0 1], los vecinos podrían ser aquellas distribuciones que tengan un valor del concepto de disimilitud que hemos definido antes = 2, es decir en las que sólo varíe una ambulancia, por ejemplo [1 0 1 1 0]; [1 1 0 0 1], moviendo una ambulancia de lugar. Este concepto de vecino se ha utilizado en el estado del arte y vamos a mantenerlo en nuestro modelo.

Para generar un vecindario entonces, debemos iterar sobre todas las estaciones y si hay al menos una ambulancia de un tipo, eliminarla, y probar a añadirla a las demás estaciones una por una si es factible. Podemos ver un ejemplo de cómo funciona en pseudocódigo en el Algoritmo 3.

Algoritmo 3.Cómo calcular vecinos de un individuo (pseudocódigo). Elaboración propia en LaTeX con Overleaf.

Algorithm 3 Vecindario de un individuo

```
1: Inicializar vecindario vacío
2:
3: Para cada tipo de ambulancia (SVA/SVB)
4:   Para cada estación (1...277)
5:     Si ambulancias > 0                          ▷ Cambiarla de lugar
6:       Eliminar 1 ambulancia de la estación
7:       Para cada estación (1...277)
8:         Añadir ambulancia a la nueva estación
9:         Añadir al vecindario si la distribución es factible
10:
11: Devolver vecindario
```

Para implementar la búsqueda tabú, dados nuestros datos tenemos varios problemas:

1. ¿Cuántos vecinos generamos por individuo?, es decir, ¿cómo de grande debe ser el vecindario?
2. ¿Cómo decidimos qué vecino es el mejor?

En realidad, las dos cuestiones van de la mano, y ahora lo vamos a ver con algunos datos:

Dado 1 individuo con 277 posibles estaciones, 10 ambulancias de tipo 1, 19 ambulancias de tipo 2 y con posibilidad de 2 ambulancias por estación, su vecindario a distancia 1 (mover una ambulancia de estación) tendrá un total de 7500 a 8000 vecinos, dependiendo de la distribución ya que se eliminan los no factibles.

Evaluar 1 individuo con todas las llamadas de un año (365 días) tarda 2.6s, como hemos visto durante el trabajo. Si evaluamos todos los individuos con todas las llamadas para saber qué vecino es el mejor, tardaríamos entre 5h 24m ~ 5h 46m. ¡Y esto es solo para mover 1 ambulancia!

En la Tabla 2 podemos encontrar algunos ejemplos de combinaciones de cantidad de vecinos y llamadas y el tiempo que tardaría en calcularse el fitness de todos los individuos. El tiempo es para realizar solo 1 iteración.

Tabla 2. Tiempo de cómputo del fitness para diferentes cantidades de llamadas y vecinos. Elaboración propia.

% Vecinos	Nº de días de llamadas	Tiempo*
100	365	5h 30m
50	100	45m 12s
10	50	4m 30s
1	365	3m 18s
1	50	27s
10	10	54s

Deberíamos encontrar un punto intermedio entre una cantidad de vecinos que sea suficiente para encontrar un individuo mejor que el actual, y una cantidad de días de llamadas que sea representativa de todo el año. La cantidad de llamadas es un parámetro bastante clave, ya que, si utilizamos pocas llamadas, el algoritmo irá mucho más rápido y hará más iteraciones, pero podemos desplazarnos hacia vecinos que realmente son peores. Si utilizamos muchas llamadas, ocurre justo lo contrario.

En realidad, no es necesario computar todos los vecinos, especialmente al principio de la búsqueda tabú, dado un individuo aleatorio, es muy posible que algún vecino sea mejor que él, para hacernos una idea, hemos hecho pruebas calculando el vecindario completo de un individuo con todas las llamadas. En 20 repeticiones hubo un promedio de 7806.65 vecinos factibles por individuo, de los cuales 2695.05 de ellos fueron mejores que el individuo original. Este valor teóricamente irá disminuyendo a menudo que el algoritmo haga más iteraciones.

$$P(X = k) = \frac{C(K,k) * C(N-K, n-k)}{C(N,n)}$$

Ecuación 4. Distribución hipergeométrica.

Podemos utilizar la fórmula de la distribución hipergeométrica (Ecuación 44) para averiguar la probabilidad de escoger al menos uno (k >= 1) de esos 2965 individuos (K = 2965) de un grupo de 7806 (N = 7806), en un cierto número de intentos. Por ejemplo, si queremos comprobar la probabilidad de que en una iteración evaluando el 1% del vecindario encontremos una solución mejor que la nuestra, n = N * (1/100):

$$P(X = 0) = \frac{(1 * 1.26 \times 10^{119})}{(2.44 * 10^{124})} = 5.16 \times 10^{-6}$$

Ecuación 5. Probabilidad de no escoger ningún individuo mejor en 78 intentos, habiendo 2965 vecinos que sí lo son y 7806 en total.

La probabilidad de escoger 0 individuos mejores es de $5.16x10^{-6}$, (Ecuación 5) por lo tanto la probabilidad de elegir al menos 1 es 1-P(X=0), luego es prácticamente seguro (99.9994%) que con un 1% del vecindario encontremos un individuo mejor en la primera iteración. Esto no quita que sea posible que conforme avance el algoritmo, con un vecindario muy pequeño no encuentre solución mejor, es por ello que entre los autores de este TFG hemos decidido establecer los valores a comprobar en 1%/5% y 10% de vecindario, y en los resultados de la experimentación veremos qué cantidad ha sido mejor. De la misma forma, el número de días de llamadas que se probará será de 5/10/15/20 con propósitos experimentales. La longitud de la lista tabú se considerará otro hiperparámetro a optimizar.

7.4.2. Resumen del algoritmo

Podemos encontrar un esquema del proceso que sigue la búsqueda tabú aplicada en nuestro problema en el Algoritmo 4.

Algoritmo 4. Resumen de la búsqueda tabú (pseudocódigo). Elaboración propia en LaTeX con Overleaf.

Algorithm 4 Búsqueda Tabú

```
Inicializar población y elegir mejor individuo
Inicializar lista tabú

Mientras tiempo total < 600s
    Obtener vecindario del individuo actual
    Evaluar un % del vecindario
    Si fitness vecino > fitness actual y el movimiento ∉ lista tabú
        Actualizar individuo actual con nuevo
        Añadir a la lista tabú el movimiento realizado

Evaluar individuo actual sobre todas las llamadas
Devolver individuo como solución
```

7.5. Solución Búsqueda Aleatoria

La búsqueda aleatoria, también conocida como *random search*, es una técnica ampliamente utilizada en la optimización que se suele emplear como modelo base para comparar los resultados obtenidos con otros algoritmos avanzados como los algoritmos genéticos. En la búsqueda aleatoria es van generando soluciones de manera aleatoria y se evalúan según el criterio de optimización establecido. Aunque el enfoque es simple y puede presentar limitaciones, su utilización como punto de referencia permite evaluar la eficacia de otras técnicas más complejas.

En el contexto de nuestro estudio, utilizaremos la búsqueda aleatoria como modelo base para comparar los resultados con la búsqueda tabú y los genéticos. Aunque nuestro objetivo es superar el estado del arte, ya sea con modelo que logre mejores valores promedios o una convergencia más rápida, no está de más evaluar la diferencia que haya con una búsqueda de soluciones trivial, que debería ser relativamente peor. Esta técnica solo requiere un hiperparámetro, que es el tiempo máximo de ejecución, el cual determina la cantidad de individuos que podemos evaluar.

En este caso en particular, los individuos serán evaluados utilizando todas las llamadas disponibles durante el año para conocer su tiempo de respuesta final. La estrategia que hemos llevado a cabo de utilizar menos llamadas para calcular la bondad intermedia durante las ejecuciones de las metaheurísticas forma parte de su propia estrategia.

Para aprovechar la capacidad de multithreading de Julia en la búsqueda aleatoria, generaremos distribuciones de 16 individuos en cada iteración, ya que contamos con 16 CPUs en la máquina donde se llevarán a cabo los experimentos. Esta técnica para acelerar la ejecución de los algoritmos también se ha utilizado en los genéticos y en la búsqueda tabú, pero en la

búsqueda aleatoria su aplicación es más clara y directa. Podemos encontrar un resumen en el Algoritmo 5

Algoritmo 5. Esquema de la búsqueda tabú (pseudocódigo). Elaboración propia en LaTeX con Overleaf.

Algorithm 5 Búsqueda Aleatoria

```
1: Inicializar un individuo vacío con fitness = +∞
2:
3: Mientras tiempo total < 600s
4:     Generar 16 individuos aleatorios          ▷ Aprovechando las CPUs
5:     Si fitness mejor individuo < fitness actual
6:         Actualizar individuo actual
7:
8: Devolver individuo actual como solución
```

Capítulo 8. Experimentación

Como vimos en el apartado 4.2. Hardware, la experimentación se lleva a cabo dentro de una máquina virtual preparada para soportar la cantidad de recursos que necesita JEMSS y las operaciones de los algoritmos que vamos a probar. También es capaz de mantenerse en ejecución sin parar durante el tiempo que haga falta, algo que necesitaremos si queremos lanzar nuestros experimentos.

Para que la comparación entre las técnicas sea justa, se va a utilizar un límite de tiempo de 600 segundos en las ejecuciones de los algoritmos, es un tiempo que puede ser relativamente alto, pero de esta forma también podemos ver qué modelo converge antes.

Además del valor del tiempo medio de respuesta, de cada ejecución también nos guardaremos el instante de tiempo y la iteración en el que fue encontrada la mejor solución, así como el total de iteraciones y una lista con las mejores soluciones que va encontrando y en qué momento, para poder hacer gráficos y ver qué métodos convergen antes. Al computar un tiempo de respuesta "no definitivo" mediante el uso de un subconjunto de los días, en los algoritmos genéticos esta evolución será muy ruidosa, pero veremos como a lo largo del tiempo el valor va disminuyendo.

8.1. Resultados de la búsqueda tabú

La Tabla 3 detalla todos parámetros que se han utilizado en la búsqueda tabú. Cada combinación de hiperparámetros se probó, al igual que en el primer modelo, por 30 repeticiones para que los resultados fueran robustos, en total se hicieron 720 ejecuciones y el tiempo total de cómputo de los experimentos fue de 1 semana. Por otra parte, los resultados promedio de cada combinación de parámetros se pueden visualizar en la Tabla 4.

Tabla 3. Parámetros de la búsqueda tabú. Elaboración propia.

Parámetros	Significado	Valores
neighb_p	% de vecinos a crear	1%/5%/10%
ncalls	Nº de días de llamadas	5/10/15/20
nTabuL	Longitud de la lista tabú	5/10

Tabla 4. Resultados de la experimentación de la búsqueda tabú. Elaboración propia en Python (pandas) mediante Visual Studio Code.

		fitness		best_time		best_iter		total_iters	
	nTabuL	5	10	5	10	5	10	5	10
neighb_p	ncalls								
0.01	5	3.087549	3.077683	398.330212	456.555061	71.933333	43.800000	109.733333	58.000000
	10	3.086961	3.085900	418.849136	448.158891	73.533333	39.466667	106.333333	53.366667
	15	3.080661	3.084631	406.758045	420.653736	71.033333	40.166667	106.266667	58.066667
	20	3.091896	3.083450	387.797615	435.852922	70.366667	41.500000	110.533333	58.333333
0.05	5	3.104386	3.128533	472.028515	506.838711	18.866667	10.733333	24.466667	12.033333
	10	3.111475	3.137164	471.035475	527.302918	18.766667	10.600000	24.300000	12.033333
	15	3.103618	3.137328	471.654279	483.736190	18.766667	9.733333	24.366667	12.000000
	20	3.099439	3.128301	470.500783	512.573554	18.800000	10.300000	24.200000	12.033333
0.10	5	3.152208	3.201205	457.038756	527.712870	9.466667	5.300000	12.000000	5.933333
	10	3.137393	3.206789	505.239452	526.565403	10.133333	5.200000	12.033333	5.866667
	15	3.135000	3.218263	482.397849	531.169332	10.100000	5.033333	12.033333	5.933333
	20	3.148807	3.222034	485.699032	537.976382	10.166667	5.433333	11.966667	6.000000

Inmediatamente si nos fijamos en las columnas de “fitness”, los mejores resultados promedios son en los que el número de días de llamadas = 5; y el % de vecindario a evaluar fue 1%, sugiriendo que es mejor utilizar una estrategia que se mueva rápidamente por las soluciones. Lo podemos ver también si nos fijamos en la columna de las iteraciones, ya que utilizar menos vecindario ha agilizado mucho la ejecución, permitiendo más iteraciones del algoritmo. Esta combinación también encuentra la mejor solución más rápido, ya que el tiempo promedio ha sido alrededor de 400s, al menos cuando la longitud de la lista tabú es 5.

Sorprende ver cómo actúa el parámetro de la lista tabú en la experimentación. Primero en cuanto al tiempo medio de respuesta, a menudo que incrementamos el % de vecindario a explorar, una longitud de lista tabú = 10 resulta en peores individuos. Esto lo podemos ver especialmente cuando % vecindario = 10. Por otra parte, en cuanto al número de iteraciones que realiza la búsqueda tabú, se puede ver como de forma consistente, la longitud de la lista es inversamente proporcional tanto al número total como al número para encontrar la mejor solución.

Podemos utilizar un test estadístico para asegurarnos de que la selección de los parámetros es realmente significativa a la hora de obtener resultados. Para ello podemos utilizar la prueba T-Student (Student, 1908), por el cual comprobamos si la media de 2 poblaciones diferentes es igual. Esta prueba asume que la distribución de las poblaciones es normal, y la varianza también debería de serlo, aunque no es necesario si el tamaño de la muestra es igual (Carol A. Markowski, 1990) . Para comprobar que la población es normal podemos utilizar otro test estadístico, por ejemplo, la prueba de Shapiro-Wilk (S.S. Shapiro, 1965), una de las más antiguas y utilizadas para el contraste de normalidad.

Cada prueba estadística que realicemos tendrá asociado un p-valor, que nos indicará si es recomendable aceptar la hipótesis que plantea el test, concretamente si el p-valor es < 0.05. Exceptuando la propia prueba de shapiro-wilk, donde las distribuciones serán normales si el p-valor > 0.05. Todos los tests los vamos a realizar utilizando la librería scipy.stats de Python.

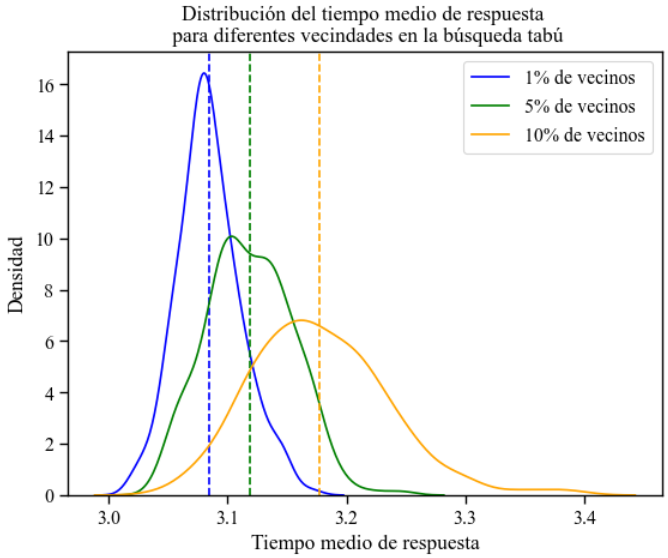

Ilustración 29. Distribución del tiempo medio de respuesta cuando el porcentaje de vecindario evaluado es 1%, 5% o 10% en la búsqueda tabú. Las líneas verticales indican la media de cada distribución. Elaboración propia en Python (seaborn).

Una distribución normal debería tener forma de campana de gauss, la podemos visualizar para nuestros datos en el tiempo medio de respuesta, por ejemplo, para diferentes valores de vecindarios, en la Ilustración 29.

En este caso, tanto el grupo de 1% de vecinos, con un p-valor de 0.06; como el grupo de 5% de vecinos, con un p-valor de 0.069, siguen una distribución normal. El último grupo no sigue una campana de gauss, por lo que para comparar las medias entre ellos deberíamos utilizar otras técnicas. Sin embargo vamos a comparar los primeros 2 grupos mediante el estadístico T, para ver si hay diferencias significativas. El test T-Student da como resultado 11.79, con un p-valor de 10^-28, sugiriendo claramente que las medias son diferentes, más específicamente que la media del segundo grupo es superior, lo cual confirma lo que vemos en el gráfico, ya que la media del primer grupo es 3.084 y la del segundo, 3.118.

Viendo que lo mejor es utilizar un 1% para el vecindario, podemos averiguar ahora si es significativo realmente el número de llamadas (Ilustración 30). Realizando el test Shapiro-Wilk, podemos averiguar que las poblaciones con vecindario = 1% y cantidad de días de llamadas = 5/10/15 siguen una distribución normal, pero no cuando los días de llamadas son = 20 (p-valor = 0.01). Esta información queda resumida en la Tabla 5.

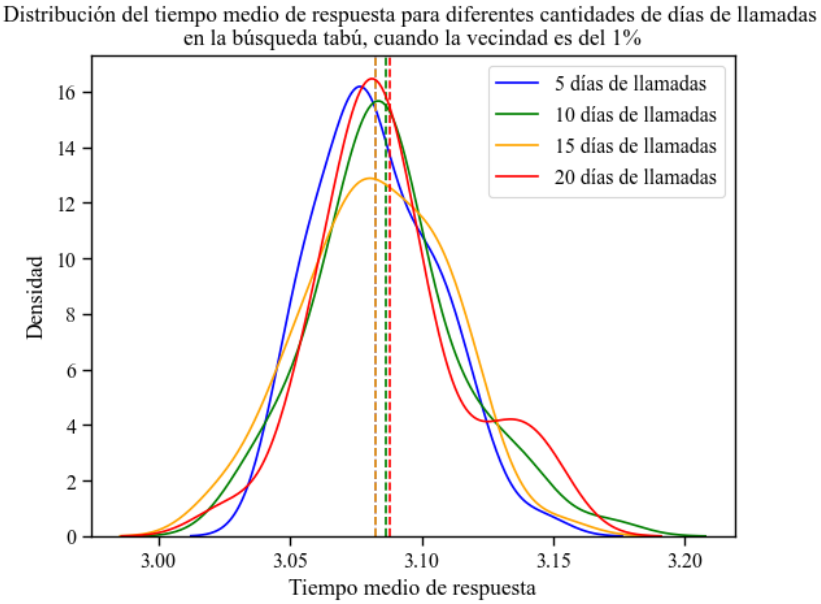

Ilustración 30. Distribuciones del tiempo medio de respuesta para diferentes cantidades de llamadas cuando mantenemos vecindad de 1% en la búsqueda tabú. Las medias son prácticamente idénticas, lo que sugiere que no hay diferencias entre variables. Elaboración propia con Python (seaborn).

Tabla 5. Resultado de la prueba shapiro-wilk para diferentes cantidades de días de llamadas en la búsqueda tabú. Elaboración propia.

Población	Shapiro-Wilk p-valor
1. neighb_p = 1%, ncalls = 5	0.289
2. neighb_p = 1%, ncalls = 10	0.183
3. neighb_p = 1%, ncalls = 15	0.959
4. neighb_p = 1%, ncalls = 20	0.0182

Como todas las medias no siguen una distribución normal, necesitamos utilizar otro test estadístico para comparar las medias que no necesite como premisa que se cumpla esa

condición. Concretamente podemos utilizar el test de Wilcoxon (Wilcoxon, 1945) de comparación de medias. Estas técnicas dentro de los test estadísticos se denominan no-paramétricas y son otro recurso igual de válido que podemos utilizar.

La prueba de wilcoxon nos indica de todas formas que las medias no son significativamente diferentes (Tabla 6), por lo que podemos concluir que el parámetro más determinante es la cantidad de vecindario explorado, y concretamente lo mejor es utilizar un 1%. Quizá con menos también hubiera ido mejor, dados los resultados y los tests que hemos hecho. Además si queremos que la convergencia a la solución sea lo más rápida posible deberíamos utilizar una lista tabú más pequeña, que encuentra la mejor solución más pronto.

Tabla 6. Resultados de la prueba de Wilcoxon entre las poblaciones de la Tabla 5. Ninguna es significativamente diferente a otra, lo que sugiere que las medias son iguales. Elaboración propia.

Población	Wilcoxon p-valor
(1,2)	0.253
(2,3)	0.658
(3,4)	0.746
(1,3)	0.912
(1,4)	0.361
(2,4)	0.734

Dentro de los propios resultados de la búsqueda tabú, el mejor tiempo promedio de respuesta a las emergencias que ha sido capaz de encontrar ha sido de 3.021 minutos, en una de las ejecuciones en los que los parámetros fueron neighb_p = 1%, ncalls = 10 y nTabuL = 5. Encontró la solución en 362s, unos 6 minutos y no mejoró más a partir de allí. El tiempo de respuesta "parcial" que encontró llegó a ser de un poco menos de 2.9 minutos, mientras que el valor final del tiempo de respuesta promedio fue de 3.021 minutos. Podemos ver la evolución en la Ilustración 31.

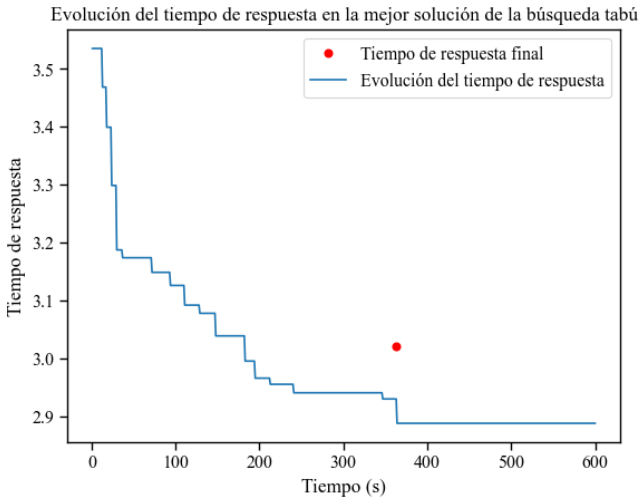

Ilustración 31. Evolución del tiempo de respuesta en la mejor ejecución de la búsqueda tabú. Elaboración propia con Python (seaborn).

La distribución de ambulancias, que propuso esta ejecución, se puede observar en el mapa de la Ilustración 32. Con solo 2 estaciones con 2 ambulancias, que además fueron de diferente

tipo, en Torrent y Alfafar, plantea una asignación muy diversa por la provincia, pero dejando claro que hay que poner la mayoría por la zona de Valencia ciudad. El sur del área metropolitana, la zona de la Albufera, Silla y Picassent se quedan sin ambulancias, que sería uno de los únicos puntos negativos.

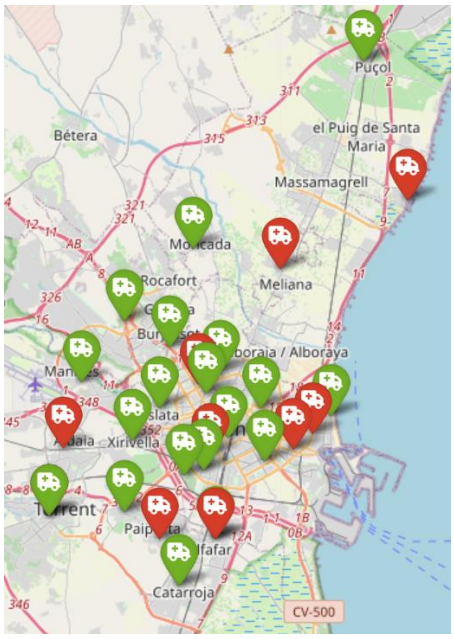

Ilustración 32. Distribución de ambulancias propuesta por la búsqueda tabú. Elaboración propia con Python (Folium).

8.2. Resultados del algoritmo genético estado del arte

Para la experimentación de este algoritmo evolutivo se van a dejar los valores de los parámetros tal y como estaban en el paper original, solamente vamos a introducir un par de cambios. Debido a nuestros datos y como hemos visto a lo largo del TFG, es inviable utilizar todas las llamadas cada vez que queremos saber cómo de buena es una distribución, es por eso que, aunque los autores que diseñaron este algoritmo para el problema de asignación de ambulancias sí las usaron, nosotros solo utilizaremos un conjunto de días para la evaluación intermedia (durante las iteraciones). Los individuos del grupo final que sobrevivan serán evaluados con todas las llamadas y se elegirá al mejor.

En el algoritmo original el proceso se corta cuando se llega a un número máximo de iteraciones, o el mejor valor no mejora en una cierta cantidad de iteraciones. En este caso y para que las comparaciones entre modelos sean iguales sólo se utilizará el criterio de parada por límite de tiempo que hemos establecido al principio (600s). Los parámetros de este modelo pueden visualizarse en la Tabla 7.

Tabla 7. Parámetros de la experimentación para el algoritmo genético "estado del arte". Elaboración propia.

Parámetros	Significado	Valores
N	Tamaño de la población	20
ncalls	Nº de días de llamadas	5/10/15
P_nextpop	% de individuos que pasan a la siguiente iteración	$^{3}/_{20}$ (los 3 mejores)
P_crossover	% de individuos que reciben cruce	$^{14}/_{20}$ (los siguientes 14 mejores)

De esta forma, el único hiperparámetro del modelo será el número de días de llamadas. Cada combinación se realizó 30 veces para que los resultados fueran robustos, para un total de 90 ejecuciones. Los resultados promedio de cada cantidad de días de llamadas pueden observarse en la Tabla 8.

Tabla 8. Resultados del primer algoritmo genético. Cada número de llamadas fue promediada 30 veces. Elaboración propia.

Ncalls	5	10	15
Fitness	3.465	3.478	3.53
Best_time	387.52	351.83	376.58
Total_iters	220.26	119.66	50.56

En cuanto al fitness, parece que no hay gran diferencia entre la cantidad de días de llamadas que utilizamos, aunque sí que es cierto que a menor número de llamadas, el tiempo de respuesta promedio es menor. Las distribuciones de cada grupo las podemos visualizar en la Ilustración 33, y para comprobar si hubiera diferencia significativa entre ellos se puede utilizar de nuevo un test estadístico.

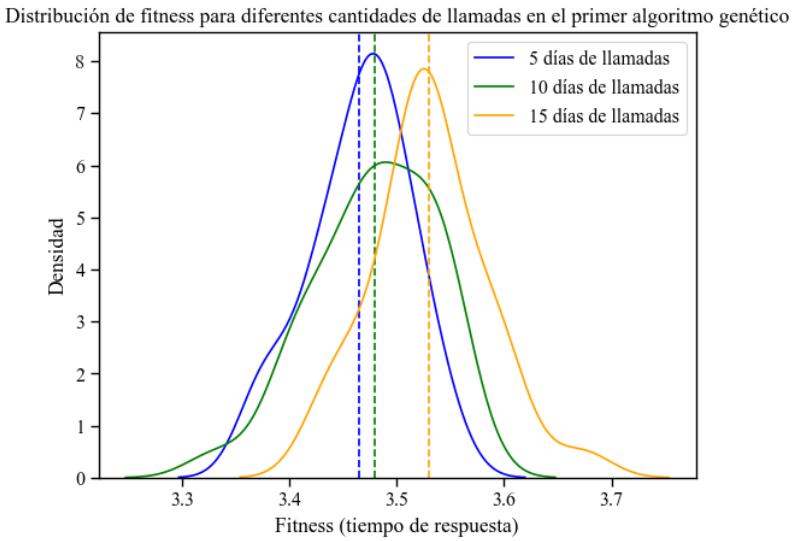

Ilustración 33. Distribución de fitness en las diferentes cantidades de llamadas del primer algoritmo genético. Todas siguen una campana de gauss. Elaboración propia con Python (seaborn).

Las distribuciones parecen seguir una campana de gauss, y empleando el test de shapiro-wilk, se confirma. Si utilizamos ahora el estadístico t-student para averiguar si hay diferencias

entre los conjuntos de datos, descubrimos que entre el primer y segundo grupo (5 y 10 días de llamadas) no las hay, pero ambos son significativamente diferentes al grupo de 15 días de llamadas. Los resultados se pueden visualizar en la Tabla 9.

Tabla 9. Resultados de la prueba T-student (p-valor) para diferentes cantidades de llamadas, en el algoritmo genético estado del arte. El p-valor = 0.30 de la primera comparación indicia que no es significativamente diferente utilizar una cantidad u otra, en cuanto al tiempo de respuesta promedio resultante. Elaboración propia.

Población	**T-student p-valor**
(5 días de llamadas, 10 días de llamadas)	0.3024
(5 días de llamadas, 15 días de llamadas)	0.0000046
(10 días de llamadas, 15 días de llamadas)	0.0005

De esta forma podemos confirmar que no hay diferencia entre utilizar 5 o 10 días de llamadas en cuanto al mejor resultado que vayamos a conseguir. En este caso podríamos decidir qué valor es mejor, mediante el tiempo en el que se encuentra la mejor solución o "*best_time*". Sin embargo, de la misma forma que con el fitness, no encontramos diferencias significativas entre este tiempo para ambas distribuciones.

Por otra parte, podemos visualizar cómo evoluciona el fitness a lo largo de una ejecución del algoritmo genético, en la Ilustración 34. En este caso la mejor solución utiliza 10 días de llamadas, y se encuentra al llegar a los 280 segundos en el algoritmo. También da la sensación de que empeora y mejora un poco aleatoriamente, pero esto es un efecto de utilizar un fitness parcial con las distribuciones. El punto rojo exactamente indica el mejor valor de fitness encontrado, utilizando todas las llamadas. Este fue obtenido por una distribución que se encontró al llegar a los 280 segundos y sobrevivió el resto del tiempo, hasta ser evaluada con todas las llamadas.

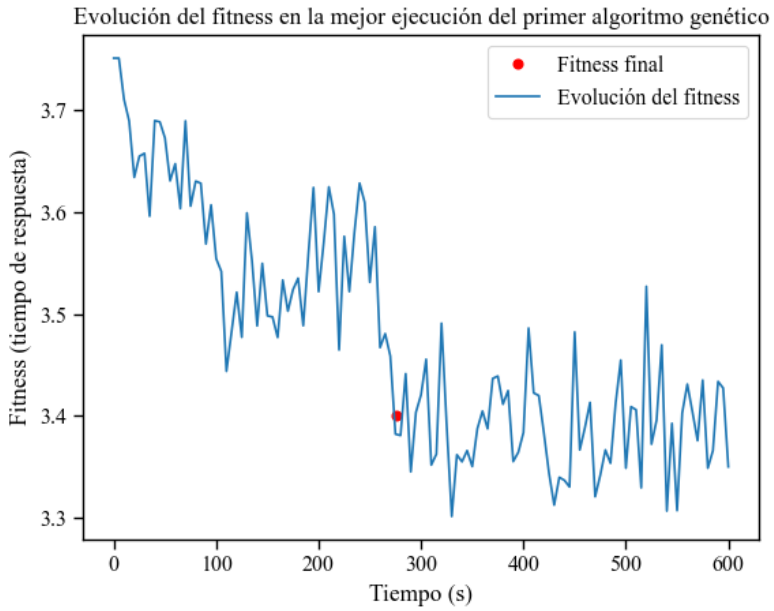

Ilustración 34. Evolución del fitness en la mejor ejecución del primer algoritmo genético. La mejor solución se encuentra poco antes de llegar a los 300 segundos de ejecución. Elaboración propia con Python (seaborn).

Esta ejecución del algoritmo encontró una solución donde el tiempo medio de respuesta era de 3.4 minutos, podemos visualizar la distribución propuesta en un mapa, al igual que con la búsqueda tabú, en la Ilustración 35.

Ilustración 35. Distribución de ambulancias propuesta por el primer algoritmo genético. Elaboración propia con Python (folium).

Los resultados muestran que se nota una falta de criterio al colocar algunas ambulancias, por ejemplo, la mayoría de SVB están en el centro de Valencia, mientras que municipios alrededores como Aldaia y Torrent solo cuentan con SVAs. Por otra parte, hay una falta de ambulancias entre Puçol y Valencia, que deja una zona del área metropolitana sin cubrir.

8.2. Resultados de nuestro algoritmo genético

Debido a la gran cantidad de combinaciones de parámetros que involucra este algoritmo genético, hemos decidido utilizar la búsqueda bayesiana para determinar los mejores. La búsqueda bayesiana es una técnica de optimización que permite encontrar la configuración óptima de los hiperparámetros a través de un proceso iterativo de evaluación y ajuste basado en principios estadísticos y probabilísticos.

En la búsqueda bayesiana, se construye un modelo probabilístico denominado modelo de respuesta, que relaciona los valores de los hiperparámetros con el rendimiento del algoritmo genético. Este modelo se actualiza de forma iterativa a medida que se exploran diferentes combinaciones de parámetros y se obtienen resultados de rendimiento. A través de la combinación de información previa y los resultados observados, la búsqueda bayesiana infiere la distribución de probabilidades sobre los hiperparámetros y sugiere la siguiente combinación

de parámetros a evaluar, basándose en el principio de selección óptima de acuerdo con la incertidumbre actual.

Una de las ventajas de este método es la capacidad que tiene de lidiar con la falta de datos o ruido en las evaluaciones del rendimiento, ya que considera la incertidumbre inherente a los resultados observados. Esto la convierte en una técnica ideal para optimizar los algoritmos genéticos, donde la evaluación precisa del rendimiento es difícil de conocer, ya que tiene cierta aleatoriedad. Por otra parte, también podemos incluir las probabilidades directamente como valores continuos a optimizar, planteándolos como rangos, y el algoritmo decidirá que valores intentar.

Para utilizar esta búsqueda bayesiana en Julia, podemos acudir a la librería TreeParzen, a través de la función fmin(), que recibe la función a optimizar, un diccionario con los parámetros y una cantidad de iteraciones, y devolverá la mejor combinación de hiperparámetros que encuentre. La función que le indicaremos a optimizar no será el algoritmo genético directamente, sino que será una función auxiliar que ejecuta la combinación de parámetros 5 veces, lo que dará más robustez a los resultados que consigamos. El valor devuelto a la función será el promedio de los resultados de cada algoritmo genético.

En concreto, los valores que hemos indicado como parámetros se pueden encontrar en la Tabla 10.

Tabla 10. Parámetros de nuestra propuesta de algoritmo genético. Elaboración propia.

Parámetros	**Significado**	**Valores**
nC1	Individuos del grupo C1	10/20/30…90/100
nC2	Individuos del grupo C2	0/2/4/6…18/20
ncalls	Nº de días de llamadas	5/10/15
nchildren	Nº de pares de hijos a generar	1/2/3/…9/10
ngroups_cross	Nº de vecindarios a cruzar	1/2/3/4/5
pmut	Probabilidad de mutación	0%-100%
tournamentSel	% de individuos a seleccionar en el torneo	10%-100%
tournamentPart	Cantidad de participantes en el torneo	2/3/4…9/10
tournamentBestProb	Probabilidad de elegir al mejor individuo en el torneo	70%-100%

La búsqueda bayesiana comenzó probando combinaciones aleatorias de hiperparámetros, pero rápidamente encontró buenas soluciones a partir de la iteración 40, como se puede ver en la Ilustración 36. De hecho, los 10 mejores resultados tienen prácticamente el mismo fitness (Tabla 11), habiéndose encontrado uno de ellos en la iteración 52 y el mejor en la 165, prácticamente al final.

Tabla 11.Top 10 mejores iteraciones de la búsqueda bayesiana en nuestro algoritmo genético. Elaboración propia.

Iteración búsqueda bayesiana	Fitness
165	3.030279
106	3.032313
146	3.032957
62	3.033685
99	3.034766
139	3.035987
115	3.036466
141	3.036530
52	3.038091
65	3.038377

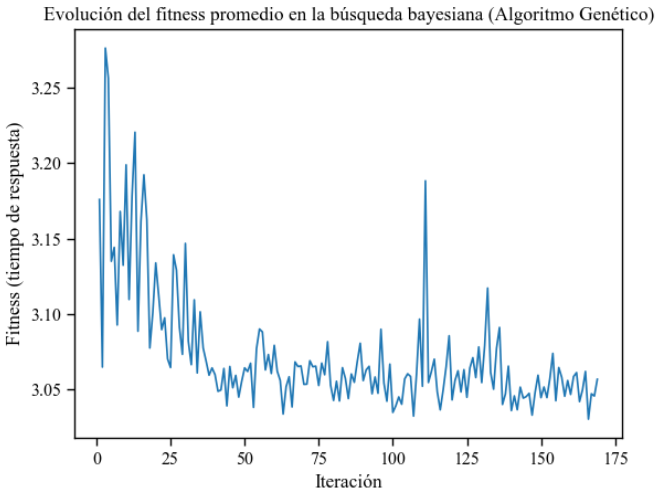

Ilustración 36. Evolución del fitness en la búsqueda bayesiana. Parece que hay un límite alrededor de 3 minutos. Elaboración propia con Python (seaborn).

De las 5 mejores iteraciones se realizará otra prueba con 30 repeticiones, como hemos hecho en el resto de algoritmos, para que la comparativa entre ellos sea justa. Analizando un poco más a fondo estas 5 combinaciones, se puede ver que comparten valores de parámetros similares entre ellos. Concretamente todos sugieren utilizar una población extremadamente pequeña (nC1 = 10), sin individuos con alta disimilitud (nc2 = 0) por lo que esta idea que propusimos no ha surgido efecto. También todos proponen utilizar sólo 5 días de llamadas para el fitness intermedio (ncalls = 5), decantándose por algoritmos más rápidos.

Además de esto, todos comparten la idea de utilizar muchos vecindarios (> 4) para la nueva función de cruce, devolver muchos hijos (>6) por pareja de padres y mutar los individuos con mucha probabilidad (>70%). En el torneo, que participen muchos individuos (>8) en cada "combate" y que se seleccione con una alta probabilidad (>83%) al mejor de todos. Lo único en lo que hay más debate es en la cantidad de individuos a seleccionar para el cruce, en el cual hay valores entre el 12% y el 75% de la población total. Estos datos se pueden ver recogidos en la Tabla 12.

Tabla 12. Parámetros escogidos en las mejores 5 ejecuciones de la búsqueda bayesiana. Elaboración propia en Python (pandas) mediante Visual Studio Code.

	nC1	nC2	ncalls	ngroups_cross	nchildren	pmut	tournamentSel	tournamentPart	tournamentBestProb	fitness
165	10	0	5	5	8	0.868741	0.725080	10	0.850768	3.030279
106	10	0	5	5	9	0.900681	0.417582	9	0.979611	3.032313
146	10	0	5	4	10	0.715194	0.509438	10	0.835597	3.032957
62	10	0	5	4	7	0.779131	0.171332	10	0.970528	3.033685
99	10	0	5	5	9	0.845135	0.119180	10	0.993048	3.034766

Para referenciarlos fácilmente, llamaremos a cada combinación de parámetros D1, D2 … D5. Así por ejemplo el grupo d1 se referirá a las 30 ejecuciones realizadas con los parámetros de la iteración 165 (primera en la Tabla 13).

Tabla 13. Resultados promedio de cada grupo en nuestra propuesta de algoritmo genético. Cada grupo se refiere a una combinación específica de parámetros, los cuales han sido los mejores en la búsqueda bayesiana. Aparentemente, no parece haber diferencia entre los resultados. Elaboración propia.

Grupo	**D1**	**D2**	**D3**	**D4**	**D5**
Fitness	3.09	3.072	3.093	3.081	3.129
Tiempo que tarda en encontrar la mejor solución	553.32	548.13	558.51	535.90	544.01
Cantidad de iteraciones total en promedio	36.2	46.16	42.46	151.53	68.36

Inmediatamente podemos ver como de forma general, la cantidad de iteraciones total es más baja que en anteriores modelos, y el tiempo que tarda en encontrar la mejor solución también es mucho más alto, prácticamente al final.

En cuanto al fitness, los resultados son muy parecidos, podemos visualizar sus distribuciones en la Ilustración 37.

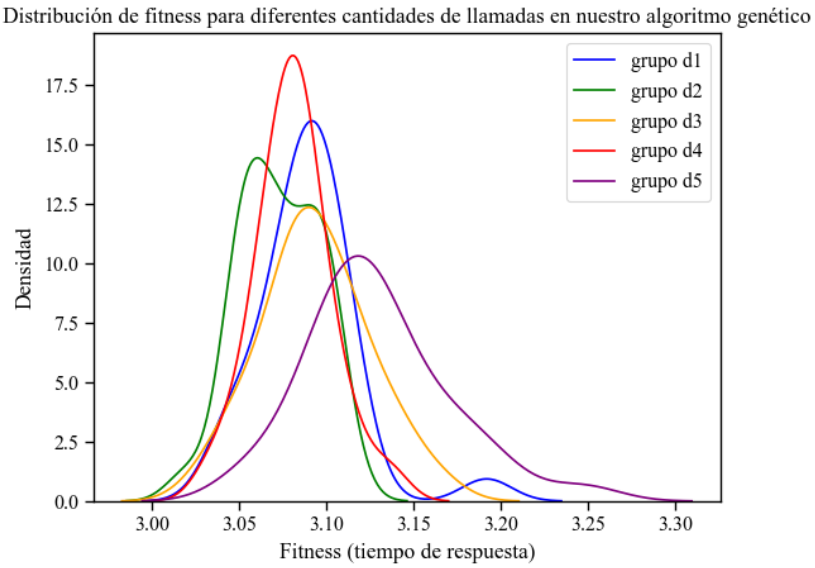

Ilustración 37. Distribución de fitness en las diferentes cantidades de llamadas del primer algoritmo genético. Las medias no parecen mostrar ninguna diferencia significativa, excepto por el grupo 5. Elaboración propia con Python (seaborn).

Curiosamente, después de realizar las pruebas estadísticas, todas las distribuciones siguen una campana de gauss, a excepción del grupo d1. En cuanto a diferencia de medias, el grupo 2 y 4 no son significativamente diferentes, con un p-valor del estadístico t de 0.16. Sin embargo, sí lo son el 2 y 3 (p-valor 0.004), mientras que el 4 del 3 no (p-valor 0.073). Podemos llegar a la conclusión de que el grupo d2 es ligeramente el mejor de todos, aunque hemos visto que los resultados son muy parecidos.

El mejor resultado individual se da en el grupo d2, con 3.0198 minutos, podemos visualizar la evolución de su ejecución, así como la distribución propuesta, en la Ilustración 38.

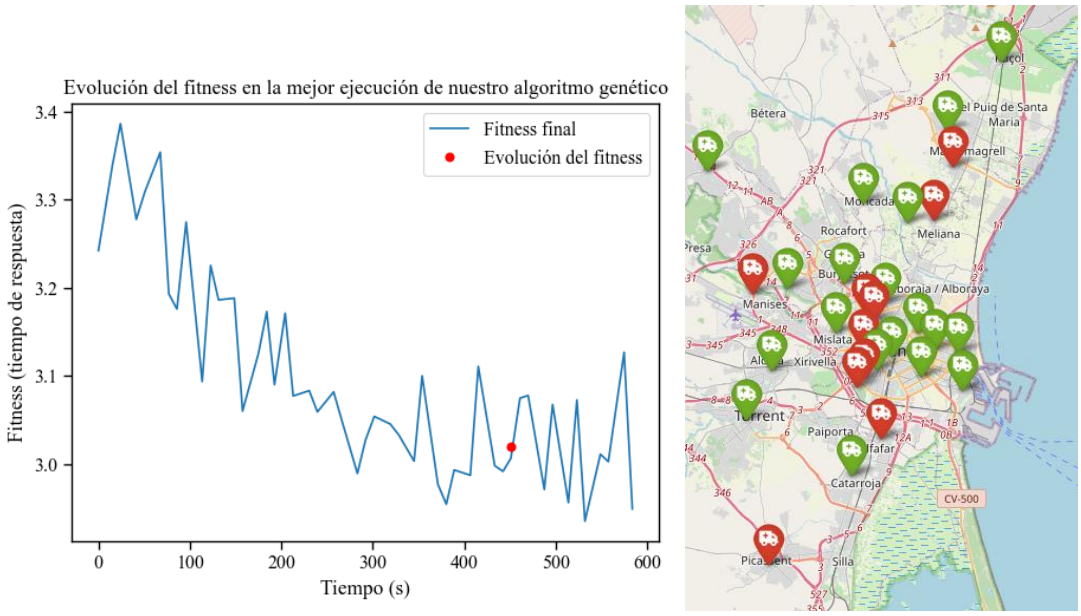

Ilustración 38. A la izquierda, evolución del fitness en la mejor ejecución de nuestra propuesta de algoritmo genético. La mejor solución se encuentra alrededor de los 450 segundos. A la derecha, la distribución de ambulancias propuesta por esta solución. Elaboración propia con Python (seaborn y folium).

La evolución en este caso es mucho más directa que en el anterior genético, alcanzando los límites de 3 minutos en fitness parciales en poco menos de 300 segundos, sin embargo la mejor solución no es encontrada hasta llegados los 450 segundos. Por otra parte la asignación propuesta parece mucho mejor distribuida a lo largo del área metropolitana, aunque la mayoría de SVAs se encuentran de nuevo, en el centro de Valencia.

8.4. Resultados de la búsqueda aleatoria

Para la búsqueda aleatoria o *random search*, también se llevaron a cabo 30 experimentos los cuales se dejaron correr durante 600 segundos. De cada experimento se guardó su evolución a lo largo del tiempo, cada vez que se encontraba una solución mejor que la anterior, se almacenaba el valor del tiempo medio de respuesta y el instante de tiempo en el que había sido encontrado. Los 5 mejores resultados de la búsqueda aleatoria se pueden encontrar en la Tabla 14.

	fitness	phenotype1	phenotype2
5	3.422248	[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0...	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0...
8	3.481883	[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0...	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0...
29	3.494077	[0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0...	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0...
23	3.504656	[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0...	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0...
13	3.510183	[0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0...	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1...

Tabla 14. Top 5 mejores resultados de la búsqueda aleatoria. Elaboración propia en Python (pandas) mediante Visual Studio Code.

El valor promedio del tiempo medio de respuesta fue de 3.598 minutos, aproximadamente 3 minutos y 35 segundos, y el mejor resultado, 3.42 minutos. De la mejor ejecución podemos visualizar la evolución del tiempo medio de respuesta, en la Ilustración 39. En este caso se

encontró la mejor solución justo antes de llegar a los 400 segundos de cómputo, y como se evalúa con todas las llamadas, coincide con el tiempo medio de respuesta final.

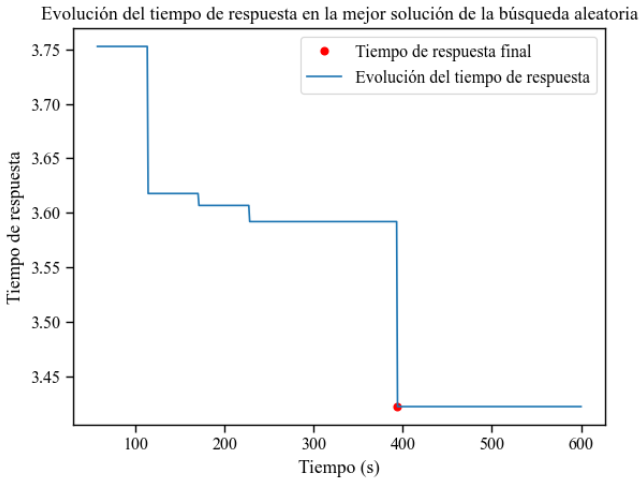

Ilustración 39. Evolución del tiempo medio de respuesta en el mejor resultado de la búsqueda aleatoria. Elaboración propia con Python (seaborn).

La mejor distribución de ambulancias propuesta por la búsqueda aleatoria la podemos visualizar en la Ilustración 40. Los vehículos están esparcidos por toda el área metropolitana, pero si nos fijamos especialmente en las SVAs, podemos ver cómo no hay ninguna en la zona centro-sur del mapa, están prácticamente todas hacia el norte excepto una que ha caído en Picassent. En general se nota como es una asignación encontrada dentro de una búsqueda aleatoria de soluciones, ya que parece carecer de cierto criterio a la hora de colocar las ambulancias.

Ilustración 40. Distribución de ambulancias propuesta por la búsqueda aleatoria. Elaboración propia con Python (folium).

8.5. Comparación de resultados entre técnicas

Una vez analizados individualmente los diferentes métodos de resolución, podemos llevar a cabo una comparativa entre las técnicas, con el fin de identificar posibles diferencias en los resultados obtenidos. Específicamente, se seleccionaron las combinaciones de hiperparámetros más óptimas para cada modelo, y se van a comparar sus distribuciones con el propósito de detectar cualquier disparidad significativa.

En la búsqueda tabú se utilizaron los parámetros neighb_p = 1%, ncalls = 5 y nTabuL = 10, habían otras combinaciones no significativamente diferentes a esta, pero era la más rápida en encontrar solución. Por otra parte en el primer algoritmo genético se utilizaron ncalls = 5 y en nuestra propuesta de algoritmo genético se utilizó el grupo D2. La Ilustración 41 refleja los resultados obtenidos durante la experimentación.

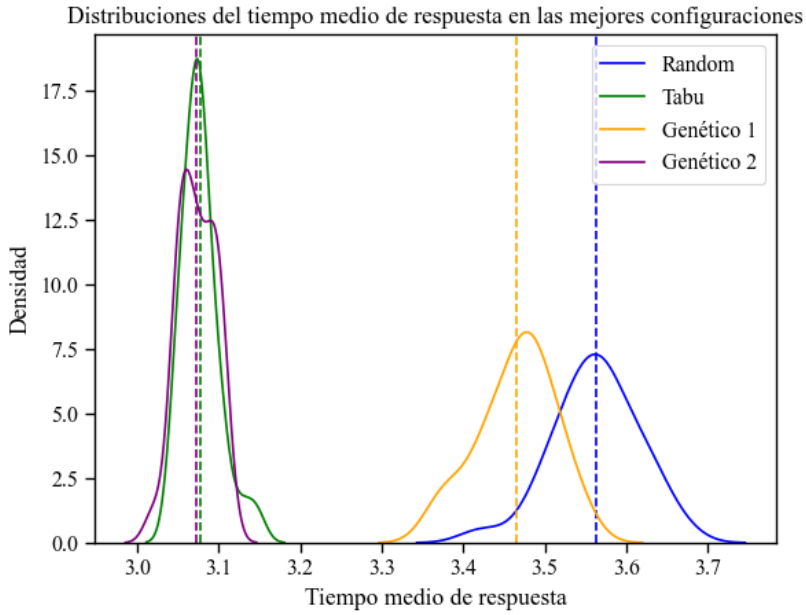

Ilustración 41. Comparativa de distribución del tiempo medio de respuesta en la mejor configuración de cada modelo. La búsqueda tabú y nuestro algoritmo genético son muy similares, mientras que la búsqueda aleatoria y el algoritmo genético de referencia dan peores resultados. Las líneas verticales indican la media de la distribución. Elaboración propia con Python (seaborn).

Los resultados del algoritmo genético que hemos construido, y la búsqueda tabú, son prácticamente idénticos, dando a pensar que 3.07, que es la media aproximada de ambos resultados, puede ser el mínimo teórico de los modelos, cuando el límite de cómputo son 600 segundos.

Si realizamos los tests de shapiro-wilk, todas las distribuciones siguen una campana de gauss, y mediante el estadístico t-student confirmamos que las medias obtenidas por la búsqueda tabú y el algoritmo genético no son significativamente diferentes, con un p-valor de 0.42. Por otra parte, la búsqueda aleatoria sí es significativamente peor que el primer algoritmo genético, con un p-valor de $2.5x10^{-10}$.

A la hora de elegir el mejor modelo, podemos comparar el tiempo que tarda en encontrar la solución, para ver si en este caso hubiera diferencias significativas. Los resultados se pueden visualizar en la Ilustración 42.

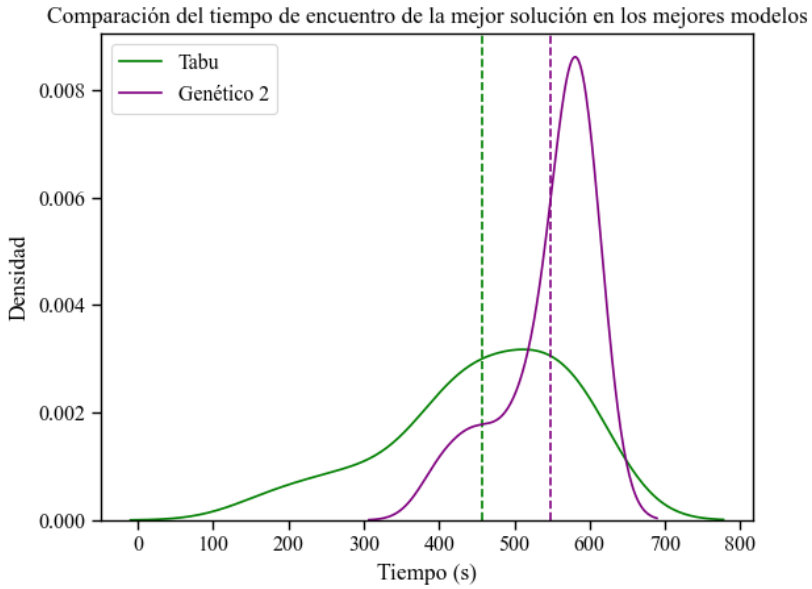

Ilustración 42. Comparativa de distribución del tiempo de encuentro de la mejor solución en el mejor modelo de algoritmo genético y búsqueda tabú. La búsqueda tabú tiene un tiempo de encuentro más disperso, pudiendo encontrar soluciones tanto más tarde como mucho más pronto. Elaboración propia con Python (seaborn).

El algoritmo genético encuentra casi todas sus mejores soluciones hacia el final del modelo, dejando la media en 550s, mientras que en la búsqueda tabú la media se queda en 450s. Esto sugiere que generalmente la búsqueda tabú será más rápida, y, por lo tanto, mejor modelo, ya que el tiempo promedio a las llamadas de emergencia era argumentablemente idéntico.

Es importante destacar cómo la búsqueda tabú tiene resultados tan dispares en cuanto al tiempo de encuentro de la mejor solución, al mismo tiempo que el valor medio no tiene prácticamente varianza. Esto confirma que los resultados dependen altamente de la solución inicial escogida, sin embargo, que siempre se encuentra con el mismo límite de 3.07 minutos, que parece ser el mínimo con nuestros modelos.

Por otra parte, la mejor distribución de ambulancias individual, en cuanto a tiempo de respuesta, ha sido la encontrada por el algoritmo genético, de 3 minutos y 1.14 segundos (Ilustración 35). Sin embargo, es posible que la búsqueda tabú encuentre una solución con un tiempo de respuesta similar, si realizáramos más experimentos.

Capítulo 9. Conclusiones

Tras realizar los experimentos y al finalizar el proyecto, hemos descubierto varias cosas acerca del sistema de emergencias del área metropolitana de valencia. En lo que a modelos metaheurísticos respecta, tanto la búsqueda tabú como el algoritmo genético consiguen resultados significativamente iguales, llegando a un límite en 3.07 minutos, eligiendo los parámetros ideales. Sin embargo, la búsqueda tabú consigue el resultado promedio en menos tiempo, ya que no encuentra una solución mejor durante los últimos 150-200 segundos de su ejecución. En cambio, el algoritmo genético prácticamente encuentra la solución en las últimas iteraciones.

Debido a la forma en la que hemos construido los métodos, utilizando solo un % de las llamadas en cada iteración, estos resultados son injustos para el algoritmo genético. Esto es debido a que, si nos fijamos en la evolución de alguno de los resultados, como la mostrada en la Ilustración 38, se alcanza el resultado límite, de alrededor de 3.07 minutos, mucho antes, sin embargo, la solución exacta se pierde ya que sigue haciendo ejecuciones. La mejor solución deja de serlo al ser evaluada con otro conjunto de llamadas, y pasa a serlo otra distribución, aunque lo sea por 1 sólo segundo. Quizá en las iteraciones posteriores vuelve a encontrar la misma distribución solución que antes, pero esta vez mucho más adelantada en el tiempo, dando la sensación de que tarda más.

Este problema no ocurre en la búsqueda tabú, ya que, una vez encontrada una distribución mejor, su fitness no se vuelve a actualizar en cada iteración, si no que se mantiene, aunque en iteraciones posteriores las poblaciones se evalúen con otro conjunto de llamadas.

En este sentido, el fallo principal ha sido no utilizar otro método de parada de los algoritmos, por ejemplo, dar un margen de fallo a la mejor solución encontrada, para que no se actualice a no ser que otra solución sea significativamente mejor. Debido a la gran cantidad de posibles distribuciones y cuan similares pueden llegar a ser unas con otras, sobre todo si los evaluamos cada vez con un conjunto de llamadas, es necesario detener el algoritmo antes de llegar a un límite de tiempo.

Por otra parte, tanto el algoritmo genético que hemos planteado como la búsqueda tabú han dado muy buenos resultados, siendo mucho mejores que la búsqueda aleatoria y que el algoritmo genético de referencia que teníamos. En el momento de construir un modelo de optimización para distribución de ambulancias, cuando tienes variables tan diferentes dependiendo de la ciudad y el sistema de emergencias, no se puede simplemente copiar uno que haya funcionado otro lugar. Es necesario pensar detenidamente las necesidades y características de tu sistema, como hemos realizado nosotros, utilizando un subconjunto de llamadas, y limitando el vecindario, entre otras estrategias.

Cómo última conclusión, este estudio ha demostrado que el sistema de emergencias sanitarias del área metropolitana de Valencia podría mejorar su tiempo medio de respuesta. El actual modelo responde a las emergencias en 4 minutos y 4 segundos, de forma promedia, mientras que el mejor modelo ha conseguido responder a las emergencias en 3 minutos y 1 segundo, una importante mejora. Hay que tener en cuenta que estos valores son los proporcionados por el simulador JEMSS, que no tiene en cuenta factores como el tráfico o el

momento del día a la hora de decidir la velocidad de los vehículos, por lo que los valores reales pueden ser un poco superiores.

9.1. Legado

Los datos utilizados sobre las llamadas son privados dentro del proyecto iReves, pero el resto de datos, los modelos utilizados y el código queremos que queden disponibles para cualquier persona que esté interesada en emplearlos. Todo lo que podemos compartir quedará subido a un repositorio de gitlab abierto al público. El link al repositorio está disponible en este enlace.

En concreto, estarán disponibles los datos de ambulancias y estaciones, todo el código utilizado para la creación de los modelos y todo el código utilizado para la experimentación de los modelos. También el código utilizado para generar los gráficos con Python, todos los resultados de los modelos, y el proceso para instalar julia y todas las librerías, y los cambios que hemos hecho, en un archivo llamado "Entorno.txt".

9.2. Relación con estudios cursados

A pesar de que muchas de las cosas que hemos necesitado para llevar a cabo este proyecto han tenido que ser aprendidas a medida que avanzábamos, hay varias asignaturas del Grado en Ciencia de Datos que han sido clave cursar y entender para poder hacer mejor este Trabajo de Fin de Grado:

- **Optimización:** La más relacionada con el trabajo, en Optimización nos enseñan qué son las técnicas metaheurísticas desde 0, y con hincapié en los algoritmos genéticos. También se desarrolla un trabajo en equipo donde hay que diseñar tu propio algoritmo genético para una tarea de optimización.
- **Visualización:** En esta asignatura enseñaron a cómo representar la información de forma correcta, aprovechando los colores y las formas de los gráficos.
- **Proyecto III:** La asignatura de proyecto III fue lo más cercano a un trabajo profesional que habíamos hecho hasta el momento, aunque se trabaja en equipo, aprendes a investigar, descubrir información por ti solo o preguntando, y creo que ha sido clave para saber qué íbamos a esperar de un TFG.
- **Programación:** Aunque la mayoría de la programación de este trabajo se hace en Julia, un lenguaje que no vemos en la carrera, las bases las aprendemos en esta asignatura con Python, especialmente a buscar soluciones cuando no nos funciona o no sabemos continuar (por falta de conocimiento). Con Julia en este proyecto me he sentido como con Python en los primeros años, con la ventaja de que ya sabía cómo manejarme.

Otra de las claves que me ha dado la carrera de ciencia de datos para este TFG es la capacidad de autoaprendizaje, es decir ser capaz de aprender por mí mismo a través de los problemas que me vayan surgiendo, debido a tener que enfrentarme a un nuevo lenguaje de programación que nadie me ha enseñado. Va ligada de otras cosas como la tolerancia a la frustración y la resolución de problemas, en esta carrera te enfrentas a muchos trabajos donde tienes que ser capaz de buscar información y no darte por vencido por muy difícil que se vea.

Creo que es algo intrínseco de la carrera debido a todos los proyectos que se hacen, aunque es posible que ocurra lo mismo en carreras de ingeniería por la dificultad que suele presentar.

La curiosidad también es una cualidad de buenos científicos de datos, que se va mejorando de forma transversal durante la carrera casi sin darte cuenta, y que ha sido importante en el TFG, en general descubrir cosas nuevas e investigar sirven de motivación para los trabajos de análisis y ciencia de datos

9.3. Líneas de trabajo futuras

En este trabajo teníamos como objetivo minimizar el tiempo de respuesta a las emergencias, buscando una distribución de ambulancias que fuera lo más rápida posible. Aunque este es un buen punto para empezar y mejorar, es posible que no sea un buen objetivo.

Utilizar la media de todas las respuestas a las emergencias puede engañar, ya que puede haber emergencias que sean respondidas en 1 minuto y puede haber otras donde tarden 5 minutos. Las llamadas ocurren sobre todo por el centro de valencia, y en los resultados de los experimentos hemos observado que es donde generalmente han caído la mayoría de las ambulancias, dejando un poco abandonados a los pueblos de alrededor. Es posible que esto esté creando una falsa percepción de “buena” distribución cuando en realidad no lo es.

En un futuro se podría plantear la posibilidad de realizar un estudio similar a este, pero teniendo en cuenta la desviación típica de las respuestas, o ponderando el tiempo de respuesta por un peso, penalizando aquellas que se pasen de un límite. Esto lo descubrimos a medida que realizábamos el proyecto, veíamos el mapa de calor de las llamadas y sobre todo con los resultados de los experimentos. En general creemos que los resultados pueden ser diferentes si se optimiza una métrica que evalúe mejor cómo de buena es una distribución de ambulancias.

Nos hubiera gustado también realizar experimentos para averiguar la mejor distribución de ambulancias en diferentes momentos del año. Por ejemplo, mantener una distribución de septiembre a mayo y otra en los meses de verano. Valencia es una ciudad que recibe mucho turismo en junio, julio y agosto, además de que muchas personas se desplazan a apartamentos costeros o se van de viaje. Es posible que la distribución de la población cambie y con ello los lugares donde más emergencias hay. Creemos que esta sería otra mejora al trabajo y podría ayudar también a la Conselleria de Sanitat a colocar mejor los vehículos de emergencia por la provincia.

Referencias

Alexis J. Comber, S. S. (2011). A modified grouping genetic algorithm to select ambulance site locations. *International Journal of Geographical Information Science, 25*(5), 807-823. doi:https://doi.org/10.1080/13658816.2010.501334

Ana Paula Iannoni, R. M. (January de 2008). A hypercube queueing model embedded into a genetic algorithm for ambulance deployment on highways. *Annals of Operations Research, 157*(1), 207-224. doi:https://doi.org/10.1007/s10479-007-0195-z

Carol A. Markowski, E. P. (1990). Conditions for the Effectiveness of a Preliminary Test of Variance. *The American Statistician*, 322-326. doi:doi:10.2307/2684360

Conover, W. J. (1999). *Practical nonparametric statistics.*

ConSalud. (16 de 11 de 2020). ¿Quiénes son los mejores médicos cardiólogos de España? *ConSalud*. Obtenido de https://www.consalud.es/profesionales/quienes-mejores-medicos-cardiologos-espana_87894_102.html

Constantine Toregas, R. S. (1971). The Location of Emergency Service Facilities. *Operations Research, 9*(6), 1363-1373. doi:http://dx.doi.org/10.1287/opre.19.6.1363

García Vecina, M. Á. (2022). *Optimización del problema de asignación de vehículos de emergencia sanitaria en la provincia de Valencia.* Valencia: Tesis de Máster.

Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*, 533-549.

Glover, F. (1997). *Tabu Search.* Kluwer Academic Publishers.

Håkon Leknes, E. S. (2017). Strategic ambulance location for heterogeneous regions. *European Journal of Operational Research, 260*(1), 122-133. doi:https://doi.org/10.1016/j.ejor.2016.12.020.

Haldun Aytug, C. S. (2002). Solving large-scale maximum expected covering location problems by genetic algorithms: A comparative study. *European Journal of Operational Research, 141*(3), 480-494. doi:https://doi.org/10.1016/S0377-2217(01)00260-0

Henderson, S. &. (2000). BartSim: A tool for Analysing and Improving Ambulance Performance in Auckland, New Zealand. *Proceedings of the 35th annual conference of the operational research society of New Zealand*, (págs. 57-64). Wellington, New Zealand.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.*

Intergeneracional, S. (26 de 10 de 2018). ESPAÑA REPITE COMO EL TERCER MEJOR SISTEMA SANITARIO DEL MUNDO. *Solidaridad Intergeneracional*. Obtenido de https://solidaridadintergeneracional.es/wp/espana-repite-como-el-tercer-mejor-sistema-sanitario-del-mundo/

Jeff Bezanson, S. K. (24 de September de 2012). Julia: A Fast Dynamic Language for Technical Computing. *arXiv*. doi:arXiv:1209.5145

John F. Repede, J. J. (30 de June de 1994). Developing and validating a decision support system for locating emergency medical vehicles in Louisville, Kentucky. *European Journal of Operational Research, 3*, 567-581. doi:https://doi.org/10.1016/0377-2217(94)90297-6

Joseph Tassone, G. P. (2020). Algorithms for optimizing fleet staging of air ambulances. *Array, 7*, 100031. doi:https://doi.org/10.1016/j.array.2020.100031

Lu Zhen, K. W. (2014). A simulation optimization framework for ambulance deployment and relocation problems. *Computers & Industrial Engineering, 72*, 12-23. doi:https://doi.org/10.1016/j.cie.2014.03.008

Mahdi Moeini, Z. J. (2013). An Integer Programming Model for the Dynamic Location and Relocation of Emergency Vehicles: A Case Study. *12th International Symposium on Operational Research (SOR)*, (págs. 343-350). Dolenjske Toplice, Slovenia. doi:https://doi.org/10.48550/arXiv.1404.3285

Mehrdad Kaveh, M. S. (2019). Improved biogeography-based optimization using migration process adjustment: An approach for location-allocation of ambulances. *Computers & Industrial Engineering, 135*, 800-813. doi:https://doi.org/10.1016/j.cie.2019.06.058

Michel Gendreau, G. L. (1997). Solving an ambulance location model by tabu search. *Location Science, 5*(2), 75-88. doi:https://doi.org/10.1016/S0966-8349(97)00015-6

Mohammad Maleki, N. M. (2014). Two new models for redeployment of ambulances. *Computers & Industrial Engineering, 78*, 271-284. doi:https://doi.org/10.1016/j.cie.2014.05.019

Moon, I. C. (2015). EMSSIM: emergency medical service simulator with geographic and medical details. *Winter Simulation Conference (WSC)*, (págs. 1272-1284). Huntington Beach, CA.

Nuşin Uncu, M. K. (14 de 9 de 2022). An improved EMS simulation-optimization model with Poisson mixture distribution. *Journal of Engineering Research*. doi:https://doi.org/10.36909/jer.17605

ReVelle, R. C. (December de 1974). The maximal covering location problem. *Papers of the Regional Science Association, 32*, 101-118. doi:https://doi.org/10.1007/BF01942293

Richard McCormack, G. C. (2015). A simulation model to enable the optimization of ambulance fleet allocation and base station location for increased patient survival. *European Journal of Operational Research, 247*(1), 294-309. doi:https://doi.org/10.1016/j.ejor.2015.05.040

S.S. Shapiro, M. W. (1 de 12 de 1965). An analysis of variance test for normality (complete samples). *Biometrika, 52*(3-4), 591–611. doi:doi:10.1093/biomet/52.3-4.591

Samuel Ridler, A. J. (2022). A simulation and optimisation package for emergency medical services. *European Journal of Operational Research, 298*(3), 1101-1113. doi:https://doi.org/10.1016/j.ejor.2021.07.038

Seyyed-Nader Shetab-Boushehri, P. R. (2022). Modeling location–allocation of emergency medical service stations and ambulance routing problems considering the variability of events and recurrent traffic congestion: A real case study. *Healthcare Analytics, 2*, 100048. doi:https://doi.org/10.1016/j.health.2022.100048

Student. (3 de 1908). The Probable Error of a Mean. *Biometrika, 6*(1), 1-25. doi:https://doi.org/10.2307/2331554

Tugba Sarac, E. A. (2014). Reducing ambulance response time: A Case Study. *IMSS14-CIE44*, (págs. 76-85). Istanbul, Turkey.

Valerie J. De Maio, I. G. (2003). Optimal defibrillation response intervals for maximum out-of-hospital cardiac arrest survival rates. *Annals of Emergency Medicine, 42*(2), 242-250. doi:https://doi.org/10.1067/mem.2003.266.

Van Buuren, M. v. (2012). Evaluating dynamic dispatch strategies for emergency medical services: TIFAR simulation tool. *Proceedings of the 2012 Winter Simulation Conference (WSC)*, (págs. 1-12). Berlin, Germany. doi:10.1109/WSC.2012.6465214

Wilcoxon, F. (12 de 1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin, 1*(6), 80-83. doi:https://doi.org/10.2307/3001968

Wilf, H. S. (2005). *Generatingfunctionology*. New York. doi:https://doi.org/10.1201/b10576

Yulia Karpova, F. V. (2023). Heuristic algorithms based on the isochron analysis for dynamic relocation of medical emergency vehicles. *Expert Systems with Applications, 212*, 118773. doi:https://doi.org/10.1016/j.eswa.2022.118773

Zaheeruddin, H. G. (2021). Optimizing Ambulance Deployment using Genetic Algorithm. *9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, (págs. 1-5). Noida, India. doi:10.1109/ICRITO51393.2021.9596352

Anexos

Cambios en el simulador

```
function initSim2(sim_copy::Simulation, calls, ambulances, doPrint::Bool = false)
    t = 0.0
    printInitMsg(msg) = doPrint && (t = time(); print(msg))
    printInitTime() = doPrint && println(": ", round(time() - t, digits = 2), " seconds")

    sim = Simulation()
    printInitMsg("reading config file data")
    sim.ambulances = ambulances
    sim.hospitals = sim_copy.hospitals
    sim.stations = deepcopy(sim_copy.stations)
    sim.calls, sim.startTime = calls, 0.0
    sim.time = sim.startTime
    sim.numAmbs = length(sim.ambulances)
    sim.numCalls = length(sim.calls)
    sim.numHospitals = length(sim.hospitals)
    sim.numStations = length(sim.stations)
    sim.net = sim_copy.net
    sim.map = sim_copy.map
    (sim.targetResponseDurations, sim.responseTravelPriorities) = sim_copy.targetResponseDurations, sim_copy.responseTravelPriorities
    sim.travel = sim_copy.travel
    printInitMsg("placing nodes in grid")
    sim.grid = sim_copy.grid
    printInitMsg("adding ambulances, calls, etc")
    sim.eventList = Vector{Event}()
    addEvent!(sim.eventList, sim.calls[1])
    for a in sim.ambulances
        initAmbulance!(sim, a)
    end
    for station in sim.stations
        station.numIdleAmbsTotalDuration = OffsetVector(zeros(Float,sim.numAmbs+1), 0:sim.numAmbs)
        station.currentNumIdleAmbsSetTime = sim.startTime
    end
    printInitMsg("storing times between fNodes and common locations")
    printInitMsg("storing times between fNodes and common locations")
    sim.addCallToQueue! = sim_copy.addCallToQueue!
    sim.findAmbToDispatch! = sim_copy.findAmbToDispatch!
    sim.redispatch = sim_copy.redispatch
    sim.moveUpData = sim_copy.moveUpData
    sim.stats = deepcopy(sim_copy.stats)
    sim.initialised = true
    if sim.numReps >= 1
        makeRepsRunnable!(sim)
    end
    return sim
end
```

Ilustración 43. Nueva función de simulación para acelerar el proceso de creación de un objeto Simulation en JEMSS. Elaboración propia en Julia mediante Visual Studio Code.

Volver al Capítulo: *5.2. Cambios en el simulador.*

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**	X			
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**		X		
ODS 10. **Reducción de las desigualdades**.				
ODS 11. **Ciudades y comunidades sostenibles**.		X		
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

En 2015, los Estados Miembros de las Naciones Unidas aprobaron los 17 Objetivos de Desarrollo Sostenible (ODS) como parte de la Agenda 2030. Estos objetivos tienen como propósito abordar la pobreza, proteger el planeta y mejorar la calidad de vida y las perspectivas de las personas en todo el mundo. En el contexto de este Trabajo de Fin de Grado (TFG), se identifican conexiones directas entre la investigación y varios de los ODS debido al ámbito en el que se desarrolla.

Salud y bienestar.

El tercer objetivo de desarrollo sostenible se centra en la salud y el bienestar, específicamente en garantizar una vida saludable y promover el bienestar para todas las edades. En este Trabajo de Fin de Grado (TFG), se busca mejorar la distribución actual de ambulancias en el área metropolitana de Valencia, con el objetivo de proporcionar una respuesta más rápida a las emergencias que la asignación actual.

La mejora en la distribución de ambulancias tiene un impacto significativo en la atención médica de emergencia. Al reducir los tiempos de respuesta, se logra una asistencia más rápida a los pacientes que requieren atención médica urgente. Esto es especialmente crucial en situaciones críticas donde cada minuto cuenta y puede marcar la diferencia entre la vida y la muerte. Al garantizar una respuesta más rápida y eficiente a las emergencias, este trabajo contribuye a mejorar la calidad de vida de las personas y a mejorar su bienestar.

Industria, innovación e infraestructuras

El noveno objetivo de desarrollo sostenible se relaciona con la industria, la innovación y las infraestructuras. Este trabajo se centra especialmente en mejorar la utilización de las actuales infraestructuras del área metropolitana de Valencia, aprovechando la gran cantidad de estaciones posibles para proponer una distribución de ambulancias mejor que la actual.

Este TFG podría ayudar a la Conselleria de Sanitat a considerar remodelar el actual modelo de respuestas a emergencias, asignando los recursos de los que dispone a diferentes infraestructuras, ya que podría acudir a los pacientes más rápidamente.

Ciudades y comunidades sostenibles

El undécimo objetivo de desarrollo sostenible se enfoca en lograr que las ciudades y las comunidades sean más sostenibles. En esta investigación, se aborda la distribución de ambulancias en el área metropolitana de Valencia, una comunidad urbana. Al mejorar la asignación de ambulancias y reducir los tiempos de respuesta en situaciones de emergencia, se contribuye a la creación de una comunidad más segura y sostenible. Además, una distribución eficiente de los recursos de emergencia puede reducir el impacto ambiental al minimizar el tiempo y los desplazamientos innecesarios.