Universidad Zaragoza
1542

Trabajo Fin de Grado

Optimización de ruta de reparto de material sanitario para los centros de salud de Zaragoza

Delivery route optimization of medical supplies for health centers in Zaragoza

Autor

Iván Sevilla Antón

Directores

Luis Mariano Esteban Escaño

César Asensio Chaves

Escuela Universitaria Politécnica La Almunia

Junio 2023

Página intencionadamente en blanco.

Escuela Universitaria Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

ESCUELA UNIVERSITARIA POLITÉCNICA
DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

MEMORIA

Optimización de ruta de reparto de material sanitario para los centros de salud de Zaragoza

Delivery route optimization of medical supplies for health centers in Zaragoza

425.22.35

Autor: Iván Sevilla Antón

Directores: Luis Mariano Esteban Escaño y César Asensio Chaves

Fecha: 06/06/2023

Página intencionadamente en blanco.

INDICE DE CONTENIDO BREVE

INDICE DE CONTENIDO

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

1. RESUMEN

Este Trabajo Final de Grado (TFG) plantea y comprende la creación de una aplicación web que permite optimizar la ruta de reparto de material sanitario para los centros de salud de Zaragoza, incluyendo como elemento diferenciador en la generación de rutas los algoritmos genéticos.

El SALUD (Servicio Aragonés de Salud), es el sistema sanitario público de la comunidad autónoma de Aragón. Está organizado territorialmente en ocho áreas o sectores de salud, los cuales a su vez se subdividen en zonas de salud; que albergan los distintos centros de salud aragoneses.

Actualmente, en la provincia de Zaragoza, existen un total de sesenta y cinco centros de salud, los cuales son abastecidos desde la Plataforma Logística del SALUD, situada en la Plataforma Logística de Zaragoza (PLAZA).

La aplicación creada resuelve y calcula eficientemente la ruta de reparto necesaria para abastecer los centros de salud requeridos en cada momento, desde la Plataforma Logística del SALUD.

En este contexto es inevitable mencionar el Problema del Agente Viajero o en inglés Travelling Salesman Problem (TSP). Se trata de un problema de combinatoria con alta dificultad que trata de resolver o calcular la ruta más corta que, dada una lista de ciudades y partiendo desde una de ellas, visite todas las restantes una sola vez, volviendo a la ciudad de partida. En este TFG no se visitan ciudades, sino que se visitan cada uno de los centros de salud que se necesiten abastecer.

Esta complejidad del TSP lo clasifica como un problema NP-Hard, es decir, no se puede encontrar la solución óptima en tiempo polinomial, o computacionalmente razonable. Debido a esta situación los algoritmos exactos son inaplicables ya que para un número relativamente "grande" de centros de salud, el tiempo de cálculo se dispara a millones de años.

Como solución a esta complejidad, se utilizan algoritmos heurísticos. En este TFG, concretamente, se utiliza el algoritmo del vecino más cercano, el algoritmo del vecino más lejano y un algoritmo de tipo genético; los cuales, en un tiempo razonable obtienen soluciones muy próximas a la óptima. En definitiva, el TSP se resuelve de una manera muy aproximada, y no exacta; ya que es mejor tener una solución (ruta), que ninguna.

Mientras que el algoritmo del vecino más cercano y el algoritmo del vecino más lejano, operan de una manera muy intuitiva, eligiendo en cada paso local la mejor opción (la ciudad más cercana y la más lejana, a la última incluida en la ruta, respectivamente), el algoritmo genético tiene una mayor complejidad, realizando una búsqueda mucho más profunda y exhaustiva de soluciones óptimas.

El algoritmo genético se basa en la teoría de la evolución de las especies y la selección natural de Charles Darwin. Al igual que los individuos mejor adaptados de las especies, tienen más posibilidades de sobrevivir, reproducirse y generar descendencia, transmitiendo sus características a la posterior generación; las mejores soluciones de cada generación calculadas por el algoritmo genético tienen más posibilidades de perdurar, combinarse y obtener mejores "soluciones hijas", que pasen de generación en generación, hasta obtener la mejor solución posible.

Esta metodología en la búsqueda de soluciones óptimas, tal y como se podrá observar en los resultados de rutas obtenidas, supone una mejora en las soluciones generadas respecto al algoritmo del vecino más cercano y del más lejano.

La aplicación creada en este TFG muestra los resultados de las rutas obtenidas con los tres algoritmos mencionados, por lo que el usuario que la esté utilizando, podrá elegir la que más le convenga; seguramente basándose en el criterio de minimizar la distancia recorrida por la ruta. Además, se representa en un mapa interactivo la ruta más corta generada, mostrando el recorrido a seguir, a través de las calles y carreteras que conectan los centros de salud seleccionados.

Con la finalidad de fomentar el uso de software libre, la aplicación queda depositada en la nube, en un repositorio shiniapps.io para su uso libre de cualquier interesado.

1.1. PALABRAS CLAVE

Problema del Agente Viajero (TSP), centros de salud, optimización de rutas, algoritmos heurísticos, aplicación web.

2. ABSTRACT

This Final Degree Project (TFG) proposes and includes the creation of a web application to optimize the route of distribution of medical supplies for health centers in Zaragoza, including genetic algorithms as a differentiating element in the generation of routes.

The SALUD (Aragonese Health Service) is the public health system of the autonomous community of Aragon. It is organized territorially into eight health areas or sectors, which in turn are subdivided into health zones, which house the different health centers in Aragon.

Currently, in the province of Zaragoza, there are a total of sixty-five health centers, which are supplied from the Logistics Platform of SALUD, located in the Logistics Platform of Zaragoza (PLAZA).

The application created solves and efficiently calculates the delivery route needed to supply the health centers required at any given time, from the Health Logistics Platform.

In this context it is inevitable to mention the Travelling Salesman Problem (TSP). This is a combinatorial problem with high difficulty that tries to solve or calculate the shortest route that, given a list of cities and starting from one of them, visits all the remaining ones once, returning to the city of departure. In this TFG you do not visit cities, but visit each of the health centers you need to supply.

This complexity of the TSP classifies it as an NP-Hard problem, so the optimal solution cannot be found in polynomial time, or computationally reasonable. Because of this situation, exact algorithms are inapplicable since for a relatively "large" number of health centers, the computation time shoots up to millions of years.

As a solution to this complexity, heuristic algorithms are used. In this TFG, specifically, the nearest neighbor algorithm, the farthest neighbor algorithm and a genetic type algorithm are used; all of which, in a reasonable time, obtain solutions very close to the optimal one. In short, the TSP is solved in a very approximate way, and not exact; since it is better to have a solution (path), than none at all.

While the nearest neighbor algorithm and the farthest neighbor algorithm operate in a very intuitive way, choosing at each local step the best option (the nearest and farthest city to the last one included in the route, respectively), the genetic algorithm has a higher complexity, performing a much deeper and more exhaustive search for optimal solutions.

The genetic algorithm is based on Charles Darwin's theory of evolution of species and natural selection. Just as the best adapted

individuals of the species are more likely to survive, reproduce and generate offspring, transmitting their characteristics to the next generation; the best solutions of each generation calculated by the genetic algorithm are more likely to endure, combine and obtain better "daughter solutions", which are passed from generation to generation, until the best possible solution is obtained.

This methodology in the search for optimal solutions, as can be seen in the results of the routes obtained, means an improvement in the solutions generated with respect to the nearest and farthest neighbor algorithm.

The application created in this TFG shows the results of the routes obtained with the three mentioned algorithms, so that the user who is using it, can choose the one that suits him best; surely based on the criterion of minimizing the distance traveled by the route. In addition, the shortest route generated is represented on an interactive map, showing the route to follow, through the streets and roads that connect the selected health centers.

In order to promote the use of free software, the application is deposited in the cloud, in a repository shiniapps.io for free use by anyone interested.

2.1. KEY WORDS

Traveling Salesman Problem (TSP), health centers, route optimization, heuristic algorithms, web application.

3. INTRODUCCIÓN

La razón en la que se sustenta la temática de este TFG es el interés de su autor y el auge que están teniendo en la actualidad la optimización de rutas en el ámbito logístico y la utilización de software libre en el ámbito profesional y estudiantil.

La planificación y optimización de las rutas de reparto es una actividad que está en auge debido a los beneficios que proporciona. Esta actividad supone un proceso que requiere la inversión de tiempo y un análisis profundo de ciertas variables. Es por ello que la recopilación de datos como la disponibilidad de conductores y vehículos, sus dimensiones, la mercancía a transportar y los puntos de entrega; se hace imprescindible. Posteriormente, el tratamiento de estos datos se realiza con un software que tenga en cuenta las variables para obtener las rutas de reparto.

El principal motivo por el que las empresas se plantean y planifican sus rutas de reparto es el ahorro de combustible, que al final de año puede suponer obtener o no beneficios. Además, una buena gestión y aprovechamiento de las rutas de reparto permite reducir el número de vehículos necesarios y todos los gastos que esto conlleva.

El ahorro de tiempo derivado de una óptima distribución de los productos es otro de los principales motivos que impulsan a las empresas a unir sus esfuerzos en la búsqueda de rutas, ya que les supone también un ahorro económico y una mayor satisfacción de los clientes.

Por lo tanto, la planificación y optimización de rutas, es una de las actividades más importantes en la logística, ya que posibilita la reducción de costes y un mejor aprovechamiento de los recursos materiales, humanos y del tiempo; suponiendo una ventaja competitiva para la empresa.

A pesar de todos los beneficios expuestos de la planificación estratégica de rutas, siguen existiendo empresas que enfocan sus esfuerzos en otras actividades, como llegar a más clientes y aumentar las ventas, tratando de conseguir mayores beneficios; pero que desconocen, o no le dan la importancia suficiente, al impacto negativo que una mala planificación de las rutas de distribución representa en sus resultados económicos, posicionándose en una clara desventaja competitiva respecto a otras.

El software libre utilizado en este TFG es R, un lenguaje y entorno de programación libre, que permite a los usuarios tener total libertad para utilizarlo, compartirlo, modificarlo y mejorarlo. Es por ello que está en constante evolución gracias a la publicación libre y gratuita de paquetes de software, los cuales permiten la resolución de diferentes problemas.

Además, R ha ganado gran popularidad, consiguiendo un incremento de su uso a nivel profesional y educativo, debido a que tiene una curva de aprendizaje sencilla en comparación con otros lenguajes de programación y permite la realización de múltiples tareas de manipulación de datos.

Es por todo aquello que este TFG pretende destacar la importancia de la planificación y optimización de rutas, promoviendo el uso de software libre y destacando su utilidad para resolver problemas reales y aplicados; a través de la creación de una aplicación libre que permita su uso a todo aquel interesado, con el fin de optimizar la ruta de reparto de material sanitario entre los distintos centros de salud de Zaragoza, en el momento que se necesite.

El uso de esta aplicación supone disponer de todos los beneficios que una buena planificación y optimización de rutas otorga, y que han sido mencionados anteriormente.

4. DESARROLLO

4.1. ANÁLISIS DE LA RED DE APROVISIONAMIENTO

El SALUD (Servicio Aragonés de Salud), es el organismo encargado de la gestión y prestación de los servicios sanitarios públicos de la Comunidad Autónoma de Aragón, en España. Está dividido territorialmente en ocho áreas o sectores, los cuales a su vez están subdivididos en zonas, en las que se encuentran los centros de salud que necesitan ser abastecidos con material sanitario.

Este TFG se focaliza en el suministro de material sanitario desde el almacén central del SALUD, situado en la plataforma logística de Zaragoza (PLAZA), hasta todos los centros de salud pertenecientes a la provincia de Zaragoza.

Cada día se determinarán los centros de salud que necesitan ser abastecidos y la aplicación optimizará la ruta de reparto a seguir.

4.1.1. Áreas y Zonas de salud en Aragón

El territorio aragonés se encuentra dividido en las siguientes áreas o sectores de salud:

- Área de Salud Alcañiz.
- Área de Salud Barbastro.
- Área de Salud Calatayud.
- Área de Salud Huesca.
- Área de Salud Teruel.
- Área de Salud Zaragoza I.
- Área de Salud Zaragoza II.
- Área de Salud Zaragoza III.

A continuación, se presenta un mapa de la comunidad aragonesa en la que se observa el fraccionamiento territorial, en las distintas áreas de salud:

Ilustración 1. Áreas de Salud de Aragón

(ARAGON, 2023)

Tal y como se ha expuesto anteriormente, estas áreas de salud, a su vez, están divididas en zonas de salud tal y como se muestra en el siguiente mapa:

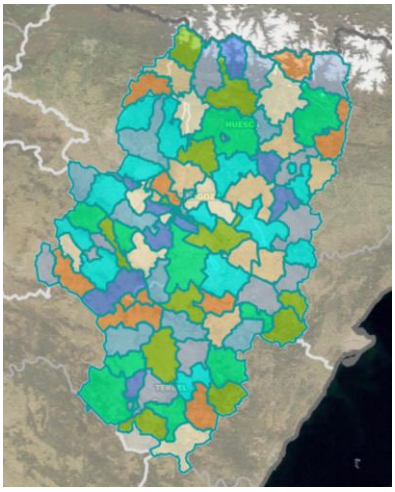

Ilustración 2. Zonas de Salud de Aragón

(ARAGON, 2023)

4.1.2. Centros de salud de Zaragoza

En este apartado, se agrupan en la siguiente tabla los centros de salud de las distintas áreas y zonas, pertenecientes a la provincia de Zaragoza.

CENTROS DE SALUD DE ZARAGOZA		
ÁREA	**ZONA**	**CENTRO DE SALUD**
ALCAÑIZ	CASPE	C.S. AMANDO LORIGA-CASPE
	MAELLA	C.S. MAELLA
BARBASTRO	MEQUINENZA	C.S. MEQUINENZA
CALATAYUD	ALHAMA DE ARAGON	C.S. ALHAMA DE ARAGÓN
	ARIZA	C.S. ARIZA
	ATECA	C.S. ATECA
	CALATAYUD RURAL	C.S. CALATAYUD
	DAROCA	C.S. DAROCA
	ILLUECA	C.S. ILLUECA
	MORATA DE JALON	C.S. MORATA DE JALÓN
	SABIÑAN	C.S. SABIÑAN
	VILLARROYA DE LA SIERRA	C.S. VILLARROYA DE LA SIERRA
ZARAGOZA I	ACTUR NORTE	C.S. ACTUR NORTE
	ACTUR OESTE	C.S. AMPARO POCH (ACTUR OESTE)
	ACTUR SUR	C.S. ACTUR SUR
	ALFAJARIN	C.S. ALFAJARÍN
	ARRABAL	C.S. ARRABAL
	AVENIDA CATALUÑA-LA JOTA	C.S. LA JOTA
	BUJARALOZ	C.S. BUJARALOZ
	LUNA	C.S. LUNA
	PARQUE GOYA	C.S. PARQUE GOYA
	SANTA ISABEL	C.S. SANTA ISABEL
	VILLAMAYOR	C.S. VILLAMAYOR
	ZALFONADA-PICARRAL	C.S. PICARRAL
	ZUERA	C.S. ZUERA
ZARAGOZA II	ALMOZARA	C.S. ALMOZARA
	CAMPO DE BELCHITE	C.S. CAMPO DE BELCHITE
	CASABLANCA	C.S. CASABLANCA
	FERNANDO EL CATOLICO	C.S. FERNANDO EL CATÓLICO
	FUENTES DE EBRO	C.S. FUENTES DE EBRO
	PARQUE ROMA	C.S. PARQUE ROMA
	PUERTA DEL CARMEN	C.S. PUERTA DEL CARMEN
	LAS FUENTES NORTE	C.S. LAS FUENTES NORTE

CENTROS DE SALUD DE ZARAGOZA		
ÁREA	**ZONA**	**CENTRO DE SALUD**
	MADRE VEDRUNA-MIRAFLORES	C.S. JOSE R. MUNOZ FERNANDEZ (SAGASTA)
	REBOLERIA	C.S. REBOLERÍA
	ROMAREDA - SEMINARIO	C.S. ROMAREDA (SEMINARIO)
	SAN JOSE NORTE	C.S. SAN JOSÉ
	SAN JOSE SUR	C.S. CANAL IMPERIAL
	SAN PABLO	C.S. SAN PABLO
	SASTAGO	C.S. SÁSTAGO
	TORRE RAMONA	C.S. TORRE RAMONA
	TORRERO LA PAZ	C.S. TORRERO-LA PAZ
	VALDESPARTERA-MONTECANAL	C.S. VALDESPARTERA
ZARAGOZA III	ALAGON	C.S. ALAGÓN
	BOMBARDA	C.S. BOMBARDA
	BORJA	C.S. BORJA
	CARIÑENA	C.S. CARIÑENA
	CASETAS	C.S. CASETAS
	DELICIAS NORTE	C.S. DELICIAS NORTE
	DELICIAS SUR	C.S. DELICIAS SUR
	EJEA DE LOS CABALLEROS	C.S. VIRGEN DE LA OLIVA-EJEA DE LOS CABALLEROS
	EPILA	C.S. ÉPILA
	GALLUR	C.S. GALLUR
	HERRERA DE LOS NAVARROS	C.S. HERRERA DE LOS NAVARROS
	LA ALMUNIA DE DONA GODINA	C.S. LA ALMUNIA DE DOÑA GODINA
	MARIA DE HUERVA	C.S. MARÍA DE HUERVA
	MIRALBUENO-GARRAPINILLOS	C.S. MIRALBUENO-GARRAPINILLOS
	OLIVER	C.S. OLIVER
	SADABA	C.S. SÁDABA
	SOS DEL REY CATOLICO	C.S. SOS DEL REY CATÓLICO
	TARAZONA	C.S. SAN ATILANO- TARAZONA
	TAUSTE	C.S. TAUSTE
	UNIVERSITAS	C.S. UNIVERSITAS
	UTEBO	C.S. UTEBO
	VALDEFIERRO	C.S. VALDEFIERRO

Tabla 1. Centros de salud de Zaragoza

Tal y como se puede observar, existen un total de 65 centros de salud localizados en la provincia de Zaragoza, pertenecientes a distintas áreas y zonas de salud.

Estos son los centros de salud para los que se realiza aplicación que sustenta este TFG y que permite la optimización del reparto de material sanitario.

4.2. Problema del Agente Viajero o TSP

Para plantear la optimización de la ruta de material sanitario para los centros de salud de Zaragoza, es inevitable mencionar, como aspecto teórico y práctico, el problema del agente viajero o en inglés Travelling Salesman Problem (TSP). Se trata de un problema de combinatoria con alta dificultad que trata de resolver o calcular la ruta más corta que, dada una lista de ciudades y partiendo desde una de ellas, visite todas las restantes una sola vez, volviendo a la ciudad de partida.

En el caso de este TFG las ciudades que se desean visitar representan cada uno de los centros de salud de Zaragoza, y la ciudad de partida y destino es el almacén desde el que se distribuye el material sanitario.

A pesar de que su planteamiento sea sencillo, al ser un problema de combinatoria tiene una alta dificultad cuando se trata de rutas que conectan un número considerable de ciudades o puntos. Esta complejidad ha despertado desde los años cincuenta un gran interés e iniciativa en la mejora de la eficiencia del cálculo de rutas. Esto se debe a que, hasta el momento, la única manera de calcular la ruta óptima "perfecta" supone ensayar todas y cada una de las posibles rutas entre las ciudades, medir las distancias, y elegir la distancia más corta como solución al problema. Este proceso puede suponer millones de años de cálculo para un ordenador cuando el número de puntos a unir se asemeja a los de este TFG.

Es por ello que, la solución a este problema, a día de hoy, consiste en la obtención de soluciones aproximadas a la ruta óptima utilizando la heurística como medio para guiar a un algoritmo en la obtención de soluciones muy próximas a la óptima, pero en un tiempo razonable.

Por lo tanto, la resolución del TSP, cuando se trata de generar rutas que unen un número considerablemente alto de ciudades, no trata de encontrar la solución perfecta, sino una lo más próxima posible a ella.

4.2.1. Formulación matemática del TSP

La relación del TSP con la teoría de grafos permite que este problema se describa de la siguiente manera:

- Considerar $G = (N , A)$ un grafo completo, es decir, que cada par de nodos está conectado por una arista; en el que $N = 1, 2, \ldots, n$ son los nodos o ciudades, y A las aristas o arcos que las unen.
- El nodo $i = 1$ es considerado como la ciudad de origen de la ruta, la cual también será el destino; y los demás nodos son el resto de las ciudades que se visitan una sola vez.
- Cada arco (i , j) tiene asociado un valor de distancia o peso no negativo d_{ij} entre el vértice i y el j.

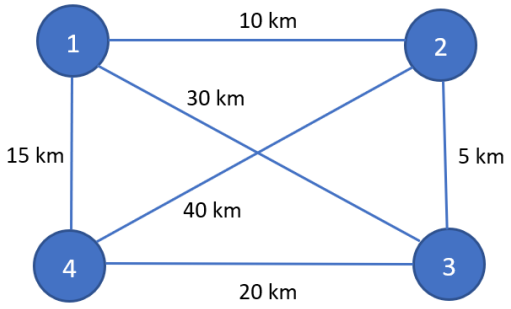

Ilustración 3. TSP como un grafo.

La solución a este problema consiste en encontrar aquel circuito hamiltoniano, de peso mínimo, que pase por cada uno de los nodos o ciudades una sola vez, volviendo a la ciudad de origen.

Según los pesos o distancias entre ciudades, el TSP puede ser:

- Asimétrico: cuando dij ≠ dji para algún par de ciudades. Esto sucede cuando la matriz de distancias es asimétrica debido a que para algún par de ciudades la distancia de ida es distinta a la de vuelta.
- Simétrico: cuando dij = dji para todo par de ciudades. Esto sucede cuando la matriz de distancias es simétrica debido a que para cada par de ciudades la distancia de ida es igual a la de vuelta.

- Métrico: cuando además de ser simétrico se cumple la desigualdad triangular. Esto quiere decir que dik ≤ dij + djk. Expresado de otra manera, la distancia entre una ciudad i y una ciudad k, es menor que la distancia de i hasta k pasando por j.
- Euclídeo: cuando las ciudades son puntos del plano y la distancia entre ellas es la distancia euclídea dij = ‖ i − j ‖. Es el caso del problema que trata este TFG.

4.2.2. Aplicaciones del TSP

Aunque el Problema del Agente Viajero, debido a su nombre, parezca tener aplicaciones muy concretas o limitadas, es posible emplearlo para reducir tiempo, distancia y coste en cualquier proceso en el que se necesiten seleccionar una serie de puntos o nodos en orden.

El campo de la logística, en concreto en la planificación de rutas, es en el que más aplicaciones tiene, ya sea en una ordenación de flujos de personas, mercancías o vehículos a través de un conjunto de localizaciones o ciudades específicas que se necesitan visitar. Es utilizado por comerciales y guías turísticos con el fin de reducir la distancia recorrida, empresas dedicadas al transporte de personas y en el reparto de paquetería.

En la industria sus aplicaciones son menos numerosas, pero igualmente útiles. En la producción de placas de circuito impreso y microprocesadores personalizados, determina que ruta deberá seguir la taladradora y el láser respectivamente, para llegar a aquellos puntos que necesitan ser taladrados o cortados, con el fin de reducir la distancia y tiempo empleados.

Otras aplicaciones son la búsqueda de planetas, estableciendo la secuencia de observaciones con el telescopio, y la secuenciación del ADN.

4.2.3. Complejidad del TSP

La teoría de la complejidad computacional estudia cómo crece el coste computacional (principalmente memoria y tiempo) al resolver un problema en relación con lo que crece su tamaño.

El tiempo de resolución del problema dependerá de la complejidad del algoritmo, que es el número de operaciones básicas que ejecuta, y de la complejidad o tamaño del problema.

La teoría de la complejidad clasifica los problemas de la siguiente manera:

- **Problemas tipo P**: son aquellos a los que se puede encontrar solución en un tiempo polinomial, o computacionalmente razonable.
- **Problemas tipo NP**: son aquellos a los que no se les puede encontrar solución en tiempo polinomial, o computacionalmente razonable; aunque sí que permiten que una respuesta se pueda comprobar si es o no correcta en un tiempo razonable.
- **Problemas tipo NP-hard**: son aquellos problemas NP que cuentan con una dificultad superior, por eso también son denominados como "difíciles". Entre ellos se encuentra el Problema del Agente Viajero o TSP.

  Se ha demostrado que este tipo de problemas son computacionalmente equivalentes, por lo que, si un algoritmo polinómico resolviese uno de ellos, resolvería el resto de esta categoría. De poder resolver este tipo de problemas con un algoritmo eficiente, se concluiría que los problemas P y NP son iguales.

La complejidad del Problema del Agente Viajero lo sitúa entre los problemas NP-Hard. Esta dificultad en su resolución radica en que el número de soluciones posibles aumenta significativamente al aumentar el número de ciudades que se desean incluir en la ruta. Dada su complejidad factorial, de la que se hablará más adelante, no existen algoritmos exactos con tiempos de convergencia en tiempo polinómico; por lo que el tiempo de espera hasta obtener la solución óptima los hace inaplicables.

Si se utilizan algoritmos exactos o de búsqueda exhaustiva que den la solución óptima mediante la obtención y evaluación de todas y cada una de las posibles rutas de un problema con n ciudades (búsqueda de fuerza bruta), el número de rutas totales a evaluar es: $(n - 1)! / 2$.

Para el caso del problema que trata este TFG, en el que el número de centros de salud junto con el almacén es de 66 ($n = 66$), el número de rutas a construir y calcular podría llegar a ser 4.12×10^{90}.

$$(n - 1)! / 2 = (66 - 1)! / 2 = 4.12 \times 10^{90}$$

Asumiendo que un ordenador realiza una operación por nanosegundo (1.000.000.000 operaciones por segundo), a continuación, se muestra el tiempo de ejecución de este problema si se utilizan algoritmos exactos, dependiendo del número de centros de salud que se quieran abastecer.

TIEMPOS DE EJECUCIÓN SEGÚN EL NÚMERO DE CIUDADES (n)		
n	**Nº Operaciones**	**Tiempo**
10	181440	2 ms
15	4.4×10^{10}	44 seg
20	6×10^{16}	2 años
25	3.1×10^{23}	1×10^{7} años
30	4.4×10^{30}	1.4×10^{14} años
40	1×10^{46}	3.3×10^{29} años
50	3×10^{62}	1×10^{46} años
60	7×10^{79}	2.2×10^{63} años
66	4.1×10^{90}	1.3×10^{74} años

Tabla 2. Tiempos de ejecución de un algoritmo exacto.

Como se puede observar el tiempo de ejecución crece rápidamente al añadir un número pequeño de centros de salud. Esto es debido a su complejidad de tiempo factorial.

En el caso de este TFG en el que el número máximo de localizaciones a ordenar en la ruta es de 66, el tiempo de cálculo de la ruta óptima sería de 1.3×10^{74} años.

Por lo tanto, que el Problema del Agente Viajero sea tipo NP-Hard, supone que, para un relativamente alto número de ciudades o centros de salud, obtener la solución óptima mediante algoritmos exactos o de búsqueda exhaustiva no sea posible y se utilicen métodos heurísticos para obtener en un tiempo razonable una solución lo más próxima posible a la óptima.

4.3. Métodos de Resolución del TSP

4.3.1. Heurística para el TSP

La complejidad del cálculo del TSP supone el uso de la heurística como medio para guiar a un algoritmo en la obtención de soluciones muy próximas a la óptima, pero en un tiempo razonable.

El término "heurística" proviene de la palabra griega *heuriskein*, que puede traducirse como encontrar o descubrir.

Aplicado a los problemas de optimización de rutas, como es el caso del TSP, se utilizan algoritmos heurísticos para encontrar o descubrir una solución, de manera eficiente, muy próxima a la óptima y en un tiempo razonable, utilizando algún tipo de lógica, sin necesidad de buscar exhaustivamente todas y cada una de las soluciones posibles.

En definitiva, no se trata de encontrar la solución perfecta, sino una aproximada; ya que es mejor tener una buena respuesta que ninguna.

Existe gran diversidad de algoritmos heurísticos que se han desarrollado para tratar de simplificar y encontrar solución al TSP. Éstos son, por ejemplo, el algoritmo del vecino más cercano y del más lejano, algoritmos de inserción o los algoritmos genéticos.

4.3.1.1. Algoritmo del vecino más cercano

El algoritmo del vecino más cercano (nearest neighbor) quizás sea el más simple e intuitivo de los utilizados en optimización de rutas.

Este algoritmo trata de construir el ciclo hamiltoniano o ruta, de menor coste, partiendo de la ciudad de origen y eligiendo sucesivamente la ciudad más cercana a la última ciudad incluida en la ruta.

Es decir, una vez que conoce la ciudad de origen, añade al recorrido la ciudad vecina más cercana. Una vez que conoce esta primera ciudad a visitar, elige como siguiente ciudad la más cercana a ésta primera, repitiendo esta operación sucesivamente con las ciudades que aún no estén incorporadas en la ruta, hasta cerrar el recorrido volviendo a la ciudad de origen.

Este algoritmo comienza construyendo la ruta óptima con gran eficacia, uniendo la ciudad más cercana a la última ciudad incluida en la ruta sucesivamente, hasta que quedan todas visitadas. Sin embargo, como este algoritmo es voraz, y elige en cada paso local la mejor opción, al final del proceso es muy probable que queden ciudades cuya conexión suponen un elevado coste y que no se habían tenido en cuenta hasta ese momento, lo que supone una relativa ineficacia en el cálculo de la solución. Esto se conoce como miopía del procedimiento.

4.3.1.2. Algoritmo del vecino más lejano

El algoritmo del vecino más lejano (farthest neighbor), es uno de los más simples, al igual que el algoritmo del vecino más cercano. Sin embargo, su enfoque en la resolución del problema de optimización de rutas es completamente contrario.

Este algoritmo construye la ruta partiendo desde la ciudad de origen, y añadiendo a la ruta, en cada paso local, la ciudad más alejada a la última que se ha añadido.

Al tratarse al igual que el algoritmo del vecino más cercano de un algoritmo voraz, ya que en cada paso elige la ciudad más alejada sin tener una visión global del problema, tiene la misma desventaja en cuanto a su relativa ineficacia de cálculo.

4.3.1.3. Algoritmos de inserción

Los algoritmos de inserción difieren del algoritmo del vecino más cercano y del más lejano en que comienzan construyendo una ruta o ciclo aleatorio formado por unas cuantas ciudades del total de ciudades que se pretenden conectar, mientras que en el algoritmo del vecino más cercano y del más lejano comienzan con la ciudad de origen.

Una vez que los algoritmos de inserción han generado una ruta, completan el ciclo añadiendo a la ruta, en cada paso, una nueva ciudad entre dos ciudades consecutivas ya añadidas, siguiendo diferentes criterios.

Los criterios más utilizados son:

- Inserción aleatoria: se inserta una ciudad al azar.
- Inserción más lejana: se inserta la ciudad más alejada a todas las ciudades ya añadidas a la ruta.
- Inserción más cercana: se inserta la ciudad más cercana a todas las ciudades ya añadidas a la ruta.
- Inserción más barata: se inserta la ciudad que menos incrementa la distancia total de la ruta.

En la siguiente imagen se muestra un ejemplo de las ciudades que se deberían de añadir, en el primer paso de inserción, a la ruta inicial (A-B-C-D), según los diferentes criterios tratados anteriormente:

- Inserción aleatoria: añadir cualquiera de las ciudades al azar.
- Inserción más lejana: añadir la ciudad E.
- Inserción más cercana: añadir ciudad F.
- Inserción más barata: añadir la ciudad G.

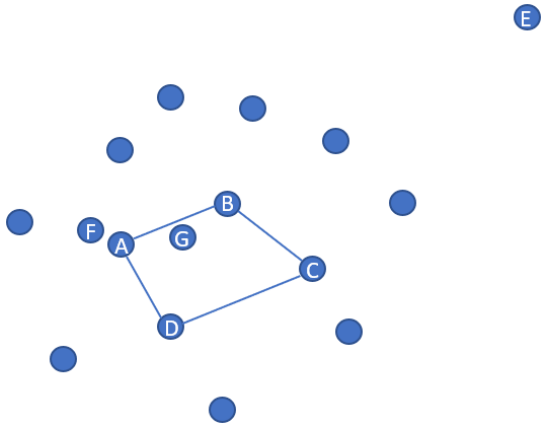

Ilustración 4. Ejemplo de algoritmo de inserción

4.3.1.4. Algoritmos genéticos

Los algoritmos genéticos se han decidido incluir en la optimización de rutas de este TFG, debido a los beneficios que aporta su interesante método de búsqueda de la solución óptima.

En este apartado se describen y explican los algoritmos genéticos de una manera general y desde su concepto original de resolución de problemas con soluciones de cadenas binarias. En el apartado "4.4.1.4. Paquete gor" se explica el algoritmo genético utilizado en este TFG y su rutina, ya que existen diferentes métodos de resolución, aunque tengan el mismo concepto y sentido las operaciones que realiza.

Un algoritmo genético es un tipo de algoritmo de búsqueda y optimización de soluciones, cuyo funcionamiento se basa en la teoría de la evolución de las especies y la selección natural de Charles Darwin.

En la evolución de cualquier especie, los individuos mejor adaptados al medio en el que viven tienen una mayor probabilidad de sobrevivir, reproducirse y obtener descendientes con características distintas, pero mejor adaptados y más capacitados para sobrevivir. En una búsqueda de soluciones mediante algoritmos genéticos, aquellas mejores soluciones, serán capaces de pasar de generación en generación, reproduciéndose y mutando, con la finalidad de encontrar una solución aún mejor. De la misma manera, al igual que los individuos peor adaptados suelen morir con una mayor probabilidad y

antelación debido a la selección natural; las peores soluciones obtenidas durante la búsqueda de la solución óptima mediante un algoritmo genético van a rechazarse y eliminarse en cada generación, tratando de quedarse con las mejores soluciones para encontrar la solución óptima.

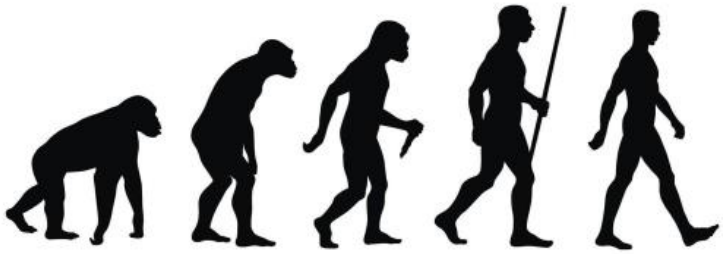

Ilustración 5. Analogía evolución de especies y soluciones.

(Fretes, 2018)

Un problema resuelto mediante algoritmos genéticos comienza con la creación de una población inicial de distintas soluciones aleatorias (primera generación). A continuación, se aplican una serie de operadores genéticos como el cruce, la mutación y la selección; con el fin de generar una nueva generación de mejores soluciones (segunda generación). Estas operaciones se realizan durante una serie de generaciones con el fin de que en cada generación las soluciones existentes sean mejores o tengan mayor capacidad para resolver el problema (en el caso de la optimización de rutas, disminuir el recorrido al máximo).

Originalmente los algoritmos genéticos se aplicaron para problemas cuyas soluciones se expresan mediante una cadena binaria formada por una serie de bits. En este tipo de problemas, las soluciones dependen del valor de cada uno de los bits que la forman, de la misma manera que las características de cada ser vivo dependen de sus genes.

1	0	0	1	0	1

Ilustración 6. Cadena binaria

Todos los seres vivos están formados por células, y éstas a su vez por cromosomas, que contienen el ADN; el cual a su vez está formado por unos segmentos llamados genes, que son las unidades que se

heredan y pasan de padres a hijos, y que determinan las características de cada individuo.

De manera análoga, en los problemas cuyas soluciones se representan mediante cadenas binarias, las posibles soluciones representan los cromosomas, y cada uno de los bits que la forman los distintos genes que determinan las características finales de una solución e individuo respectivamente.

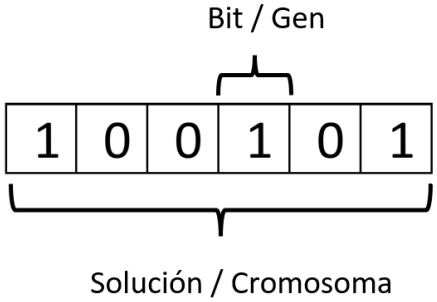

Ilustración 7. Analogía solución/cromosoma y bit/gen

4.3.1.4.1. Pasos de un algoritmo genético

Un algoritmo genético comienza generando una población inicial de soluciones del tamaño que se le indique.

Posteriormente se seleccionan parejas de soluciones y se les aplica el operador genético de cruce, obteniendo descendencia.

A estos descendientes se les aplica el siguiente operador genético, la mutación.

Finalmente se realiza una selección de las soluciones mejor adaptadas, las cuales pasan a la siguiente generación. Este proceso se realiza tantas veces como generaciones se le indique al algoritmo que realice.

Obviamente, cuanta mayor sea la población de soluciones de cada generación y el número de generaciones que se deseen calcular, el algoritmo tardará más en resolver el problema, pero mejores soluciones se obtendrán.

A continuación, se presenta mediante un diagrama de flujo la sucesión de pasos que sigue una búsqueda con algoritmos genéticos:

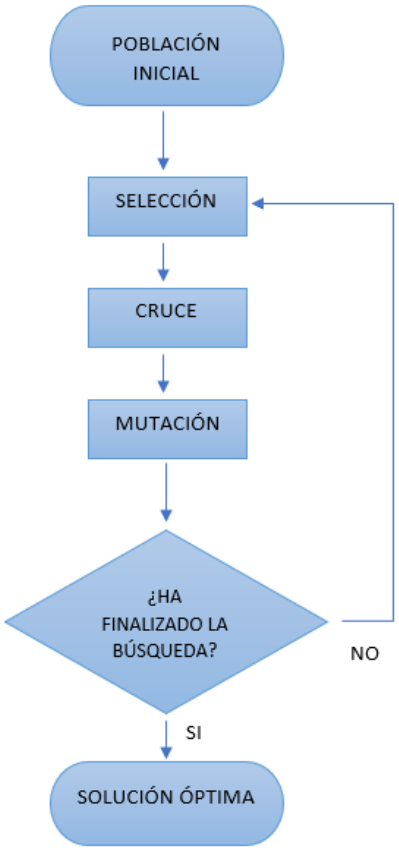

Ilustración 8. Diagrama de flujo de un algoritmo genético.

Una vez se han introducido los algoritmos genéticos, los pasos que siguen en la resolución de un problema y búsqueda de su solución; y se han visualizado en el diagrama de flujo anterior, se profundiza en cada uno de estos pasos con las siguientes explicaciones:

- Población inicial: el primer paso que realiza un algoritmo genético es generar una población o conjunto inicial de soluciones, ya que es el requisito para que pueda comenzar a buscar soluciones óptimas al problema en las sucesivas generaciones.

  Un número pequeño de soluciones iniciales aumenta el riesgo de converger a un óptimo local y no obtener el óptimo global que se busca, debido a la poca variedad de soluciones y descendencia, en cada generación. Sin embargo, una población inicial elevada, hace que el trabajo informático crezca, aumentando la complejidad y tiempo de resolución; por lo que hay que encontrar un equilibrio entre complejidad temporal y probabilidad de encontrar una solución de alta calidad.

- Selección: la operación de selección se utiliza tanto en la selección de "soluciones padres" para reproducirse en el operador de cruce en cada generación, como en la elección de las soluciones que van a formar la siguiente generación en cada una de las iteraciones o generaciones que se propongan calcular.

  En general, lo que pretende la selección en los algoritmos genéticos, al igual que la selección natural en las especies, es permitir que las soluciones o individuos respectivamente más aptos se reproduzcan para aprovechar sus buenas características en la siguiente generación.

- Cruce: es el operador genético que simula la reproducción, y combina la información de dos "soluciones padres" para crear descendencia y obtener "soluciones hijas".

  El objetivo del cruce es crear descendencia con las mejores características de sus padres, lo que permite mejorar la calidad de la población en cada generación.

  Un ejemplo de cruce es seleccionar un punto de corte entre los cromosomas de los padres o "soluciones padres", y combinar las secciones resultantes. Así las "soluciones hijas" tendrán información de sus dos progenitores.

Realizando el corte a partir del tercer bit en las "soluciones padres", se obtienen dos "soluciones hijas", que poseen la información y características de sus progenitores:

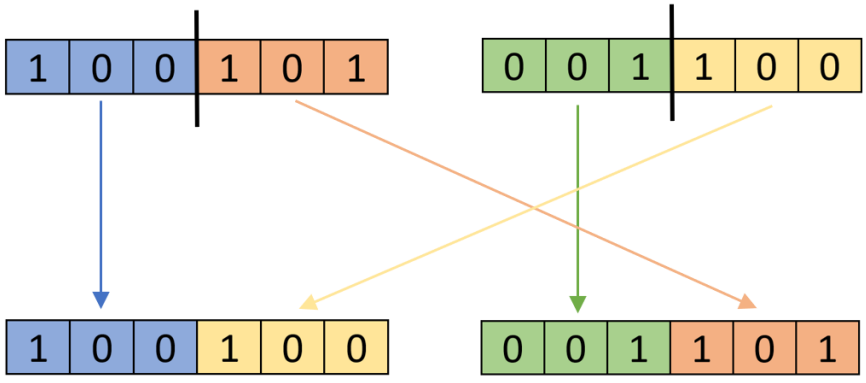

Ilustración 9. Ejemplo de cruce en cadenas binarias

- Mutación: este operador genético implica modificar de forma aleatoria los genes o características de las soluciones que se hayan obtenido hasta el momento, sin que este cambio guarde relación con las características de sus respectivos padres como en el cruce.

  Esta operación permite introducir diversidad a las soluciones y por tanto explorar nuevas regiones del espacio de búsqueda, con la finalidad de evitar la convergencia a óptimos locales, con respecto a la operación de cruce.

  Un ejemplo de mutación es cambiar el estado de un bit en una solución de cadena binaria:

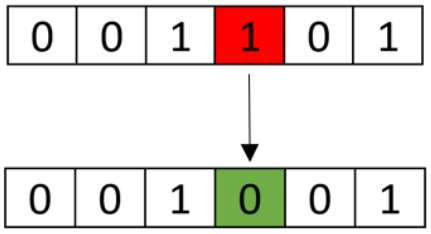

Ilustración 10. Ejemplo de mutación en cadenas binarias

4.3.1.4.2. Algoritmos genéticos para el TSP

El Problema del Agente Viajero o TSP es uno de los problemas que pueden ser resueltos, al menos de una forma aproximada debido a su complejidad NP-Hard, mediante la utilización de los algoritmos genéticos.

Cabe destacar que la representación de los recorridos en el TSP no se hace mediante cadenas binarias por conveniencia, ya que no es una representación natural del problema y las soluciones no son inmediatamente traducibles a bits. Las cadenas binarias se utilizan cuando las soluciones representan una combinación de valores booleanos (verdadero o falso), como en el problema de la mochila, en la que cada bit de la cadena binaria representa un objeto que se decide introducir (representado con el valor de 1) o que se decide no introducir (representado con el valor de 0).

En el TSP lo más razonable es representar las soluciones en forma de lista, que recoja las ciudades ordenadas según deben visitarse para minimizar el recorrido. Estas soluciones permiten de una manera más sencilla la aplicación de los operadores genéticos y evitan que se produzcan soluciones inválidas como, por ejemplo, rutas que no visitan todas las ciudades, o rutas que visitan varias veces las mismas.

A pesar de esta diferencia, se siguen aplicando los mismos operadores genéticos, con el mismo sentido y finalidad, pero utilizando otros métodos a los anteriormente expuestos. En el apartado "4.4.1.4 Paquete gor" de este TFG, se explican los operadores genéticos utilizados en la optimización de la ruta de reparto de material sanitario en Zaragoza.

4.4. HERRAMIENTAS

A continuación, se presentan las herramientas más implicadas en la generación de rutas de reparto y creación de la aplicación.

4.4.1. R-Project

Para la realización de este TFG se ha utilizado R, un lenguaje y entorno de software libre diseñado para la manipulación de datos y su análisis estadístico con visualización gráfica.

Se trata de un proyecto GNU (General Public License), es decir, los usuarios tienen libertad para utilizar, compartir, modificar y mejorar el software. Es por ello que está en constante evolución gracias a la publicación libre y gratuita de paquetes de software. Estos paquetes de R están disponibles en la CRAN (Comprehensive R Archive Network). Esto permite que científicos de todo el mundo puedan compartir código de una manera sencilla para contribuir con sus investigaciones.

R ha ganado gran popularidad, consiguiendo un incremento de su uso a nivel profesional y educativo, debido a que tiene una curva de aprendizaje sencilla en comparación con otros lenguajes de programación y permite la realización de múltiples tareas de manipulación de datos.

4.4.1.1. RStudio

R-Studio es el entorno de desarrollo integrado (IDE) para el lenguaje de programación R, que se ha utilizado, junto con el paquete Shiny, para la creación de la aplicación de este TFG.

Se trata de un software libre que pretende utilizar y visualizar R de manera más cómoda mediante una interfaz intuitiva, la cual permite su entendimiento y uso a expertos en la materia y usuarios nuevos.

Como se muestra en la imagen inferior, consta de un editor de texto en el que se escribe el código y las órdenes que se desean ejecutar (panel superior izquierdo), una consola donde se ejecutan las órdenes presentes en el editor de texto (panel inferior izquierdo), un panel en el que aparecen todos los objetos guardados en la memoria (panel superior derecho) y, por último, un panel que muestra los paquetes instalados, una pestaña de ayuda, el historial de archivos trabajados con el programa y la visualización de gráficos generados.

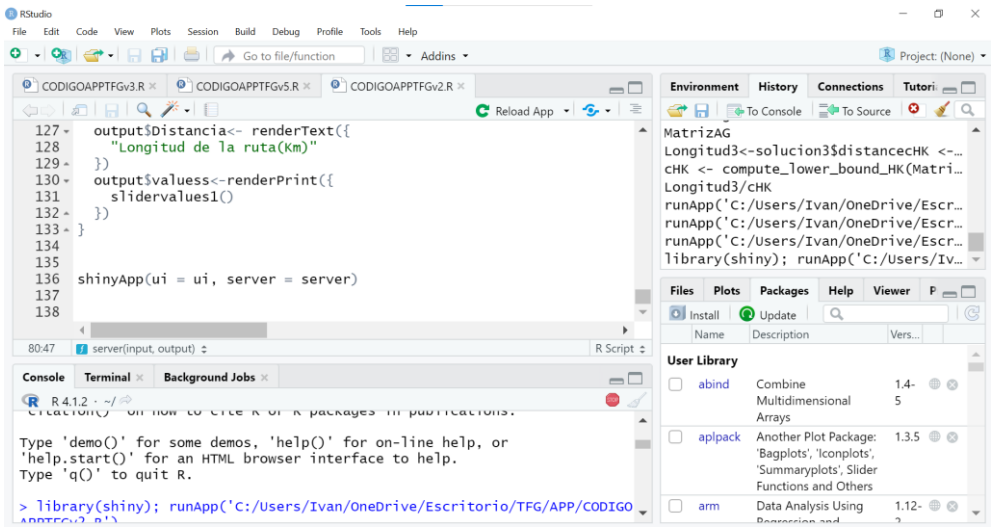

Ilustración 11. Interfaz RStudio

4.4.1.2. Paquete shiny

El paquete de R utilizado para la creación de la aplicación en este TFG recibe el nombre de Shiny.

Este paquete facilita la creación de aplicaciones interactivas, permitiendo al usuario interactuar con los datos sin necesidad de manipular el código desde el editor de texto de RStudio. En el caso de la aplicación creada para este TFG, el usuario únicamente deberá seleccionar aquellos centros de salud que necesite visitar y hacer click en el botón de "Calcular ruta" para obtener la ruta óptima, junto con su distancia y un mapa interactivo.

El script de la aplicación está formado por tres componentes:

- El objeto de la interfaz de usuario (ui): permite establecer el diseño y la apariencia de la aplicación. En él se definen las variables de entrada que elegirá el usuario y los elementos de salida que la aplicación mostrará.
- La función del servidor (server): permite generar los cálculos que supondrán el funcionamiento de la aplicación. En él se definen las funciones y objetos que darán la solución.
- La función de llamada a la aplicación (shinyApp): permite visualizar la aplicación Shiny a partir del ui y el server.

Cabe destacar que actualmente las aplicaciones pueden estar contenidas en un único archivo, como en el caso de la aplicación creada en este TFG. Sin embargo, antes de la versión 0.10.2, Shiny no lo admitía y el objeto de la interfaz de usuario y la función del servidor debían estar contenidos en dos scripts diferentes (ui.R y server.R).

4.4.1.3. Paquete TSP

El paquete TSP de R, tal y como su nombre indica, proporciona un conjunto de métodos heurísticos y exactos para resolver el Problema del Agente Viajero o TSP.

En el caso de este TFG se ha utilizado este paquete para utilizar 2 métodos heurísticos con los que obtener la solución al problema de optimización de rutas. Estos dos métodos son:

- Algoritmo del vecino más cercano (nearest neighbour)
- Algoritmo del vecino más lejano (farthest neighbour)

La función para resolver el problema en R es como sigue: solve_TSP().

A esta función hay que indicarle el método que se quiere utilizar:

- Algoritmo del vecino más cercano: method="nearest_insertion"
- Algoritmo del vecino más lejano: method="farthest_insertion"

El funcionamiento y metodología de búsqueda de la solución óptima han sido explicados en los puntos "4.3.1.1 Algoritmo del vecino más cercano" y "4.3.1.2 Algoritmo del vecino más lejano", respectivamente.

4.4.1.4. Paquete gor

Tal y como se ha mencionado anteriormente en varias ocasiones, el paquete de R utilizado para realizar la búsqueda de la ruta óptima de este TFG, mediante un algoritmo genético, es el paquete denominado "gor".

Dependiendo del creador de cada algoritmo genético, los operadores genéticos y las características de la búsqueda de la solución pueden ser distintas, aunque siempre tengan el mismo sentido y finalidad.

A continuación, se detallan y explican los pasos, métodos y características del algoritmo genético que ha sido utilizado en la búsqueda de la solución óptima de la ruta de reparto de material sanitario en Zaragoza:

Selección de la población inicial y número de generaciones:

En primera instancia se definen los valores del tamaño de la población inicial y el número de generaciones que se desean construir en la búsqueda de la ruta.

El número de individuos de la población inicial de cada generación se elige utilizando el parámetro "Npop". Los individuos o soluciones de la primera generación se crean aleatoriamente.

El número de generaciones o iteraciones se elige con el parámetro "Ngen".

Cruce (Algoritmo de Christofides):

Una vez se tiene la población inicial de una generación se procede a realizar el cruce de dos "soluciones padres" o ciclos, con la finalidad de obtener dos "soluciones hijas" con características de sus progenitores, utilizando el algoritmo de Christofides.

Mientras que, si se dispone de una ruta, todos las ciudades o vértices del grafo tienen grado par (entra una arista y sale otra), si se superponen en el cruce dos rutas o "soluciones padres", debido a que éstas a priori no son iguales, quedarán vértices o ciudades con grado impar.

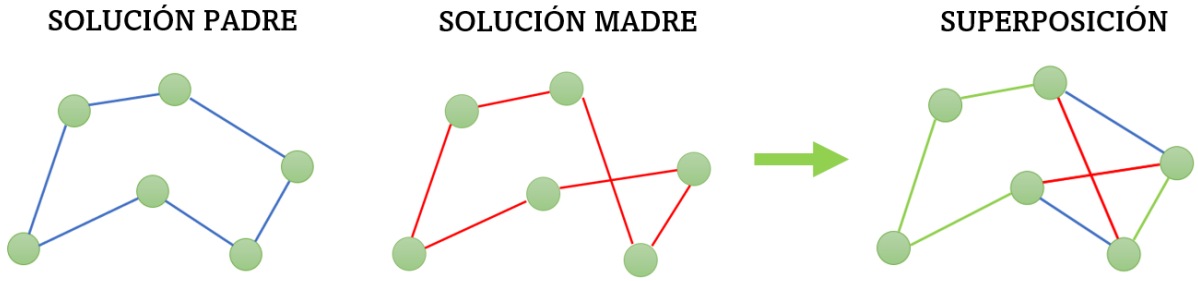

Ilustración 12. Superposición de dos soluciones

A continuación, se explica cómo el algoritmo de Christofides genera las dos "soluciones hijas":

- Para calcular la primera de las dos "soluciones hijas", se genera un árbol recubridor de distancia mínima que une todas las ciudades de ese grafo obtenido de la superposición de las dos rutas o "soluciones padre".
- Posteriormente se unen mediante aristas aquellos vértices o ciudades que han quedado con grado impar de manera que tengan distancia mínima, y se busca aquel recorrido de distancia mínima que una todas las ciudades (esa es la primera solución).
- Para buscar la segunda solución hija, se eliminan las aristas de la solución anterior y se realiza la misma búsqueda.

La forma en que se crean pares de parejas en el cruce, es mediante una distribución de probabilidad exponencial en la distancia total de las rutas, la cual se explica en el siguiente apartado de "Selección de las parejas para el cruce".

Se realizan cruces hasta obtener tantas "soluciones hijas" como "soluciones padres" tenía la población antes del cruce.

Si las soluciones hijas obtenidas son repetidas a alguna ya existente o son muy malas, se descartan. En caso contrario, se añaden a una "reserva" hasta obtener tantas soluciones hijas como soluciones padres hay.

Selección de las parejas para el cruce:

La selección de las parejas de soluciones padres que se cruzan se realiza con una cierta probabilidad exponencial en la distancia total recorrida por cada solución.

De esta manera, cuanto mayor es la distancia total recorrida, la exponencial es más pequeña y estas soluciones son elegidas con menor probabilidad. De la misma manera, cuanto menor es la distancia total recorrida, la exponencial es más grande y estas soluciones son elegidas con mayor probabilidad.

El parámetro "beta", el cual se puede modificar, es el que favorece la selección de las mejores soluciones.

La razón por la que se utiliza la distribución exponencial es porque si siempre se eligieran únicamente las mejores soluciones, es probable que saliesen los mismos padres, obteniendo también los mismos hijos; y por lo tanto no se mejorarían las soluciones, perdiendo el sentido del algoritmo genético.

Esta inclusión de soluciones "peores" según la distribución exponencial permite la variedad de nuevas soluciones dando lugar a la evolución de las mismas, consiguiendo mejores soluciones. Realmente es así como se consigue dispersar la búsqueda de soluciones por todo el espacio de búsqueda.

Mutación (4-intercambio):

Una vez que, a la población inicial se le han sumado las soluciones hijas obtenidas en el cruce, se aplica el 4-intercambio.

Se trata de un método de perturbación que consiste en eliminar 4 aristas no coincidentes de un ciclo, de forma aleatoria, y formar otro realizando una unión, distinta a la inicial, de los vértices que han quedado con grado 1 (sin unir o cerrar el ciclo).

Esta perturbación permite alejarse de los mínimos locales que se han obtenido y muestrear mejor el espacio de búsqueda, permitiendo encontrar nuevos óptimos locales, que se puedan acercar más al mínimo global en las consecutivas iteraciones o generaciones.

Esta operación de mutación se realiza sobre todas las soluciones presentes en cada generación.

A continuación, se presenta un 4-intercambio realizado sobre un grafo de 16 nodos, o ciudades:

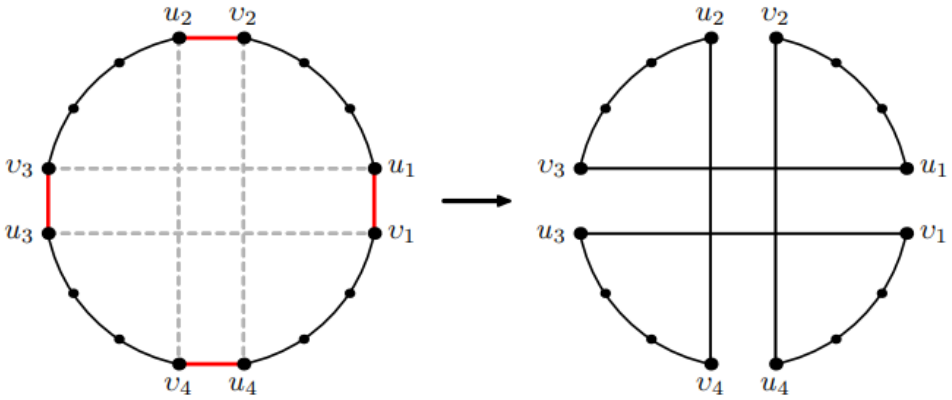

Ilustración 13. Ejemplo de 4-intercambio

(Asensio, 2022)

Búsqueda local (2-opt):

Esta operación no está presente en el proceso común de búsqueda de soluciones de un algoritmo genético, sin embargo, en el algoritmo genético utilizado en este TFG se aplica con una intención muy concreta y beneficiosa que se explica a continuación.

Una vez la población está formada por la población inicial, las soluciones hijas obtenidas en el cruce y las soluciones obtenidas en el 4-intercambio, se aplica una búsqueda local.

La búsqueda local es una técnica que consiste en mejorar soluciones mediante algún método heurístico introduciendo en ellas pequeñas modificaciones.

Cuando se trata de esforzarse en obtener las mejores soluciones posibles, y una solución está cerca de un mínimo, el algoritmo 2-opt permite bajar aún más esa solución y obtener en este caso una ruta más corta.

Para comprender el algoritmo denominado 2-opt, conviene explicar el 2-intercambio. El sentido es el mismo que el 4-intercambio, pero únicamente con 2 artistas.

Por lo tanto, el 2-intercambio es un método heurístico que consiste en eliminar dos aristas no incidentes de una solución o ciclo, y construir otro ciclo introduciendo dos aristas que no estaban presentes en la solución inicial, tal y como se muestra en la siguiente imagen:

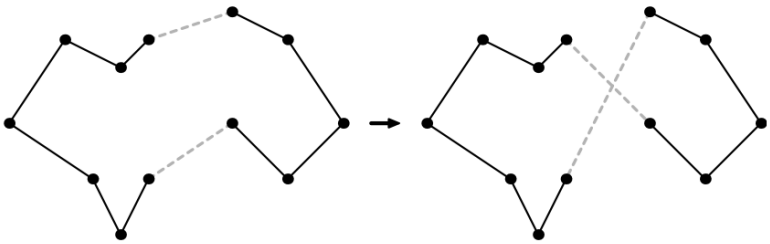

Ilustración 14. Ejemplo de 2-intercambio

(Asensio, 2022)

El 2-intercambio es la base para realizar el 2-opt.

El algoritmo 2-opt comienza con una ruta o solución inicial, y construye todos los posibles ciclos obtenidos a través de un 2-intercambio. Existen un total de n(n-3)/2 ciclos, siendo *n* el número de ciudades de la ruta.

Si existe un ciclo obtenido de ese 2-intercambio con una menor distancia que el original, este ciclo obtenido se toma como nuevo ciclo de partida.

Este proceso se repite hasta que ningún ciclo obtenido tras aplicar todos los posibles 2-intercambio sobre el ciclo que en ese momento sea el de partida sea todavía mejor.

Este algoritmo se aplica sobre todas las soluciones presentes en la generación, aunque existe un parámetro denominado "local" para seleccionar aleatoriamente y con una determinada probabilidad las soluciones que iniciarán una búsqueda local. Este parámetro permite disminuir el tiempo de ejecución del algoritmo genético, ya que realizar una búsqueda local sobre grandes poblaciones o muchas generaciones, es computacionalmente muy intensivo.

Selección de soluciones para formar la población inicial de la siguiente generación:

Cabe destacar que en la selección de las soluciones que pasan a la siguiente generación, además de utilizar la probabilidad exponencial en la distancia de la ruta, también se utiliza una técnica denominada "elitismo".

Esta técnica implica la selección de las mejores soluciones de una determinada población y asegurar que se transmiten a la siguiente. Su finalidad es asegurarse de que las mejores soluciones no se pierdan en

las distintas generaciones, aumentando la calidad de cada generación sucesivamente.

Inicialmente el algoritmo genético utilizado elige de entre todas las soluciones obtenidas en una generación (población inicial + soluciones obtenidas en el cruce + soluciones obtenidas en la mutación + soluciones obtenidas en la búsqueda local), las dos mejores soluciones y las transmite a la siguiente generación. Este número de soluciones que pasan directamente a la siguiente generación se controla con el parámetro "elite".

El resto de las soluciones que pasan a la siguiente solución, hasta completar el número de individuos de la población inicial de cada generación, se eligen según la distribución exponencial explicada anteriormente.

De esta manera, la siguiente generación siempre se queda con la "crema" de la anterior, y permite la posibilidad, según la distribución exponencial, de que pasen otras soluciones peores, que permitan explorar otra región del espacio de búsqueda.

4.4.1.4.1. Resumen del algoritmo genético

A modo de resumen, la forma de operar que tiene el algoritmo genético utilizado en este Trabajo Final de Grado es la siguiente:

- Conviene seleccionar el tamaño de la población inicial de soluciones y el número de generaciones que se generarán en búsqueda de la solución.
  Esto se realiza con los parámetros "Npop" y "Ngen", respectivamente.

- Mediante una distribución exponencial en la distancia de las rutas de las soluciones, se procede al cruce entre parejas de soluciones padres, y mediante el algoritmo de Christofides se obtienen dos soluciones hijas de cada pareja de progenitores.

- Las soluciones obtenidas hasta el momento se someten a la operación de mutación mediante un 4-intercambio, tratando de alejarse de los mínimos locales que se han obtenido y muestrear mejor el espacio de búsqueda, permitiendo encontrar nuevos óptimos locales, que se puedan acercar más al mínimo global en las consecutivas iteraciones o generaciones.

- Las soluciones obtenidas hasta el momento se someten a una búsqueda local mediante el algoritmo 2-opt, que trata de bajar aún más aquellas soluciones que están cerca de un mínimo.

- Se seleccionan mediante la técnica denominada elitismo algunas de las mejores soluciones obtenidas hasta ese momento en la generación, y mediante la distribución exponencial en la distancia total de la ruta, el resto de las soluciones hasta completar la población inicial de la siguiente generación.

Este proceso se repite tantas veces como generaciones se hayan especificado inicialmente, hasta que se termina la búsqueda y se obtiene la mejor solución encontrada.

4.5. GENERACIÓN DE LA APLICACIÓN

A continuación, se detalla y explica el código desarrollado para crear la aplicación en RStudio.

4.5.1. Librerías

Inicialmente, se deben cargar las librerías de R que van a ser utilizadas en la creación de la aplicación, así como en la obtención de las rutas y mapas:

```
library(shiny)

library(TSP)

library(gdata)

library(readxl)

library(gor)

library(leaflet)

library(openrouteservice)
```

Ilustración 15. Librerías Aplicación

Estas librerías tienen distintas funciones en el desarrollo de la aplicación:

- Librería shiny: permite crear aplicaciones web interactivas en R.
- Librería TSP: permite resolver el TSP mediante los algoritmos del vecino más lejano y el más cercano.
- Librería gdata: lectura y escritura de datos de hojas de cálculo como Excel.
- Librería readxl: está enfocado en la lectura de archivos de Excel en formatos diferentes (.xls y .xlsx)
- Librería gor: permite resolver el TSP mediante algoritmos genéticos.
- Librería leaflet: utilizado en la creación de mapas interactivos.
- Librería openrouteervice: permite tener acceso a la API de OpenRouteService y disponer de datos geográficos y de enrutamiento en tiempo real.

4.5.2. Acondicionamiento de objetos para aplicación de librerías

Se introducen ciertas modificaciones en los archivos de Excel generados con las coordenadas de los centros de salud y la matriz de distancias; y se crean nuevos objetos, para posteriormente utilizarlos con mayor facilidad:

```
MatrizDistancias <- read_excel("MatrizDistanciasnueva.xlsx")
Distancias<-as.matrix(MatrizDistancias)
rownames(Distancias)<-colnames(Distancias)

Coordenadas<-read_excel("CoordenadasIvan.xlsx",col_names = TRUE,sheet=3)
names(Coordenadas)<-c("LATITUD","LONGITUD")
Coordenadas<-as.matrix(Coordenadas)
coordenadas<-Coordenadas

ID1=rownames(coordenadas)<-rownames(Distancias)
```

Ilustración 16. Datos iniciales

Se lee el archivo de Excel que contiene la matriz de distancias ("MatrizDistanciasnueva.xlsx") y se guarda como un objeto con el nombre de "MatrizDistancias", que se convierte en una matriz con el nombre de "Distancias", a cuyas filas se les añade el mismo nombre que tienen las columnas (son los nombres de los centros de salud).

Se crea un objeto con el nombre de "Coordenadas" con las coordenadas presentes en el archivo de Excel "CoordenadasIvan.xlsx". Se renombran las columnas de latitud y longitud para que sean identificadas como tal; y convierte el objeto en una matriz ("Coordenadas"). Finalmente crea un nuevo objeto ("coordenadas") al que le asigna el objeto "Coordenadas".

La última línea crea la variable "ID1" a la que le asigna los nombres de las filas de la matriz "Distancias".

4.5.3. Objeto de la interfaz de usuario "ui"

Esta parte del código permite generar el diseño y la apariencia de la aplicación. En él también se definen las variables de entrada que elegirá el usuario y los elementos de salida que la aplicación mostrará.

```
ui<-fluidPage(
  titlePanel("RUTA DE REPARTO DE MATERIAL SANITARIO EN ZARAGOZA"),
  sidebarPanel(
    tabsetPanel(
```

Ilustración 17. Objeto interfaz de usuario

La función "fluidPage" permite la creación de páginas web que se adaptan a las dimensiones del navegador.

Se introduce el título de la aplicación con el "titlePanel", y un panel lateral (sidebarPanel), que contiene una serie de pestañas (tabsetPanel).

Estas pestañas (tabPanel) corresponden a cada uno de los sectores de salud de Aragón.

Contienen un título que las identifica y un "checkboxGroupInput". Esta función permite crear las casillas de cada uno de los centros de salud dentro de las pestañas, para su posterior selección como variables de entrada y reparto.

El parámetro "choices" permite la introducción de estas casillas en cada una de las pestañas, y el "selected" muestra las casillas o centros de salud preseleccionados en la aplicación una vez se abra.

```
      tabPanel(title="Sector Zaragoza I",
               checkboxGroupInput("variable1","Seleccione los centros de
salud que desea abastecer",

choices=c("C.S. ACTUR NORTE"="C.S. ACTUR NORTE","C.S. AMPARO POCH (ACTUR
OESTE)"="C.S. AMPARO POCH (ACTUR OESTE)", "C.S. ACTUR SUR"="C.S. ACTUR
SUR","C.S. ALFAJARÍN"="C.S. ALFAJARÍN","C.S. ARRABAL"="C.S.
ARRABAL","C.S. LA JOTA"="C.S. LA JOTA","C.S. BUJARALOZ"="C.S.
BUJARALOZ","C.S. LUNA"="C.S. LUNA","C.S. PARQUE GOYA"="C.S. PARQUE
GOYA","C.S. SANTA ISABEL"="C.S. SANTA ISABEL","C.S. VILLAMAYOR"="C.S.
VILLAMAYOR","C.S. PICARRAL"="C.S. PICARRAL","C.S. ZUERA"="C.S. ZUERA"),


selected=c("C.S. ACTUR NORTE"="C.S. ACTUR NORTE","C.S. LUNA"="C.S. LUNA",
"C.S. PARQUE GOYA"="C.S. PARQUE GOYA","C.S. SANTA ISABEL"="C.S. SANTA
ISABEL","C.S. VILLAMAYOR"="C.S. VILLAMAYOR","C.S. PICARRAL"="C.S.
PICARRAL"),inline=TRUE)),
```

Ilustración 18. Pestaña Sector Zaragoza I

```
      tabPanel(title="Sector Zaragoza II",
               checkboxGroupInput("variable2","Seleccione los centros de
salud que desea abastecer",

choices=c("C.S. ALMOZARA"="C.S. ALMOZARA","C.S. CAMPO DE BELCHITE"="C.S.
CAMPO DE BELCHITE","C.S. CASABLANCA"="C.S. CASABLANCA","C.S. FERNANDO EL
CATÓLICO"="C.S. FERNANDO EL CATÓLICO","C.S. FUENTES DE EBRO"="C.S.
FUENTES DE EBRO","C.S. PARQUE ROMA"="C.S. PARQUE ROMA", "C.S. PUERTA DEL
CARMEN"="C.S. PUERTA DEL CARMEN","C.S. LAS FUENTES NORTE"="C.S. LAS
FUENTES NORTE","C.S. JOSÉ R. MUÑOZ FERNÁNDEZ (SAGASTA)"="C.S. JOSÉ R.
MUÑOZ FERNÁNDEZ (SAGASTA)","C.S. REBOLERÍA"="C.S. REBOLERÍA","C.S.
ROMAREDA (SEMINARIO)"="C.S. ROMAREDA (SEMINARIO)","C.S. SAN JOSÉ"="C.S.
SAN JOSÉ","C.S. CANAL IMPERIAL"="C.S. CANAL IMPERIAL","C.S. SAN
PABLO"="C.S. SAN PABLO","C.S. SÁSTAGO"="C.S. SÁSTAGO","C.S. TORRE
RAMONA"="C.S. TORRE RAMONA","C.S. TORRERO-LA PAZ"="C.S. TORRERO-LA
PAZ","C.S. VALDESPARTERA"="C.S. VALDESPARTERA"),

selected=c("C.S. ALMOZARA"="C.S. ALMOZARA"),inline=TRUE)),
```

Ilustración 19. Pestaña Sector Zaragoza II

```
      tabPanel(title="Sector Zaragoza III",
               checkboxGroupInput("variable3","Seleccione los centros de
salud que desea abastecer",

choices=c("C.S. ALAGÓN"="C.S. ALAGÓN","C.S. BOMBARDA"="C.S.
BOMBARDA","C.S. BORJA"="C.S. BORJA","C.S. CARIÑENA"="C.S. CARIÑENA",
C.S. CASETAS"="C.S. CASETAS","C.S. DELICIAS NORTE"="C.S. DELICIAS NORTE",
"C.S. DELICIAS SUR"="C.S. DELICIAS SUR","C.S. VIRGEN DE LA OLIVA-EJEA DE
LOS CABALLEROS"="C.S. VIRGEN DE LA OLIVA-EJEA DE LOS CABALLEROS","C.S.
ÉPILA"="C.S. ÉPILA","C.S. GALLUR"="C.S. GALLUR","C.S. HERRERA DE LOS
NAVARROS"="C.S. HERRERA DE LOS NAVARROS","C.S. LA ALMUNIA DE DOÑA
GODINA"="C.S. LA ALMUNIA DE DOÑA GODINA","C.S. MARÍA DE HUERVA"="C.S.
MARÍA DE HUERVA","C.S. MIRALBUENO-GARRAPINILLOS"="C.S. MIRALBUENO-
GARRAPINILLOS","C.S. OLIVER"="C.S. OLIVER","C.S. SÁDABA"="C.S. SÁDABA",
"C.S. SOS DEL REY CATÓLICO"="C.S. SOS DEL REY CATÓLICO","C.S. SAN
ATILANO- TARAZONA"="C.S. SAN ATILANO- TARAZONA","C.S. TAUSTE"="C.S.
TAUSTE","C.S. UNIVERSITAS"="C.S. UNIVERSITAS", "C.S. UTEBO"="C.S. UTEBO",
"C.S. VALDEFIERRO"="C.S. VALDEFIERRO"),

selected=c("C.S. ALAGÓN"="C.S. ALAGÓN"),inline=TRUE)),
```

Ilustración 20. Pestaña Sector Zaragoza III

```
      tabPanel(title="Sector Alcañiz",
               checkboxGroupInput("variable4","Seleccione los centros de
salud que desea abastecer",

choices=c("C.S. AMANDO LORIGA-CASPE"="C.S. AMANDO LORIGA-CASPE","C.S.
MAELLA"="C.S. MAELLA"),

selected=c("C.S. AMANDO LORIGA-CASPE"="C.S. AMANDO LORIGA-
CASPE"),inline=TRUE)),
```

Ilustración 21. Pestaña Sector Alcañiz

```
      tabPanel(title="Sector Barbastro",
               checkboxGroupInput("variable5","Seleccione los centros de
salud que desea abastecer",

choices=c("C.S. MEQUINENZA"="C.S. MEQUINENZA"),

selected=c("C.S. MEQUINENZA"="C.S. MEQUINENZA"),inline=TRUE)),
```

Ilustración 22. Pestaña Sector Barbastro

```
  tabPanel(title="Sector Calatayud",
               checkboxGroupInput("variable6","Seleccione los centros de
salud que desea abastecer",

choices=c("C.S. ALHAMA DE ARAGÓN"="C.S. ALHAMA DE ARAGÓN",C.S.
ARIZA"="C.S. ARIZA","C.S. ATECA"="C.S. ATECA","C.S. CALATAYUD"="C.S.
CALATAYUD","C.S. DAROCA"="C.S. DAROCA","C.S. ILLUECA"="C.S. ILLUECA",
"C.S. MORATA DE JALÓN"="C.S. MORATA DE JALÓN","C.S. SABIÑAN"="C.S.
SABIÑAN","C.S. VILLARROYA DE LA SIERRA"="C.S. VILLARROYA DE LA SIERRA"),

selected=c("C.S. ALHAMA DE ARAGÓN"="C.S. ALHAMA DE
ARAGÓN"),inline=TRUE)),
```

Ilustración 23. Pestaña Sector Calatayud

Además de las pestañas que permiten seleccionar como "input" las variables o centros de salud que se desea abastecer, para cada uno de los sectores de salud de Aragón; se crea una última pestaña que permite modificar los parámetros del algoritmo genético, y que guían su proceso de búsqueda de la solución. La implicación de cada parámetro ha sido explicada en el punto "4.4.1.4 Paquete gor".

Estos parámetros son: "Npop", "Ngen", "Local" y "Elite".

La función "numericInput" permite crear un campo de entrada numérico para cada parámetro del algoritmo genético. Estos valores de entrada serán utilizados posteriormente en la búsqueda de la solución.

El valor "value" sirve para indicar el valor predeterminado del parámetro una vez se abra la aplicación. Los valores "min" y "max" sirven para establecer el límite de los valores posibles de cada parámetro. Finalmente, el valor "step" se refiere al tamaño del paso incremental de cada parámetro, cuando se decidan aumentar o disminuir sus valores.

```
      tabPanel(title="Parámetros algoritmo genético",


        numericInput("Npop","Seleccione la población inicial",value=10,
min=2,max=100, step=1),
        numericInput("Ngen","Seleccione el número de
generaciones",value=1, min=1,max=100, step=1),
        numericInput("Local","Seleccione el parámetro Local",value=1,
min=0,max=1, step=0.05),
        numericInput("Elite","Seleccione el parámetro Elite",value=2,
min=1,max=6, step=1))),
```

Ilustración 24. Pestaña Parámetros Algoritmo genético

Se establece la anchura del panel lateral creado con todas las pestañas anteriormente expuestas, mediante la función "width".

```
    width=45),
```

Ilustración 25. Anchura del panel lateral

Se añade el botón "Calcular ruta", para que cada vez que se pulse sobre él, se generen las rutas y mapa con los centros de salud y parámetros del algoritmo genético seleccionados en ese momento.

```
submitButton("Calcular ruta"),
```

Ilustración 26. Botón "Calcular ruta"

Por último, se crea un panel principal ("mainPanel") que contiene la tabla que devuelve las distintas rutas obtenidas con los distintos algoritmos utilizados y el mapa interactivo con la ruta más corta calculada.

```
mainPanel(
          tableOutput("values"),
          leafletOutput("mapa"))
)
```

Ilustración 27. Panel principal de rutas y mapa

4.5.4. Función del servidor "server"

Esta parte del código permite la generación de los cálculos necesarios en para el funcionamiento de la aplicación, resolución del problema de optimización y generación de mapas.

4.5.4.1. Función para el cálculo de la optimización de rutas

Se define la primera función reactiva "slidervalues", que realiza el cálculo de la optimización y devuelve la lista de las rutas.

```
server<-function(input,output){

  slidervalues<-reactive({
```

Ilustración 28. Función "slidervalues"

Se crea la "MatrizTrabajo" como un subconjunto de la matriz "Distancias", con la Plataforma Logística del Salud y los centros de salud seleccionados en las pestañas creadas en el "ui".

```
   MatrizTrabajo<-Distancias[

c("PLATAFORMA LOGÍSTICA DEL SALUD",c(input$variable1),
c(input$variable2),c(input$variable3),
c(input$variable4),c(input$variable5),c(input$variable6)),

c("PLATAFORMA LOGÍSTICA DEL SALUD",c(input$variable1),
c(input$variable2),c(input$variable3),
c(input$variable4),c(input$variable5), c(input$variable6))]
```

Ilustración 29. Creación "Matriz Trabajo"

En el caso de que no se haya seleccionado ningún centro de salud, la ruta únicamente está formada por el almacén de reparto. De esta manera, si se pulsa sobre el botón "Calcular ruta", salta un mensaje de aviso "Elegir como mínimo un centro de salud para obtener ruta", y se detiene.

```
  if(length(c("PLATAFORMA LOGÍSTICA DEL SALUD",c(input$variable1),
              c(input$variable2),c(input$variable3),
              c(input$variable4),c(input$variable5),
              c(input$variable6)))<2){
    stop("Elegir como mínimo un centro de salud para obtener ruta")}
```

Ilustración 30. Condicional de tamaño mínimo de ruta

Si se ha seleccionado al menos un centro de salud al que repartir, se continúa con los cálculos.

Cabe destacar una característica de la aplicación, y es que para rutas que contengan menos de 9 centros de salud a los que repartir, a los que hay que añadir la Plataforma Logística del Salud; es decir, un total de 10 localizaciones que unir, la aplicación sólo calcula las rutas mediante el algoritmo del vecino más lejano y el más cercano.

Esto se debe a que las características de ejecución y cálculo del algoritmo genético utilizado no permiten unir menos de 10 localizaciones, destacando el 4-intercambio que realiza en la fase de mutación de soluciones, como principal impedimento. Este operador genético necesita un número mínimo de ciudades para tener la posibilidad de eliminar 4 aristas no coincidentes de unión de los centros de salud.

Si el número de localizaciones a unir es menor que 10, se obtienen la matriz de distancias simétrica "MatrizTrabajo" y se convierte en un objeto TSP para poder utilizarla en la búsqueda de soluciones con la librería TSP. Además, se indica que el punto de partida de la ruta es la Plataforma Logística del Salud.

```
     else{

    if(length(c("PLATAFORMA LOGÍSTICA DEL SALUD",c(input$variable1),
              c(input$variable2),c(input$variable3),
              c(input$variable4),c(input$variable5),
              c(input$variable6)))<10){


MatrizTrabajo[lower.tri(MatrizTrabajo)]=t(MatrizTrabajo)[lower.tri(Matriz
Trabajo)]
MatrizTSP<-TSP(as.matrix(MatrizTrabajo), labels = NULL)
initialtour<-as.integer(which(labels(MatrizTrabajo)[[1]]=="PLATAFORMA
LOGÍSTICA DEL SALUD"))
```

Ilustración 31. "MatrizTrabajo" para rutas con menos de 10 puntos

Como se ha mencionado, se crean dos soluciones, la "solucion1" y la "solucion2", utilizando los algoritmos del vecino más lejano ("farthest_insertion") y del más cercano ("nearest_insertion") de la librería TSP, respectivamente.

Se extraen las longitudes ("Longitud1" y "Longitud2") obtenidas en ambas soluciones.

```
solucion1<-
solve_TSP((MatrizTSP),method="farthest_insertion",control=list(start=init
ialtour))
solucion2<-
solve_TSP((MatrizTSP),method="nearest_insertion",control=list(start=initi
altour))
Longitud1<-tour_length(solucion1)
Longitud2<-tour_length(solucion2)
```

Ilustración 32. Soluciones TSP

Por último, se crea y se devuelve en pantalla una lista denominada "Ruta" que contiene las longitudes obtenidas según el algoritmo del vecino más lejano y del más cercano, y los nombres de los centros de salud ordenados según las rutas obtenidas.

```
Ruta<-
list(c(paste("Longitud(km):",round(Longitud1,digits=2)),labels(solucion1)
),c(paste("Longitud (km):",round(Longitud2,digits=2)),labels(solucion2)))

Ruta}
```

Ilustración 33. Lista de rutas

El código anterior devuelve las rutas cuando hay menos de 9 centros de salud a los que suministrar material sanitario.

A continuación, en el siguiente fragmento de código, se presenta el cálculo de rutas para los casos en los que existan, al menos, 9 centros de salud. Como se ha mencionado anteriormente, en esta situación sí que se aplican los algoritmos genéticos, juntos con los algoritmos del vecino más lejano y del más cercano.

En este caso se añade la creación de la MatrizAG (en referencia a los Algoritmos Genéticos), a partir de la "MatrizTrabajo"; la función "search_tour_genetic" para buscar la solución mediante algoritmos genéticos, teniendo en cuenta los parámetros que se pueden seleccionar en las pestañas de la interfaz y, por último, la inclusión de esta solución en la tabla de distancias y rutas que devuelve la aplicación, en forma de lista con el nombre "Ruta".

```
    else {

MatrizTrabajo[lower.tri(MatrizTrabajo)]=t(MatrizTrabajo)[lower.tri(Matriz
Trabajo)]
      isSymmetric(MatrizTrabajo)
      MatrizAG<-MatrizTrabajo
      colnames(MatrizAG)<-NULL
      MatrizTSP<-TSP(as.matrix(MatrizTrabajo), labels = NULL)
      initialtour<-
as.integer(which(labels(MatrizTrabajo)[[1]]=="PLATAFORMA LOGÍSTICA DEL
SALUD"))

solucion1<-
solve_TSP((MatrizTSP),method="farthest_insertion",control=list(start=init
ialtour))
solucion2<-
solve_TSP((MatrizTSP),method="nearest_insertion",control=list(start=initi
altour))
solucion3<-
search_tour_genetic(MatrizAG,dim(MatrizAG)[1],input$Npop,input$Ngen,input
$Local,input$Elite)


Longitud1<-tour_length(solucion1)
Longitud2<-tour_length(solucion2)
Longitud3<-solucion3$distance

Ruta<-
list(c(paste("Longitud(km):",round(Longitud1,digits=2)),labels(solucion1)
),c(paste("Longitud(km):",round(Longitud2,digits=2)),labels(solucion2)),c
(paste("Longitud (km):",round(Longitud3,digits =
2)),colnames(MatrizTrabajo)[solucion3$tour]))

Ruta}
  }})
```

Ilustración 34. Cálculo para rutas con 10 o más puntos de unión (incluye algoritmo genético)

4.5.4.2. Función para la generación de mapas

La segunda función reactiva presente en la función del servidor "server" es "mapaselect". Esta función permite la creación de mapas interactivos con la librería "leaflet". Estos mapas se actualizarán cada vez que se genera una nueva ruta.

```
mapaselect<-reactive({
```

Ilustración 35. Función "mapaselect"

A continuación, se presenta el mismo condicionante que en la función "slidervalues", comentada anteriormente. Para rutas con menos de 9 centros de salud, únicamente se calculan las rutas con los algoritmos del vecino más cercano y del más lejano.

```
    if(length(c("PLATAFORMA LOGÍSTICA DEL SALUD",c(input$variable1),
                c(input$variable2),c(input$variable3),
                c(input$variable4),c(input$variable5),
                c(input$variable6)))<10){
```

Ilustración 36. Condicional para rutas que unen menos de 10 puntos

La única diferencia, respecto al código utilizado en la función reactiva "slidervalues", utilizada para generar la lista con las distintas rutas; es que se crea un objeto "coordenadas", como un subconjunto de la matriz "coordenadas", con las propias coordenadas de los centros de salud seleccionados, para cada sector de salud.

```
      coordenadas<-coordenadas[c("PLATAFORMA LOGÍSTICA DEL
SALUD",c(input$variable1), c(input$variable2),c(input$variable3),
c(input$variable4),c(input$variable5), c(input$variable6)),]

      coordenadas<-as.data.frame(coordenadas)
```

Ilustración 37. Subconjunto "coordenadas"

El resto del código generado para el cálculo de las rutas, según los dos algoritmos mencionados, es idéntico que, en la función reactiva anteriormente mencionada.

Se genera un subconjunto ("MatrizTrabajo") de la matriz de distancias "Distancias", con las distancias entre los centros de salud seleccionados y la Plataforma Logística del Salud, para generar la ruta.

```
      MatrizTrabajo<-Distancias[c("PLATAFORMA LOGÍSTICA DEL
SALUD",c(input$variable1), c(input$variable2),c(input$variable3),
c(input$variable4),c(input$variable5),c(input$variable6)),

c("PLATAFORMA LOGÍSTICA DEL SALUD",c(input$variable1),
c(input$variable2),c(input$variable3),c(input$variable4),c(input$variable
5),c(input$variable6))]


MatrizTrabajo[lower.tri(MatrizTrabajo)]=t(MatrizTrabajo)[lower.tri(Matriz
Trabajo)]
MatrizTSP<-TSP(as.matrix(MatrizTrabajo), labels = NULL)
```

Ilustración 38. Subconjunto "MatrizTrabajo"

Se indica que el origen es el almacén del SALUD, y finalmente se calcula la ruta según el algoritmo del vecino más lejano y del más cercano; extrayendo las respectivas longitudes.

```
initialtour<-as.integer(which(labels(MatrizTrabajo)[[1]]=="PLATAFORMA
LOGÍSTICA DEL SALUD"))

solucion1<-
solve_TSP((MatrizTSP),method="farthest_insertion",control=list(start=init
ialtour))
solucion2<-
solve_TSP((MatrizTSP),method="nearest_insertion",control=list(start=initi
altour))


Longitud1<-tour_length(solucion1)
Longitud2<-tour_length(solucion2)
```

Ilustración 39. Soluciones TSP

El siguiente paso, una vez se dispone de las rutas calculadas, es generar el mapa interactivo, con la ruta más corta obtenida.

Para ello se genera, de nuevo, una lista con las etiquetas de ambas soluciones, y se selecciona aquella ruta con menor longitud como "Rutafinal", para generar el mapa.

```
Ruta<-list(labels(solucion1),labels(solucion2))
Longitudes<-c(Longitud1,Longitud2)
Seleccion<-which.min(Longitudes)
Rutafinal<-Ruta[[Seleccion]]
```

Ilustración 40.Creación de "Ruta" y "Rutafinal"

A continuación, se seleccionan los centros de salud de la matriz "coordenadas" que estarán en la ruta, se ordenan y se convierten en un DataFrame, con el nombre de "coordenadas". Se crea el objeto "ruta" con las columnas de longitud y latitud de la matriz creada; y se asignan los nombres "LONGITUD" y "LATITUD" a las columnas.

Finalmente se crea el objeto "ID1" con los nombres de las filas de "coordenadas".

```
coordenadas<-coordenadas[order(Rutafinal),]
coordenadas<-as.data.frame(coordenadas)
ruta<-data.frame(coordenadas$LONGITUD,coordenadas$LATITUD)
names(ruta)<-c("LONGITUD","LATITUD")
ID1=rownames(coordenadas)
```

Ilustración 41. Creación de la lista "ruta" y objeto "ID1"

Por último, se genera el mapa.

Se establece la clave secreta de API de OpenRouteService para acceder a sus servicios de datos geográficos y enrutamiento.

```
ors_api_key('5b3ce3597851110001cf6248c461a41c55e84e7fac8efbda9e23fc4e')
```

Ilustración 42. Clave secreta de API de OpenRouteService

Se genera un objeto "route" con la ruta entre los centros de salud definidos en el objeto "ruta".

```
route <- ors_directions(asplit(ruta,1),
format="geojson",radiuses=1000000)
```

Ilustración 43. Objeto "route"

Se crea el "mapa" interactivo gracias a la librería "leaflet" utilizando las coordenadas de los centros de salud que forman la ruta, y se

establece que el mapa generado se centre inicialmente en el primer punto de la variable "coordenadas", el cual es la Plataforma Logística del Salud.

```
mapa <- leaflet(data = coordenadas) %>%
        addTiles() %>%
        setView(lng = coordenadas$LONGITUD[1],lat=coordenadas$LATITUD[1]
, zoom = 1) %>%
```

Ilustración 44. Creación del mapa interactivo "mapa"

Finalmente, se añaden marcadores a los puntos del mapa según lo establecido en el objeto "ID1", que contiene el nombre de los centros de salud, y se realizan ciertos ajustes al mapa.

```
addMarkers(~LONGITUD, ~LATITUD, popup = ~as.character(ID1), label =
~as.character(ID1))%>%
addGeoJSON(route, fill=FALSE) %>%
fitBBox(route$bbox)}
```

Ilustración 45. Marcadores del mapa y ajustes

El código anterior se enfoca en la creación de mapas, cuando el número de centros de salud seleccionados es menor que 9.

A continuación, se presenta el código para el caso en el que se hayan seleccionado más de 9 centros de salud.

La estructura y operaciones son idénticas, exceptuando que, en este caso, la ruta también se calcula con el algoritmo genético, y se tendrá en cuenta el resultado obtenido con este método.

```
    else{
    coordenadas<-coordenadas[c("PLATAFORMA LOGÍSTICA DEL
SALUD",c(input$variable1), c(input$variable2),c(input$variable3),
(input$variable4),c(input$variable5), c(input$variable6)),]

    coordenadas<-as.data.frame(coordenadas)
    MatrizTrabajo<-Distancias[c("PLATAFORMA LOGÍSTICA DEL
SALUD",c(input$variable1),c(input$variable2),c(input$variable3),c(input$v
ariable4),c(input$variable5),c(input$variable6)),

c("PLATAFORMA LOGÍSTICA DEL SALUD",c(input$variable1),
c(input$variable2),c(input$variable3),c(input$variable4),c(input$variable
5),(input$variable6))]


MatrizTrabajo[lower.tri(MatrizTrabajo)]=t(MatrizTrabajo)[lower.tri(Matriz
Trabajo)]
isSymmetric(MatrizTrabajo)
MatrizAG<-MatrizTrabajo
colnames(MatrizAG)<-NULL
MatrizTSP<-TSP(as.matrix(MatrizTrabajo), labels = NULL)
initialtour<-as.integer(which(labels(MatrizTrabajo)[[1]]=="PLATAFORMA
LOGÍSTICA DEL SALUD"))
```

Ilustración 46. Creación de "MatrizTrabajo" para rutas con 10 o más puntos de unión

Hasta aquí, el código es muy similar. Únicamente se renombrado la "MatrizTrabajo" como "MatrizAG", haciendo referencia a que será la que se utilice en el cálculo con algoritmos genéticos.

Posteriormente, tal y como se ha indicado, también se introduce la "solucion3" de la ruta como resultado a la ruta calculada con algoritmos genéticos, utilizando la función "search_tour_genetic".

```
solucion1<-
solve_TSP((MatrizTSP),method="farthest_insertion",control=list(start=init
ialtour))
solucion2<-
solve_TSP((MatrizTSP),method="nearest_insertion",control=list(start=initi
altour))
solucion3<-
search_tour_genetic(MatrizAG,dim(MatrizAG)[1],input$Npop,input$Ngen,input
$Local,input$Elite)

Longitud1<-tour_length(solucion1)
Longitud2<-tour_length(solucion2)
Longitud3<-solucion3$distance
```

Ilustración 47. Soluciones TSP y algoritmo genético

En la "Rutafinal", con la que se genera el mapa, se tiene en cuenta la selección de la ruta más corta obtenida de las soluciones obtenidas, en esta ocasión mediante el algoritmo del vecino más lejano, del más cercano y genético.

```
Ruta<-
list(labels(solucion1),labels(solucion2),colnames(MatrizTrabajo)[solucion
3$tour])
Longitudes<-c(Longitud1,Longitud2,Longitud3)
Seleccion<-which.min(Longitudes)
Rutafinal<-Ruta[[Seleccion]]
```

Ilustración 48. Ruta final mapa

El resto de código para generar el mapa con la ruta es igual que el desarrollado para el caso anterior (rutas con menos de 9 centros de salud).

```
ors_api_key('5b3ce3597851110001cf6248c461a41c55e84e7fac8efbda9e23fc4e')
    route <- ors_directions(asplit(ruta,1),
format="geojson",radiuses=1000000)
mapa <- leaflet(data = coordenadas) %>%
addTiles() %>%
setView(lng = coordenadas$LONGITUD[1], lat = coordenadas$LATITUD[1] ,
zoom = 1) %>%
addMarkers(~LONGITUD, ~LATITUD, popup = ~as.character(ID1), label =
~as.character(ID1))%>%
      addGeoJSON(route, fill=FALSE) %>%
      fitBBox(route$bbox)
    }})
```

Ilustración 49. Creación del mapa interactivo "mapa"

La función del servidor "server" finaliza representando los resultados obtenidos.

Primero muestra en forma de tabla los resultados de las rutas obtenidas, tanto la distancia, como el orden de los centros de salud a visitar.

Si la ruta contiene menos de 9 centros de salud, devuelve una tabla con 2 columnas, para los resultados del algoritmo del vecino más lejano y del más cercano.

```
  output$values<-renderTable({

    Tabla<-data.frame(slidervalues())

    if(length(c("PLATAFORMA LOGÍSTICA DEL SALUD",c(input$variable1),
                c(input$variable2),c(input$variable3),
                c(input$variable4),c(input$variable5),
                c(input$variable6)))<10){

    names(Tabla)<-c("ALGORITMO DEL VECINO MÁS LEJANO","ALGORITMO DEL
VECINO MÁS CERCANO")
                Tabla}
```

Ilustración 50. Tabla rutas TSP

Para los casos en los que se hayan seleccionado más de 9 centros de salud, la tabla tendrá 3 columnas, incluyendo también el resultado obtenido con el algoritmo genético.

```
    else{
      names(Tabla)<-c("ALGORITMO DEL VECINO MÁS LEJANO","ALGORITMO DEL
VECINO MÁS CERCANO","ALGORITMO GENÉTICO")
    Tabla}

     })
```

Ilustración 51. Tabla rutas TSP y algoritmo genético

Finalmente genera el mapa interactivo con los centros de salud que se han seleccionado, mostrando la ruta a través de carreteras y calles.

```
  output$mapa<-renderLeaflet({mapaselect()})
}
```

Ilustración 52. Representación del mapa interactivo

4.5.5. Función "shinyApp"

Por último, se ejecuta la función "shinyApp", la cual crea y abre la aplicación con la que el usuario podrá interactuar.

```
shinyApp(ui = ui, server = server)
```

Ilustración 53. Función "shinyApp"

4.6. INTERFAZ DE LA APLICACIÓN

Este apartado tiene la intención de mostrar la interfaz de la aplicación creada y de explicar su utilización.

Como se podrá observar, la aplicación ha sido creada con la intención de ser intuitiva y de fácil utilización.

4.6.1. Aspecto global

Al abrir la aplicación para la optimización de reparto de material sanitario entre los centros de salud de Zaragoza, se presenta la siguiente interfaz:

Ilustración 54. Aspecto global de la aplicación

Como se puede observar, consta de:

- Un título referido a la funcionalidad de la aplicación ("Ruta de reparto de material sanitario en Zaragoza").
- Un conjunto de pestañas referidas a cada uno de los sectores de salud que contienen centros de salud en Zaragoza.
- Una pestaña de "Parámetros algoritmo genético".
- Un botón de "Calcular ruta"

4.6.1.1. Pestañas de sectores de salud

Cada una de estas pestañas contiene el conjunto de casillas referidas a los centros de salud que contiene, y que se encuentran en Zaragoza.

Estas casillas tienen la finalidad de seleccionar aquellos centros de salud que se desean abastecer con material sanitario.

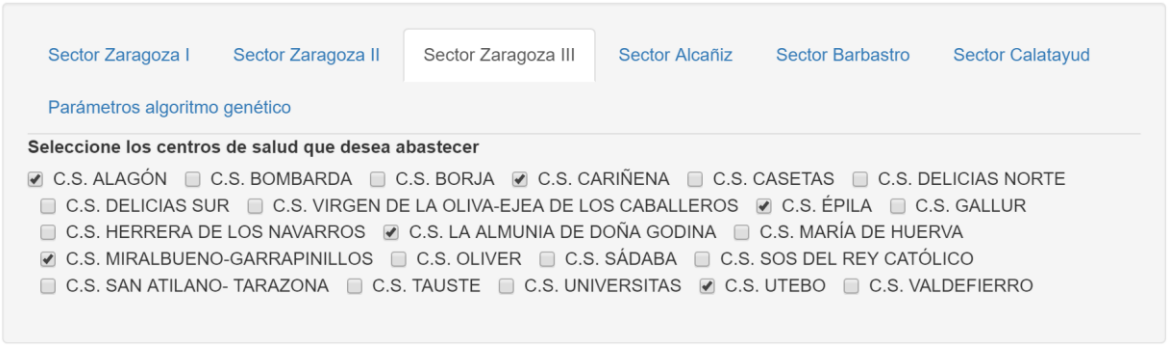

Ilustración 55. Pestañas de sectores de salud

4.6.1.2. Pestaña de parámetros del algoritmo genético

Esta pestaña muestra aquellos parámetros modificables de la función que resuelve el problema de optimización con el algoritmo genético.

El ajuste de estos parámetros dependerá del interés del usuario, que buscará encontrar un equilibrio entre la obtención de una buena solución y el tiempo de ejecución del algoritmo.

Ilustración 56. Pestaña de parámetros del algoritmo genético

4.6.1.3. Botón "Calcular ruta"

Este botón tiene la finalidad de que, una vez seleccionados los centros de salud y los parámetros del algoritmo genético, la aplicación comience a realizar los cálculos necesarios en búsqueda de soluciones y devuelva las rutas en forma de lista, junto con el mapa.

Ilustración 57. Botón para calcular las rutas

4.6.2. Representación de soluciones

Una vez se han seleccionado los centros de salud que se quieren abastecer, los parámetros del algoritmo genético y se ha pulsado sobre el botón "Calcular ruta"; la aplicación devuelve una lista con las rutas obtenidas y el mapa con la ruta más corta.

4.6.2.1. Lista de rutas

Bajo las pestañas de selección y el botón de "Calcular ruta", se presenta una lista con las rutas obtenidas en el cálculo.

Muestran los algoritmos utilizados en la búsqueda de la solución, la longitud de la ruta obtenida y su respectiva ordenación de los centros de salud a visitar.

ALGORITMO DEL VECINO MÁS LEJANO	ALGORITMO DEL VECINO MÁS CERCANO	ALGORITMO GENÉTICO
Longitud (km): 505.47	Longitud (km): 528.56	Longitud (km): 498.05
PLATAFORMA LOGÍSTICA DEL SALUD	PLATAFORMA LOGÍSTICA DEL SALUD	PLATAFORMA LOGÍSTICA DEL SALUD
C.S. UTEBO	C.S. MIRALBUENO-GARRAPINILLOS	C.S. MIRALBUENO-GARRAPINILLOS
C.S. MIRALBUENO-GARRAPINILLOS	C.S. ACTUR NORTE	C.S. ALMOZARA
C.S. ALMOZARA	C.S. VILLAMAYOR	C.S. ACTUR NORTE
C.S. ACTUR NORTE	C.S. SANTA ISABEL	C.S. PICARRAL
C.S. PICARRAL	C.S. PICARRAL	C.S. SANTA ISABEL
C.S. SANTA ISABEL	C.S. ALMOZARA	C.S. VILLAMAYOR
C.S. VILLAMAYOR	C.S. UTEBO	C.S. MEQUINENZA

Ilustración 58. Lista de rutas generadas

4.6.2.2. Mapa interactivo

Finalmente, tras la representación de la lista de rutas obtenidas, se presenta un mapa interactivo con la mejor ruta obtenida.

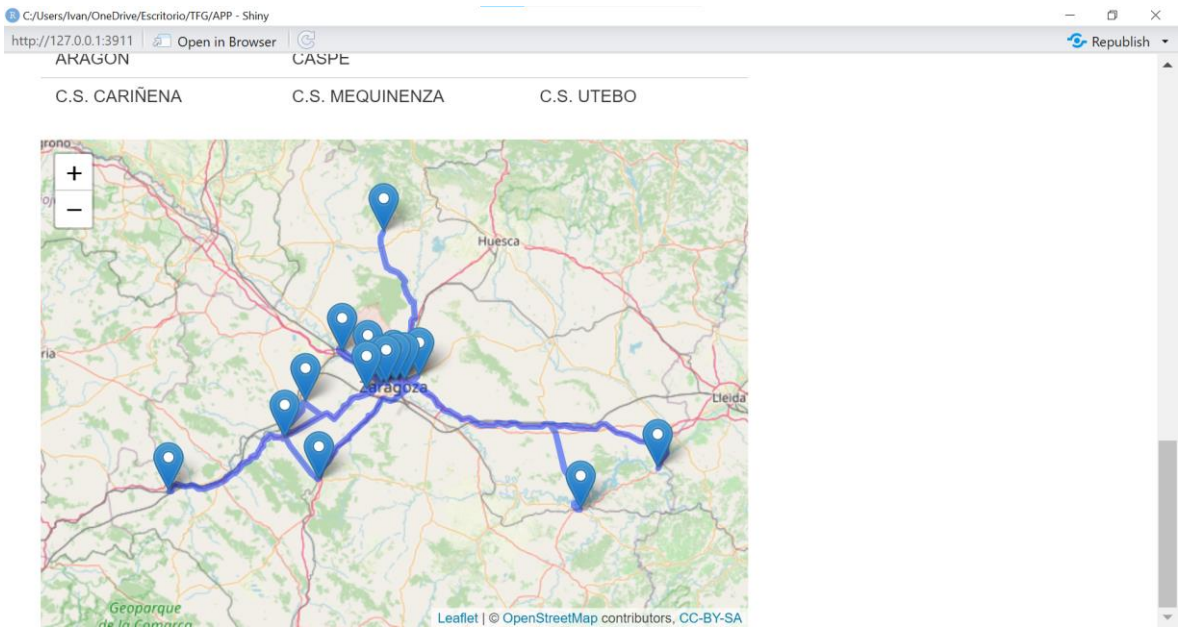

Ilustración 59. Mapa interactivo

4.7. ACCESO LIBRE A LA APLICACIÓN

La aplicación queda depositada en la nube, en un repositorio shinyapps.io para que todo aquel interesado pueda utilizarla libremente.

Se puede acceder a través del siguiente enlace:
https://rutarepartocs.shinyapps.io/Opt_reparto_CS_Zgz/

También se puede acceder escaneando el siguiente código QR:

Ilustración 60. Código QR Aplicación

5. RESULTADOS

A continuación, se presentan, comentan y analizan los resultados obtenidos con el uso de la aplicación creada para optimizar la ruta de reparto de material sanitario entre los centros de salud de Zaragoza.

Primero se presentan y se comentan los resultados obtenidos en unos ejemplos aleatorios de rutas, obtenidas a través de los distintos algoritmos.

Posteriormente se realiza un análisis comparativo de los resultados obtenidos con los tres tipos de algoritmos (vecino más lejano, vecino más cercano y genético), para rutas que visitan distinto número de centros de salud.

5.1. EJEMPLOS DE RUTAS ALEATORIAS

A continuación, se presentan y comentan ejemplos de resultados obtenidos en el cálculo de rutas de reparto aleatorias, que suministran a distinto número de centros de salud, desde el almacén de material sanitario.

5.1.1. Ejemplo para 4 centros de salud

Tal y como se ha mencionado anteriormente, para rutas con menos de 9 centros de salud, la ruta se resuelve únicamente con los algoritmos del vecino más lejano y del más cercano.

Para este ejemplo, las rutas obtenidas por ambos métodos tienen la misma longitud (30.19 km). Sin embargo, se puede observar que el orden de los centros de salud a visitar es opuesto; esto significa que la ruta tiene el sentido contrario.

En este caso, se podría elegir cualquiera de las dos rutas, ya que ninguna, a priori, resulta más beneficiosa.

ALGORITMO DEL VECINO MÁS LEJANO	ALGORITMO DEL VECINO MÁS CERCANO
Longitud (km): 30.19	Longitud (km): 30.19
PLATAFORMA LOGÍSTICA DEL SALUD	PLATAFORMA LOGÍSTICA DEL SALUD
C.S. PARQUE GOYA	C.S. ACTUR SUR
C.S. SANTA ISABEL	C.S. PICARRAL

ALGORITMO DEL VECINO MÁS LEJANO	ALGORITMO DEL VECINO MÁS CERCANO
C.S. PICARRAL	C.S. SANTA ISABEL
C.S. ACTUR SUR	C.S. PARQUE GOYA

Tabla 3. Ejemplo de ruta con 4 centros de salud

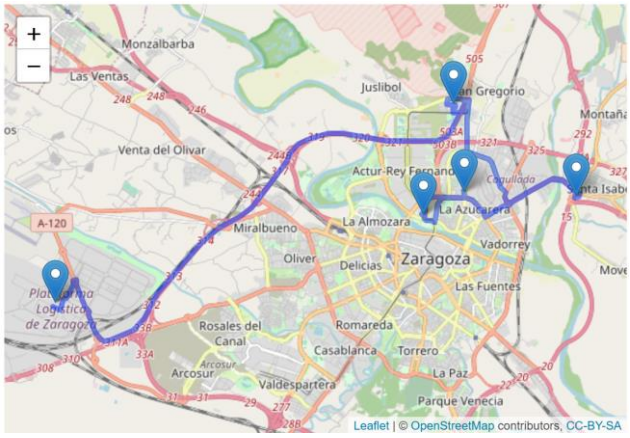

Ilustración 61. Ejemplo de mapa con 4 centros de salud

5.1.2. Ejemplo para 9 centros de salud

Cuando se trata de calcular una ruta que contiene un número mayor a 9 centros de salud, ya se puede aplicar el algoritmo genético; obteniendo un total de 3 rutas diferentes.

Como se puede observar, para esta ruta se obtienen distintas distancias entre el algoritmo del vecino más lejano y del más cercano (31.46 km y 33.86 km respectivamente).

Cabe destacar que, con el algoritmo genético, para el caso de esta ruta en particular, no se ha obtenido un mejor resultado; ya que se obtiene una ruta con la misma longitud que la obtenida con el algoritmo del vecino más lejano.

En este caso, la ruta obtenida mediante el algoritmo del vecino más lejano y el algoritmo genético es la misma, pero de sentido contrario; por lo que se podría elegir cualquiera de las dos como mejor opción.

ALGORITMO DEL VECINO MÁS LEJANO	ALGORITMO DEL VECINO MÁS CERCANO	ALGORITMO GENÉTICO
Longitud (km): 31.46	Longitud (km): 33.86	Longitud (km): 31.46
PLATAFORMA LOGÍSTICA DEL SALUD	PLATAFORMA LOGÍSTICA DEL SALUD	PLATAFORMA LOGÍSTICA DEL SALUD
C.S. PARQUE ROMA	C.S. AMPARO POCH (ACTUR OESTE)	C.S. AMPARO POCH (ACTUR OESTE)
C.S. ACTUR SUR	C.S. ACTUR NORTE	C.S. ACTUR NORTE

ALGORITMO DEL VECINO MÁS LEJANO	ALGORITMO DEL VECINO MÁS CERCANO	ALGORITMO GENÉTICO
C.S. ARRABAL	C.S. PARQUE GOYA	C.S. PARQUE GOYA
C.S. PICARRAL	C.S. ACTUR SUR	C.S. SANTA ISABEL
C.S. LA JOTA	C.S. PICARRAL	C.S. LA JOTA
C.S. SANTA ISABEL	C.S. SANTA ISABEL	C.S. PICARRAL
C.S. PARQUE GOYA	C.S. LA JOTA	C.S. ARRABAL
C.S. ACTUR NORTE	C.S. ARRABAL	C.S. ACTUR SUR
C.S. AMPARO POCH (ACTUR OESTE)	C.S. PARQUE ROMA	C.S. PARQUE ROMA

Tabla 4. Ejemplo de ruta con 9 centros de salud

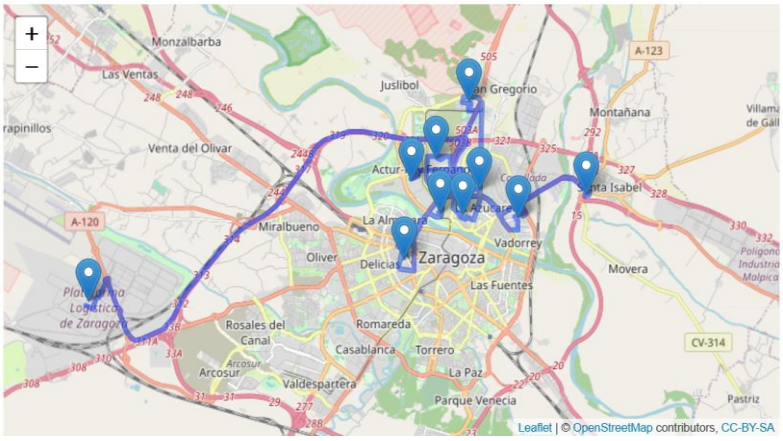

Tabla 5. Ejemplo de mapa con 9 centros de salud

A continuación, se presenta la ejecución y resultados generados con el algoritmo genético.

Como se puede observar se han elegido una serie de parámetros, los cuales se mantienen para el resto de los ejemplos de rutas calculadas posteriormente:

- Población inicial de 6 soluciones en cada generación (Npop = 6)
- Número de generaciones que resuelve es 10 ([Gen = 1], [Gen = 2], …, [Gen = 10])

Para este ejemplo en concreto, en el cruce (Crossover) de cada generación obtiene 6 soluciones hijas (6 offsprings).

En las mutaciones de cada generación obtiene 12 nuevas soluciones (12 mutants).

En las búsquedas locales, solamente en la de la primera generación obtiene 2 2-minima nuevas soluciones.

Finalmente, como se puede observar, la solución de distancia mínima que ha obtenido en la primera generación ([Lmin =

31.46414(km)]), es la que se obtiene en cada una de las siguientes generaciones, lo que indica que durante el proceso de búsqueda, esta solución ha conseguido "sobrevivir" las 10 generaciones, sin encontrarse una mejor solución; a pesar de todas las soluciones generadas en las sucesivas generaciones.

```
Generated initial population (Npop = 6)
[Gen = 1] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(2 2-minima)
Selection...done [Lmin = 31.46414]
[Gen = 2] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(0 2-minima)
Selection...done [Lmin = 31.46414]
[Gen = 3] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(0 2-minima)
Selection...done [Lmin = 31.46414]
[Gen = 4] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(0 2-minima)
Selection...done [Lmin = 31.46414]
[Gen = 5] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(0 2-minima)
Selection...done [Lmin = 31.46414]
[Gen = 6] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(0 2-minima)
Selection...done [Lmin = 31.46414]
[Gen = 7] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(0 2-minima)
Selection...done [Lmin = 31.46414]
[Gen = 8] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(0 2-minima)
Selection...done [Lmin = 31.46414]
[Gen = 9] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(0 2-minima)
Selection...done [Lmin = 31.46414]
[Gen = 10] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(0 2-minima)
Selection...done [Lmin = 31.46414]
```

Ilustración 62. Cálculo del algoritmo genético para un ejemplo de ruta con 9 centros de salud

5.1.3. Ejemplo para 33 centros de salud

Para este ejemplo de ruta aleatoria que une 33 centros de salud, ya es posible observar los beneficios que el algoritmo genético aporta en la obtención de mejores soluciones.

Además, cabe destacar que el algoritmo del vecino más lejano, como en el ejemplo anterior para 9 centros de salud, ha generado una mejor solución que el algoritmo del vecino más cercano.

Con la solución obtenida del algoritmo genético, se consigue reducir la longitud de la ruta 10.39 km y 85.59 km, respecto a la obtenida con el algoritmo del vecino más lejano y del más cercano, respectivamente.

ALGORITMO DEL VECINO MÁS LEJANO	ALGORITMO DEL VECINO MÁS CERCANO	ALGORITMO GENÉTICO
Longitud (km): 685.55	Longitud (km): 760.75	Longitud (km): 675.16
PLATAFORMA LOGÍSTICA DEL SALUD	PLATAFORMA LOGÍSTICA DEL SALUD	PLATAFORMA LOGÍSTICA DEL SALUD
C.S. CASETAS	C.S. CASETAS	C.S. CASETAS

ALGORITMO DEL VECINO MÁS LEJANO	ALGORITMO DEL VECINO MÁS CERCANO	ALGORITMO GENÉTICO
C.S. OLIVER	C.S. OLIVER	C.S. ZUERA
C.S. ALMOZARA	C.S. ALMOZARA	C.S. LUNA
C.S. CANAL IMPERIAL	C.S. AMPARO POCH (ACTUR OESTE)	C.S. SOS DEL REY CATÓLICO
C.S. ARRABAL	C.S. ACTUR NORTE	C.S. VIRGEN DE LA OLIVA-EJEA DE LOS CABALLEROS
C.S. ACTUR SUR	C.S. PARQUE GOYA	C.S. TAUSTE
C.S. AMPARO POCH (ACTUR OESTE)	C.S. PICARRAL	C.S. BORJA
C.S. ACTUR NORTE	C.S. ALFAJARÍN	C.S. MORATA DE JALÓN
C.S. PARQUE GOYA	C.S. AMANDO LORIGA-CASPE	C.S. SABIÑAN
C.S. PICARRAL	C.S. MAELLA	C.S. VILLARROYA DE LA SIERRA
C.S. LA JOTA	C.S. MEQUINENZA	C.S. ATECA
C.S. SANTA ISABEL	C.S. BUJARALOZ	C.S. ALHAMA DE ARAGÓN
C.S. VILLAMAYOR	C.S. VILLAMAYOR	C.S. DAROCA
C.S. ALFAJARÍN	C.S. ZUERA	C.S. CARIÑENA
C.S. BUJARALOZ	C.S. SANTA ISABEL	C.S. HERRERA DE LOS NAVARROS
C.S. AMANDO LORIGA-CASPE	C.S. LA JOTA	C.S. AMANDO LORIGA-CASPE
C.S. MAELLA	C.S. ARRABAL	C.S. MAELLA
C.S. MEQUINENZA	C.S. ACTUR SUR	C.S. MEQUINENZA
C.S. ZUERA	C.S. CANAL IMPERIAL	C.S. BUJARALOZ
C.S. LUNA	C.S. MARÍA DE HUERVA	C.S. ALFAJARÍN
C.S. SOS DEL REY CATÓLICO	C.S. HERRERA DE LOS NAVARROS	C.S. VILLAMAYOR
C.S. VIRGEN DE LA OLIVA-EJEA DE LOS CABALLEROS	C.S. CARIÑENA	C.S. SANTA ISABEL
C.S. TAUSTE	C.S. DAROCA	C.S. LA JOTA
C.S. BORJA	C.S. SABIÑAN	C.S. ARRABAL
C.S. MORATA DE JALÓN	C.S. ATECA	C.S. ACTUR SUR
C.S. SABIÑAN	C.S. ALHAMA DE ARAGÓN	C.S. PICARRAL
C.S. VILLARROYA DE LA SIERRA	C.S. VILLARROYA DE LA SIERRA	C.S. PARQUE GOYA
C.S. ATECA	C.S. MORATA DE JALÓN	C.S. ACTUR NORTE
C.S. ALHAMA DE ARAGÓN	C.S. BORJA	C.S. AMPARO POCH (ACTUR OESTE)
C.S. DAROCA	C.S. TAUSTE	C.S. ALMOZARA
C.S. HERRERA DE LOS NAVARROS	C.S. VIRGEN DE LA OLIVA-EJEA DE LOS CABALLEROS	C.S. OLIVER
C.S. CARIÑENA	C.S. SOS DEL REY CATÓLICO	C.S. CANAL IMPERIAL
C.S. MARÍA DE HUERVA	C.S. LUNA	C.S. MARÍA DE HUERVA

Tabla 6. Ejemplo de ruta con 33 centros de salud

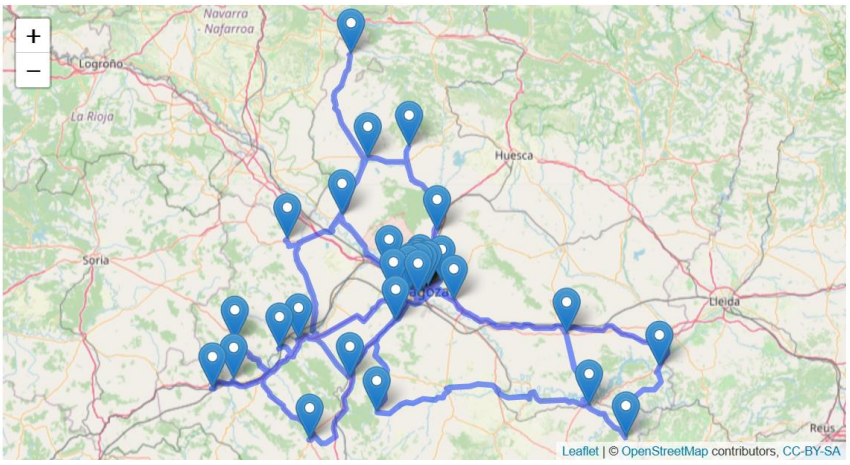

Ilustración 63. Ejemplo de mapa con 4 centros de salud

En este ejemplo, se puede observar que el algoritmo genético vuelve a encontrar la solución final en la primera generación de búsqueda ([Lmin = 675.1631(km)]). Esto significa, como se ha comentado en el ejemplo anterior, que esta solución consigue "sobrevivir" a las distintas generaciones, sin encontrarse una mejor solución; a pesar de todas las soluciones generadas en las sucesivas generaciones.

Esta situación, muestra la gran eficacia que tienen los algoritmos genéticos en la obtención de buenas soluciones al TSP, debido a que en la primera generación ya se había obtenido la mejor solución; y las demás generaciones han servido para tratar de combinar y generar mejores soluciones, en búsqueda de una todavía mejor, sin éxito.

```
Generated initial population (Npop = 6)
[Gen = 1] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(4 2-minima)
Selection...done [Lmin = 675.1631]
[Gen = 2] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(3 2-minima)
Selection...done [Lmin = 675.1631]
[Gen = 3] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(2 2-minima)
Selection...done [Lmin = 675.1631]
[Gen = 4] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(2 2-minima)
Selection...done [Lmin = 675.1631]
[Gen = 5] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(2 2-minima)
Selection...done [Lmin = 675.1631]
[Gen = 6] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(1 2-minima)
Selection...done [Lmin = 675.1631]
[Gen = 7] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(3 2-minima)
Selection...done [Lmin = 675.1631]
[Gen = 8] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(3 2-minima)
Selection...done [Lmin =675.1631]
[Gen = 9] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(2 2-minima)
Selection...done [Lmin = 675.1631]
[Gen = 10] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(2 2-minima)
Selection...done [Lmin =675.1631]
```

Ilustración 64. Cálculo del algoritmo genético para un ejemplo de ruta con 33 centros de salud

5.1.4. Ejemplo para 65 centros de salud

Como último ejemplo de ruta aleatoria, se genera la que une los 65 centros de salud presentes en Zaragoza.

De nuevo, el algoritmo del vecino más lejano obtiene una mejor ruta que el algoritmo del vecino más cercano, siendo sus longitudes 827.38 km y 915.58 km, respectivamente; consiguiendo un ahorro de 88.2 km.

Como cabía esperar, el algoritmo genético vuelve a obtener la mejor ruta, con una longitud de 805.14 km, suponiendo un ahorro de 22.24 km y 110.44 km, respecto al algoritmo del vecino más lejano y del más cercano.

ALGORITMO DEL VECINO MÁS LEJANO	ALGORITMO DEL VECINO MÁS CERCANO	ALGORITMO GENÉTICO
Longitud (km): 827.38	Longitud (km): 915.58	Longitud (km): 805.14
PLATAFORMA LOGÍSTICA DEL SALUD	PLATAFORMA LOGÍSTICA DEL SALUD	PLATAFORMA LOGÍSTICA DEL SALUD
C.S. MARÍA DE HUERVA	C.S. ÉPILA	C.S. MARÍA DE HUERVA
C.S. CAMPO DE BELCHITE	C.S. BUJARALOZ	C.S. VALDESPARTERA
C.S. SÁSTAGO	C.S. MEQUINENZA	C.S. CASABLANCA
C.S. AMANDO LORIGA-CASPE	C.S. MAELLA	C.S. ROMAREDA (SEMINARIO)
C.S. MAELLA	C.S. AMANDO LORIGA-CASPE	C.S. FERNANDO EL CATÓLICO
C.S. MEQUINENZA	C.S. SÁSTAGO	C.S. DELICIAS SUR
C.S. BUJARALOZ	C.S. CAMPO DE BELCHITE	C.S. UNIVERSITAS
C.S. FUENTES DE EBRO	C.S. HERRERA DE LOS NAVARROS	C.S. BOMBARDA
C.S. ALFAJARÍN	C.S. DAROCA	C.S. VALDEFIERRO
C.S. TORRE RAMONA	C.S. CARIÑENA	C.S. OLIVER
C.S. SAN JOSÉ	C.S. LA ALMUNIA DE DOÑA GODINA	C.S. MIRALBUENO-GARRAPINILLOS
C.S. TORRERO-LA PAZ	C.S. MORATA DE JALÓN	C.S. ALMOZARA
C.S. CANAL IMPERIAL	C.S. SABIÑAN	C.S. DELICIAS NORTE
C.S. ROMAREDA (SEMINARIO)	C.S. CALATAYUD	C.S. AMPARO POCH (ACTUR OESTE)
C.S. CASABLANCA	C.S. ATECA	C.S. ACTUR NORTE
C.S. VALDESPARTERA	C.S. ALHAMA DE ARAGÓN	C.S. PARQUE GOYA
C.S. VALDEFIERRO	C.S. ARIZA	C.S. PICARRAL
C.S. OLIVER	C.S. VILLARROYA DE LA SIERRA	C.S. ARRABAL
C.S. MIRALBUENO-GARRAPINILLOS	C.S. ILLUECA	C.S. ACTUR SUR
C.S. ALMOZARA	C.S. BORJA	C.S. SAN PABLO
C.S. DELICIAS NORTE	C.S. SAN ATILANO- TARAZONA	C.S. PUERTA DEL CARMEN
C.S. BOMBARDA	C.S. GALLUR	C.S. PARQUE ROMA
C.S. UNIVERSITAS	C.S. TAUSTE	C.S. JOSÉ R. MUÑOZ GERNANDEZ (SAGASTA)

ALGORITMO DEL VECINO MÁS LEJANO	ALGORITMO DEL VECINO MÁS CERCANO	ALGORITMO GENÉTICO
C.S. DELICIAS SUR	C.S. VIRGEN DE LA OLIVA-EJEA DE LOS CABALLEROS	C.S. CANAL IMPERIAL
C.S. PARQUE ROMA	C.S. SÁDABA	C.S. TORRERO-LA PAZ
C.S. PUERTA DEL CARMEN	C.S. SOS DEL REY CATÓLICO	C.S. TORRE RAMONA
C.S. FERNANDO EL CATÓLICO	C.S. LUNA	C.S. SAN JOSÉ
C.S. JOSÉ R. MUÑOZ FERNÁNDEZ (SAGASTA)	C.S. ALAGÓN	C.S. LAS FUENTES NORTE
C.S. LAS FUENTES NORTE	C.S. CASETAS	C.S. REBOLERÍA
C.S. REBOLERÍA	C.S. UTEBO	C.S. LA JOTA
C.S. ARRABAL	C.S. MIRALBUENO-GARRAPINILLOS	C.S. SANTA ISABEL
C.S. SAN PABLO	C.S. OLIVER	C.S. VILLAMAYOR
C.S. ACTUR SUR	C.S. ALMOZARA	C.S. ALFAJARÍN
C.S. AMPARO POCH (ACTUR OESTE)	C.S. BOMBARDA	C.S. FUENTES DE EBRO
C.S. ACTUR NORTE	C.S. DELICIAS NORTE	C.S. BUJARALOZ
C.S. PARQUE GOYA	C.S. PARQUE ROMA	C.S. MEQUINENZA
C.S. PICARRAL	C.S. AMPARO POCH (ACTUR OESTE)	C.S. MAELLA
C.S. LA JOTA	C.S. ACTUR NORTE	C.S. AMANDO LORIGA-CASPE
C.S. SANTA ISABEL	C.S. PARQUE GOYA	C.S. SÁSTAGO
C.S. VILLAMAYOR	C.S. ACTUR SUR	C.S. CAMPO DE BELCHITE
C.S. ZUERA	C.S. PICARRAL	C.S. HERRERA DE LOS NAVARROS
C.S. LUNA	C.S. ARRABAL	C.S. CARIÑENA
C.S. SOS DEL REY CATÓLICO	C.S. LA JOTA	C.S. DAROCA
C.S. SÁDABA	C.S. SANTA ISABEL	C.S. CALATAYUD
C.S. VIRGEN DE LA OLIVA-EJEA DE LOS CABALLEROS	C.S. ZUERA	C.S. ATECA
C.S. TAUSTE	C.S. VILLAMAYOR	C.S. ALHAMA DE ARAGÓN
C.S. GALLUR	C.S. ALFAJARÍN	C.S. ARIZA
C.S. BORJA	C.S. FUENTES DE EBRO	C.S. VILLARROYA DE LA SIERRA
C.S. SAN ATILANO- TARAZONA	C.S. SAN JOSÉ	C.S. ILLUECA
C.S. ILLUECA	C.S. TORRE RAMONA	C.S. SABIÑAN
C.S. MORATA DE JALÓN	C.S. LAS FUENTES NORTE	C.S. MORATA DE JALÓN
C.S. SABIÑAN	C.S. REBOLERÍA	C.S. LA ALMUNIA DE DOÑA GODINA
C.S. VILLARROYA DE LA SIERRA	C.S. SAN PABLO	C.S. ÉPILA
C.S. ARIZA	C.S. PUERTA DEL CARMEN	C.S. BORJA
C.S. ALHAMA DE ARAGÓN	C.S. JOSÉ R. MUÑOZ FERNÁNDEZ (SAGASTA)	C.S. SAN ATILANO-TARAZONA
C.S. ATECA	C.S. TORRERO-LA PAZ	C.S. GALLUR
C.S. CALATAYUD	C.S. CANAL IMPERIAL	C.S. TAUSTE
C.S. DAROCA	C.S. DELICIAS SUR	C.S. VIRGEN DE LA OLIVA-EJEA DE LOS CABALLEROS
C.S. HERRERA DE LOS NAVARROS	C.S. FERNANDO EL CATÓLICO	C.S. SÁDABA
C.S. CARIÑENA	C.S. ROMAREDA (SEMINARIO)	C.S. SOS DEL REY CATÓLICO
C.S. LA ALMUNIA DE DOÑA GODINA	C.S. CASABLANCA	C.S. LUNA
C.S. ÉPILA	C.S. VALDESPARTERA	C.S. ZUERA

ALGORITMO DEL VECINO MÁS LEJANO	ALGORITMO DEL VECINO MÁS CERCANO	ALGORITMO GENÉTICO
C.S. ALAGÓN	C.S. UNIVERSITAS	C.S. ALAGÓN
C.S. CASETAS	C.S. VALDEFIERRO	C.S. CASETAS
C.S. UTEBO	C.S. MARÍA DE HUERVA	C.S. UTEBO

Tabla 7. Ejemplo de ruta con los 65 centros de salud existentes en Zaragoza

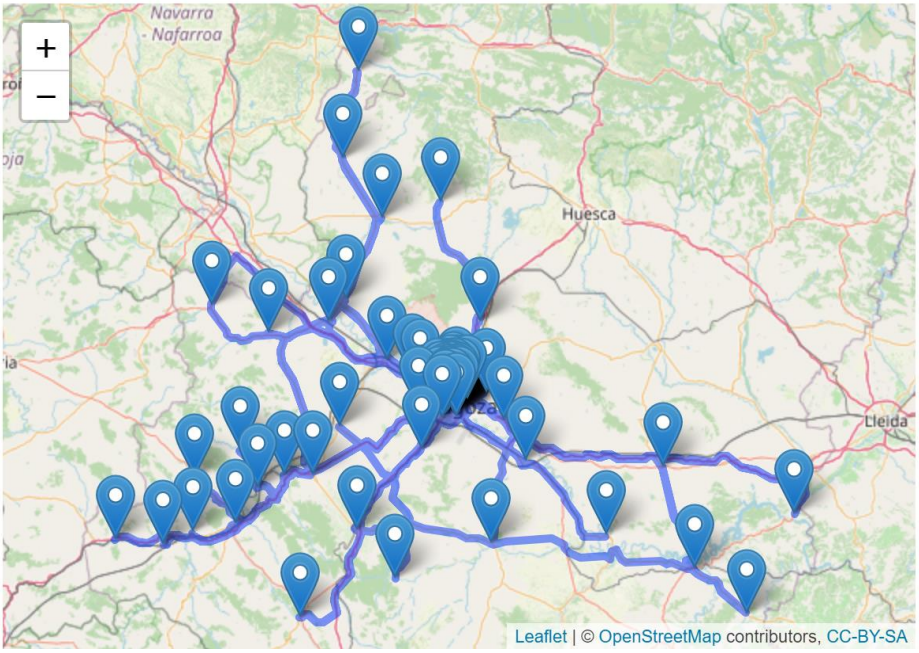

Ilustración 65. Ejemplo de mapa con los 65 centros de salud existentes en Zaragoza

En este caso, se puede observar que el algoritmo genético no obtiene la mejor solución en la primera generación. Sin embargo, esta primera solución que obtiene ([Lmin = 805.6461(km)]), es también mucho mejor que las obtenidas con los algoritmos del vecino más lejano (827.38 km) y el más cercano (915.58 km).

En las sucesivas generaciones, el algoritmo genético obtiene nuevas y mejores soluciones, hasta llegar a la cuarta generación ([Gen = 4]), en la que obtiene una mejor solución a la obtenida en la primera. Esta nueva mejor ruta ([Lmin = 802.9273(km)]), consigue una reducción en la distancia de 2.7188 km.

Dos generaciones más adelante ([Gen = 6]), el algoritmo genético consigue obtener la solución definitiva ([Lmin = 802.8811(km)]). En este caso el ahorro de distancia respecto a la solución anteriormente encontrada es despreciable (0.0462 km).

Finalmente, las generaciones de búsqueda posteriores no consiguen obtener rutas mejores.

```
Generated initial population (Npop = 6)
[Gen = 1] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(24 2-minima)
Selection...done [Lmin = 805.6461]
[Gen = 2] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(17 2-minima)
Selection...done [Lmin = 805.6461]
[Gen = 3] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(15 2-minima)
Selection...done [Lmin = 805.6461]
[Gen = 4] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(11 2-minima)
Selection...done [Lmin = 802.9273]
[Gen = 5] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(13 2-minima)
Selection...done [Lmin = 802.9273]
[Gen = 6] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(15 2-minima)
Selection...done [Lmin = 802.8811]
[Gen = 7] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(12 2-minima)
Selection...done [Lmin = 802.8811]
[Gen = 8] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(12 2-minima)
Selection...done [Lmin = 802.8811]
[Gen = 9] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(8 2-minima)
Selection...done [Lmin = 802.8811]
[Gen = 10] : Crossover...(6 offsprings) Mutation...(12 mutants) Local search...(13 2-minima)
Selection...done [Lmin = 802.8811]
```

Ilustración 66. Cálculo del algoritmo genético para un ejemplo de ruta con los 65 centros de salud existentes en Zaragoza

5.2. ANÁLISIS COMPARATIVO DE RESULTADOS

A continuación, se realiza un análisis comparativo de los resultados que se obtienen en la optimización de rutas aleatorias, con los 3 tipos de algoritmos utilizados en este TFG (algoritmo del vecino más lejano, algoritmo del vecino más cercano y algoritmo genético), con la finalidad de establecer una clasificación final, en función de la eficacia de cada uno.

De nuevo, debido a las características del algoritmo genético, este apartado comprende, por una parte, las rutas con un número de centros de salud inferior a 9, y el resto de las rutas, formadas por 9 o más centros de salud.

Cabe destacar que para cada tamaño de ruta "n", se han realizado 10 muestras de rutas aleatorias, y las tablas que se muestran en este apartado muestran la media de la variación entre las distancias obtenidas con los distintos algoritmos para cada "n".

Para una mayor información se presentan las tablas de los anexos, en la que se muestran las distancias de todas y cada una de las rutas generadas aleatoriamente, sus respectivas variaciones y la media.

5.2.1. Rutas con menos de 9 centros de salud

Para realizar este análisis de resultados, se han generado rutas de forma aleatoria, de tamaño máximo "n" igual a 9.

Por lo tanto, el análisis comienza con la generación de rutas que unen 4 localizaciones ("n=4", 3 centros de salud y el almacén), hasta rutas que unen 9 localizaciones ("n"=9).

El motivo por el que se comienzan a crear rutas con 3 centros de salud y el almacén, es porque para un tamaño de ruta menor no tiene sentido la comparación, ya que solo existe una posible ruta.

Como se ha explicado en varias ocasiones, para rutas de estos tamaños, la optimización se resuelve con el algoritmo del vecino más lejano y del más cercano.

5.2.1.1. Algoritmo del vecino más lejano vs algoritmo del vecino más cercano

Tras generar 10 rutas aleatorias para cada tamaño de ruta "n", las medias en las variaciones de las distancias de ruta, obtenidas con los dos algoritmos, se muestran a continuación:

n	LEJANO VS CERCANO
4	0,00%
5	0,10%
6	0,39%
7	0,61%
8	1,90%
9	3,02%
PROMEDIO	**1,00%**

Tabla 8. Variaciones entre el algoritmo del vecino más lejano y el del vecino más cercano

Se puede comprobar, como, de media, en ningún caso se obtienen resultados mejores con el algoritmo del vecino más cercano. Únicamente se obtienen los mismos resultados de rutas con ambos algoritmos, para un tamaño de ruta "n"=4. Para el resto de casos, hasta rutas con 9 localizaciones ("n"=9), el algoritmo del vecino más lejano genera rutas más cortas que el algoritmo del vecino más cercano, el cual, de media, genera rutas un 1% más largas.

Además, tal y como se muestra en la "Tabla 2" del anexo, del total de 60 rutas generadas de forma aleatoria, únicamente se obtuvieron 4, en las que la longitud de la ruta era menor con el algoritmo del vecino más cercano, con unas variaciones que, apenas, superan el 1% de la distancia de las rutas, de media.

Para las 56 rutas restantes, el resultado obtenido con el algoritmo del vecino más lejano es mejor que el del más cercano, tal y como cabía esperar, tras el análisis de la tabla anterior.

Si se representan las variaciones en forma de gráfico, se puede observar cómo, según aumenta el tamaño de la ruta, y por tanto la complejidad del problema; las variaciones entre ambos algoritmos aumentan a favor del algoritmo del vecino más lejano.

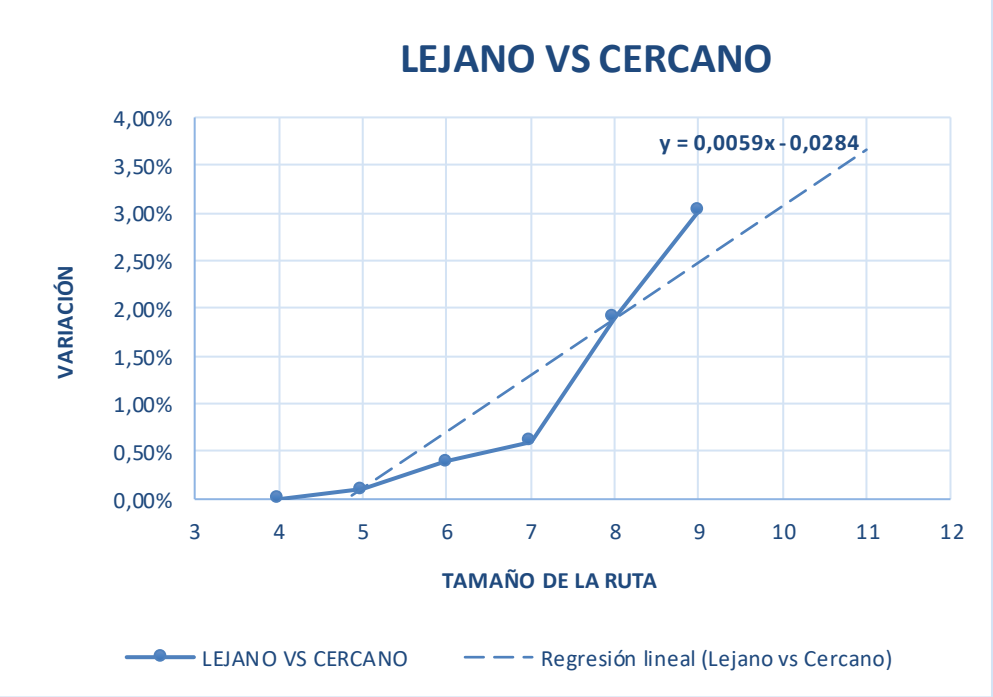

Ilustración 67. Gráfico comparativo de variaciones entre el algoritmo del vecino más lejano y del más cercano (1)

Esta gráfica muestra de forma visual, y a través de la regresión lineal, la tendencia alcista de las variaciones en la longitud de las rutas generadas, a favor del algoritmo del vecino más lejano respecto al del más cercano.

Se puede concluir afirmando que, para rutas con menos de 9 centros de salud, el algoritmo del vecino más lejano devuelve rutas más optimizadas, y por tanto más interesantes para el usuario.

5.2.2. Rutas con 9 centros de salud o más

En los casos en los que las rutas generadas tengan 9 o más centros de salud, es decir, un tamaño de ruta "n" mayor o igual que 10 (9 centros de salud y el almacén); el algoritmo genético también genera un resultado, permitiendo su comparación con el algoritmo del vecino más lejano y del más cercano.

Al igual que en el análisis anterior, se generan 10 muestras aleatorias de rutas para cada tamaño de muestra "n"; excepto para n=66, ya que es el máximo número de puntos de unión (los 65 centros de salud presentes en Zaragoza y el almacén) y, por lo tanto, los centros de salud presentes en la ruta son los mismos, por más rutas que se generen.

En este caso, se estudian los resultados para 8 tamaños de rutas "n". Comenzando con rutas que unen 10 localizaciones, e incrementando ese número de localizaciones en saltos de 8 unidades; hasta llegar a los 66 puntos que permite cómo máximo unir la aplicación.

5.2.2.1. Algoritmo del vecino más lejano vs algoritmo del vecino más cercano

Inicialmente se comparan los resultados obtenidos con los algoritmos del vecino más lejano y el del más cercano.

Tras generar 10 rutas aleatorias para cada tamaño de ruta "n", las medias en las variaciones de las distancias de ruta, obtenidas con los dos algoritmos, se muestran a continuación:

n	LEJANO VS CERCANO
10	2,46%
18	6,88%
26	6,19%
34	9,01%
42	10,71%
50	5,64%
58	8,06%
66	10,66%
PROMEDIO	**7,45%**

Tabla 9. Variaciones entre el algoritmo del vecino más lejano y el del vecino más cercano

Se puede observar que, tal y como sucedía en el análisis anteriormente realizado, los resultados obtenidos con el algoritmo del vecino más lejano son mejores que los del más cercano, el cual, de media, genera rutas un 7,45% más largas.

Además, tal y como muestra la "Tabla 3" del anexo, del total de las 71 rutas generadas aleatoriamente en este análisis, únicamente en 4 de ellas, el algoritmo del vecino más cercano obtiene rutas más cortas, es decir, en un 5,6% de los casos. Sin embargo, la variación media en estos casos es del 1,77%, un porcentaje mucho menor a la media de mejora obtenida a favor del algoritmo del vecino más lejano (7,45%).

Si se representan las variaciones en forma de gráfico, se puede observar cómo, según aumenta el tamaño de la ruta, y por tanto la complejidad del problema; las variaciones entre ambos algoritmos aumentan, a favor del algoritmo del vecino más lejano:

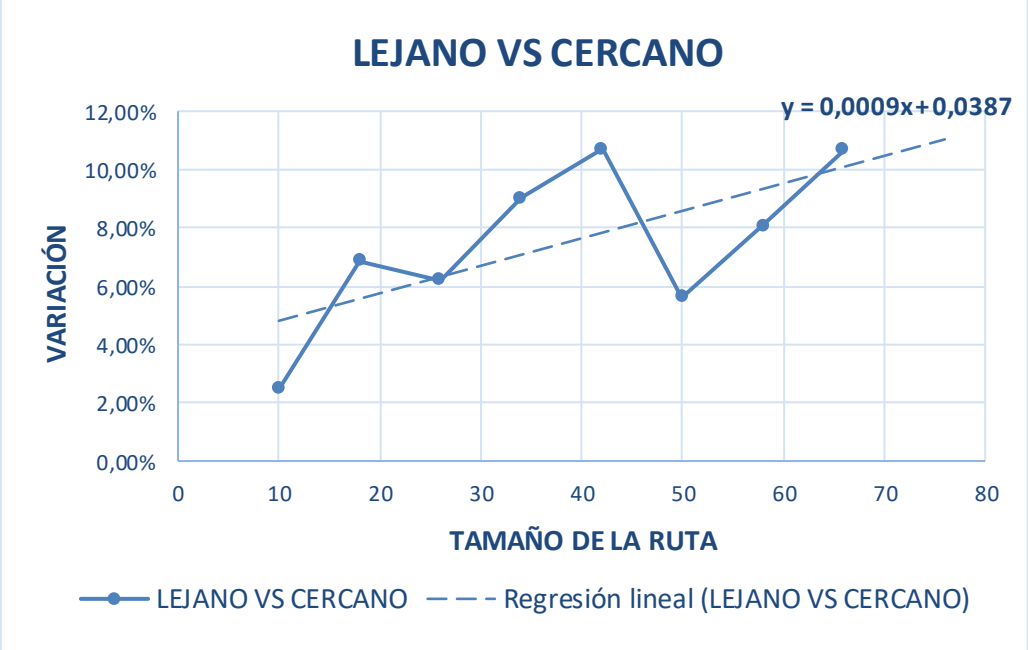

Ilustración 68. Gráfico comparativo de variaciones entre el algoritmo del vecino más lejano y del más cercano (2)

Esta gráfica muestra de forma visual, y a través de la regresión lineal, la tendencia alcista de las variaciones en la longitud de las rutas generadas, a favor del algoritmo del vecino más lejano respecto al del más cercano.

Una vez analizados los resultados obtenidos para rutas con menos de 9 centros de salud y, posteriormente, con más de 9 centros de salud; se pueden extraer ciertas conclusiones acerca del beneficio de

utilización del algoritmo del vecino más cercano, sobre el del más cercano, en la optimización de rutas.

5.2.2.2. Algoritmo genético vs algoritmo del vecino más cercano

A continuación, se realiza el análisis comparativo entre los resultados generados con el algoritmo genético y el del vecino más cercano.

Como se ha podido comprobar anteriormente, el algoritmo del vecino más cercano genera peores soluciones que el algoritmo del más lejano. En este apartado se comprueba si el algoritmo genético, es más eficaz que el del vecino más cercano, y por lo tanto, si posteriormente tiene posibilidades de mejorar los resultados del vecino más lejano.

Tras generar 10 rutas aleatorias para cada tamaño de ruta "n", las medias en las variaciones de las distancias de ruta, obtenidas con los dos algoritmos, se muestran a continuación:

n	GENÉTICO VS CERCANO
10	2,70%
18	8,14%
26	8,30%
34	11,48%
42	13,73%
50	10,67%
58	12,09%
66	14,04%
PROMEDIO	**10,14%**

Tabla 10. Variaciones entre el algoritmo genético y del vecino más cercano

Se puede observar que, los resultados obtenidos con el algoritmo genético, para cualquier tamaño de ruta, son mejores que los del más cercano, el cual, de media, genera rutas un 10,14% más largas.

Cabe destacar que, tal y como se muestra en "Tabla 3" del anexo, en ninguna ruta, del total de las 71 generadas aleatoriamente, el algoritmo del vecino más cercano ha sido mejor. Ni siquiera se ha obtenido, como mínimo, una solución igual de buena que con el algoritmo genético.

Si se representan las variaciones en forma de gráfico, se puede observar cómo, según aumenta el tamaño de la ruta, y por tanto la

complejidad del problema; las variaciones entre ambos algoritmos aumentan, a favor del algoritmo genético:

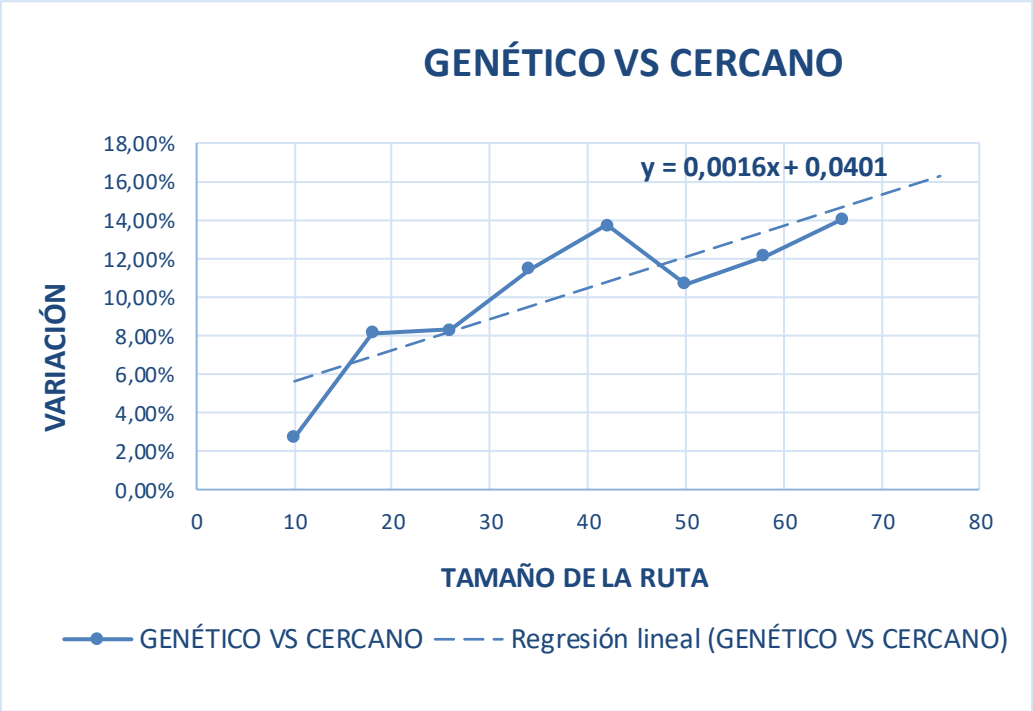

Ilustración 69. Gráfico comparativo de variaciones entre el algoritmo genético y del vecino más cercano

Esta gráfica muestra de forma visual, y a través de la regresión lineal, la tendencia alcista de las variaciones en la longitud de las rutas generadas, a favor del algoritmo genético respecto al del vecino más cercano.

Además, gracias a los análisis hasta el momento realizados, en los que se ha comprobado cómo el algoritmo del vecino más cercano genera peores rutas que el del más lejano y que el genético; se concluye valorándolo cómo el menos eficaz en la optimización de rutas.

5.2.2.3. Algoritmo genético vs algoritmo del vecino más lejano

Finalmente, se obtienen como algoritmos "finalistas", o que mejores soluciones generan, el algoritmo genético y el algoritmo del vecino más lejano.

A continuación, se realiza el análisis comparativo entre los resultados generados con ambos algoritmos.

Tras generar 10 rutas aleatorias para cada tamaño de ruta "n", las medias en las variaciones de las distancias de ruta, obtenidas con los dos algoritmos, se muestran a continuación:

n	GENÉTICO VS LEJANO
10	0,24%
18	1,28%
26	2,09%
34	2,37%
42	2,74%
50	4,79%
58	3,76%
66	3,05%
PROMEDIO	**2,54%**

Tabla 11. Variaciones entre el algoritmo genético y del vecino más lejano

Se puede observar que, los resultados obtenidos con el algoritmo genético, para cualquier tamaño de ruta, son mejores que los del más lejano, el cual, de media, genera rutas un 2,54% más largas. Realmente es un porcentaje pequeño, sin embargo, cuando se trata de rutas con una gran longitud, pueden suponer un gran ahorro de recursos, como tiempo y combustible.

Cabe destacar que, tal y como se muestra en la "Tabla 3" del anexo, en este caso, al contrario que en el análisis anterior, sí que se obtienen rutas de igual distancia con ambos algoritmos (12 de 71, lo que representa un 17%).

Sin embargo, al no obtener ninguna ruta mejor con el algoritmo del vecino más lejano, no hay indicios de que su método de búsqueda pueda ser mejor en ciertas ocasiones.

Si se representan las variaciones en forma de gráfico, se puede observar cómo, según aumenta el tamaño de la ruta, y por tanto la complejidad del problema; las variaciones entre ambos algoritmos aumentan, a favor del algoritmo genético:

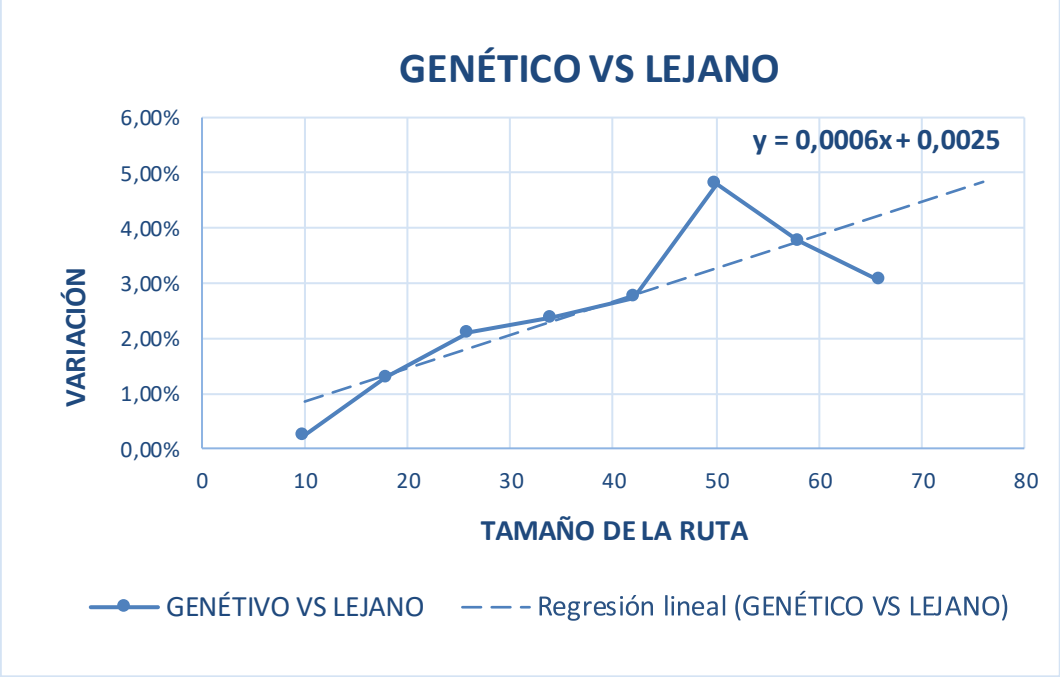

Ilustración 70. Gráfico comparativo de variaciones entre el algoritmo genético y del vecino más lejano

Esta gráfica muestra de forma visual, y a través de la regresión lineal, la tendencia alcista de las variaciones en la longitud de las rutas generadas, a favor del algoritmo genético respecto al del vecino más lejano.

Además, gracias a los análisis realizados, se puede concluir afirmando que el algoritmo genético es el más eficaz en la optimización de rutas, seguido por el del vecino más lejano, y finalmente el del vecino más cercano.

6. CONCLUSIONES

El objetivo principal de este TFG es la creación de una aplicación libre que, de forma intuitiva y sencilla, permita optimizar la ruta de reparto de material sanitario para los centros de salud de Zaragoza. Tras el desarrollo y creación de esta aplicación, se puede concluir afirmando que el objetivo se ha cumplido satisfactoriamente. Además, también se ha conseguido indagar acerca del Problema del Agente Viajero (TSP), su complejidad, algunos de los métodos heurísticos que existen actualmente para resolverlo y la programación en R.

Además, con la intención de seguir fomentando el uso de software libre, la aplicación creada está disponible en la nube, en un repositorio shintapps.io, para que cualquier interesado pueda utilizarla de forma libre. Esta aplicación permite basar la búsqueda de rutas de reparto de material sanitario para los centros de salud de Zaragoza en tres tipos de métodos heurísticos, generando un mapa interactivo con la ruta más corta obtenida.

Tal y como se ha mencionado, el TSP es un problema con alta complejidad, la cual radica en que es un problema de combinatoria, en el que, para encontrar la solución óptima, se deben de calcular todas las combinaciones de rutas posibles, dados un conjunto de localizaciones que visitar. Cuando este número es relativamente alto, el tiempo de cálculo podría dispararse a millones de años, por lo que no resulta eficiente. Es por ello que se utilizan métodos heurísticos, que brindan muy buenas soluciones, en un tiempo razonable.

Quizás, esta complejidad en la resolución del TSP y la optimización de rutas, es la que lleva a ciertas empresas a obviar o no valorar demasiado los beneficios que suponen una buena planificación de éstas. Sin embargo, está claro que es mejor obtener una buena solución, utilizando algún tipo de heurística, que ninguna.

En la aplicación creada, se utilizan tres algoritmos heurísticos: algoritmo del vecino más lejano, algoritmo del vecino más cercano y algoritmo genético. Estos algoritmos, resuelven la optimización de rutas siguiendo diferentes operaciones y cálculos, lo que genera diferentes soluciones en muchos casos. Esta variedad en las soluciones obtenidas permite la comparación de su eficacia, en lo que a optimización de rutas se refiere.

El paquete de R utilizado para resolver este problema utilizando el algoritmo del vecino más lejano y del más cercano, es uno de los más utilizados en la optimización de rutas, y muestra de ello es que recibe el nombre del propio problema que resuelve: TSP. Sin embargo, tal y como se ha podido observar en la comparación de resultados realizada,

el algoritmo genético del paquete gor, genera rutas más cortas y, por lo tanto, más óptimas e interesantes a seguir.

A continuación, no se presentan nuevos resultados, sino los resultados presentados en el apartado anterior ("5. Resultados"), de una forma resumida y conjunta, a modo de conclusión. Estos resultados representan la variación en la distancia de las rutas aleatorias generadas.

n	GENÉTICO VS LEJANO	GENÉTICO VS CERCANO	LEJANO VS CERCANO
10	0,24%	2,70%	2,46%
18	1,28%	8,14%	6,88%
26	2,09%	8,30%	6,19%
34	2,37%	11,48%	9,01%
42	2,74%	13,73%	10,71%
50	4,79%	10,67%	5,64%
58	3,76%	12,09%	8,06%
66	3,05%	14,04%	10,66%
PROMEDIO	**2,54%**	**10,14%**	**7,45%**

Tabla 12. Variaciones entre el algoritmo genético, del vecino más lejano y del vecino más cercano

Estos resultados permiten confirmar lo expuesto en el apartado anterior, y concluir afirmando que el algoritmo genético es el que mejores soluciones genera en la optimización de rutas, seguido por el algoritmo del vecino más lejano; situándose el algoritmo del vecino más cercano en tercer lugar como el más ineficaz.

Además, en el gráfico de la siguiente página, se muestran de forma conjunta las tendencias de las variaciones de la longitud de las rutas, representadas en las ilustraciones 79 y 80 del apartado "5. Resultados". Claramente, se puede observar cómo, según aumenta el tamaño de la ruta, y por tanto la complejidad del problema; las variaciones entre los algoritmos aumentan, a favor del genético.

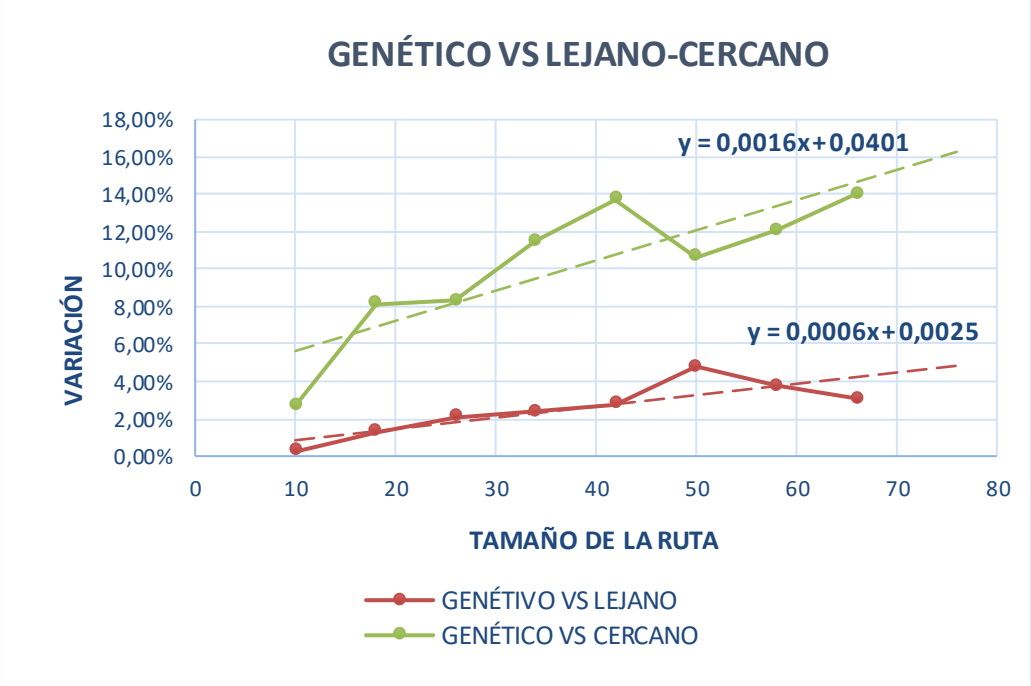

Ilustración 71. Gráfico comparativo de variaciones entre el algoritmo genético, del vecino más lejano y del más cercano

También cabe destacar que, siendo el algoritmo genético el más eficaz en la optimización de rutas, las variaciones generadas con el algoritmo del vecino más lejano son mucho menores que con el del más cercano; además de tener un menor crecimiento. Esta situación vuelve a confirmar que el algoritmo genético es el más eficaz en la optimización de rutas, seguido por el del vecino más lejano y, por último, el del más cercano.

A pesar de la mejora obtenida con los algoritmos genéticos, la optimización de rutas sigue siendo un desafío a la hora de encontrar la solución óptima, la cual permitiría a las empresas reducir todavía más los costes en su cadena de valor, haciéndolas más competitivas.

Sin embargo, dado que los algoritmos genéticos están basados en la teoría de la evolución de las especies y la selección natural de Charles Darwin, se plantea la siguiente reflexión:

Si los seres humanos han conseguido convertirse en "máquinas perfectas", en un futuro, ¿Podrán los algoritmos genéticos encontrar la solución perfecta que se lleva décadas buscando al TSP?

7. OBJETIVOS DE DESARROLLO SOSTENIBLE

Los objetivos de este Trabajo Fin de Grado están alineados con los siguientes Objetivos de Desarrollo Sostenible (ODS) y metas, de la Agenda 2030:

Objetivo 4 - Garantizar una educación inclusiva y equitativa de calidad y promover oportunidades de aprendizaje permanente para todos

1. **Meta 4.4** De aquí a 2030, aumentar considerablemente el número de jóvenes y adultos que tienen las competencias necesarias, en particular técnicas y profesionales, para acceder al empleo, el trabajo decente y el emprendimiento

Objetivo 8 - Promover el crecimiento económico sostenido, inclusivo y sostenible, el empleo pleno y productivo y el trabajo decente para todos

1. **Meta 8.2** Lograr niveles más elevados de productividad económica mediante la diversificación, la modernización tecnológica y la innovación, entre otras cosas centrándose en los sectores con gran valor añadido y un uso intensivo de la mano de obra

2. **Meta 8.4** Mejorar progresivamente, de aquí a 2030, la producción y el consumo eficientes de los recursos mundiales y procurar desvincular el crecimiento económico de la degradación del medio ambiente, conforme al Marco Decenal de Programas sobre modalidades de Consumo y Producción Sostenibles, empezando por los países desarrollados.

Objetivo 11 - Lograr que las ciudades sean más inclusivas, seguras, resilientes y sostenibles.

1. **Meta 11.6** De aquí a 2030, reducir el impacto ambiental negativo per cápita de las ciudades, incluso prestando especial atención a la calidad del aire y la gestión de los desechos municipales y de otro tipo.

Objetivo 12 - Garantizar modalidades de consumo y producción sostenibles.

1. **Meta 12.2** De aquí a 2030, lograr la gestión sostenible y el uso eficiente de los recursos naturales.
2. **Meta 12.a** Ayudar a los países en desarrollo a fortalecer su capacidad científica y tecnológica para avanzar hacia modalidades de consumo y producción más sostenibles.

Objetivo 13 - Adoptar medidas urgentes para combatir el cambio climático y sus efectos.

1. **Meta 13.2** Incorporar medidas relativas al cambio climático en las políticas, estrategias y planes nacionales

8. BIBLIOGRAFÍA

576c2a3064481.pdf. (s. f.). Recuperado 8 de marzo de 2023, de http://posgrado.lapaz.tecnm.mx/uploads/archivos/576c2a3064481.pdf

Algoritmos Genéticos—Fernando Sancho Caparrini. (s. f.). Recuperado 8 de marzo de 2023, de http://www.cs.us.es/~fsancho/?e=65

ARAGON, G. D. (2023). *Visor 2D de IDEAragon: Infraestructura de Datos Espaciales de Aragon* [Document]. https://idearagon.aragon.es/visor/index.html?ACTIVELAYER=v206_zonas_salud&QUERY=codigo=VISIBLELAYERS=v206_zonas_salud

Asensio, C. (2022). *El problema del viajante*.

Capitulo+8.pdf. (s. f.). Recuperado 8 de marzo de 2023, de https://biblus.us.es/bibing/proyectos/abreproy/70238/fichero/Capitulo+8.pdf

Cunquero, R. M. (s. f.). *Algoritmos Heurísticos en Optimización Combinatoria*.

Duran, M. I. (s. f.). *EL PROBLEMA DEL VIAJANTE (TSP)*.

Fretes, F. (2018). *EVOLUCIÓN DEL HOMBRE | Concepto y sus etapas*. https://historiando.org/evolucion-del-hombre/

La importancia de la optimización de rutas—Entrevista Alfredo García Hernández-Díaz. (2017, marzo 20). JULIÁN SASTRE. https://juliansastre.com/la-importancia-de-la-optimizacion-de-rutas/

La importancia de optimizar las rutas de transporte. (s. f.). Recuperado 8 de marzo de 2023, de https://www.transeop.com/blog/La-importancia-de-optimizar-las-rutas-de-transporte/130/#optimizacion-rutas-transporte-beneficios

López, B. S. (2019, junio 12). Problema del agente viajero—TSP » Ingenieria Industrial Online. *Ingenieria Industrial Online*. https://www.ingenieriaindustrialonline.com/investigacion-de-operaciones/problema-del-agente-viajero-tsp/

Memoria_OptimizacionRutasTransporte.pdf. (s. f.). Recuperado 8 de marzo de 2023, de https://eprints.ucm.es/id/eprint/23027/1/Memoria_OptimizacionRutasTransporte.pdf

Morante, S. (2021, noviembre 12). *8 Ventajas de la Optimización de Rutas en Logística*. Tookane. https://tookane.com/ventajas-optimizacion-rutas-logistica/

Object, object. (s. f.). *Resolución del problema del viajante de comercio (TSP) y su variante con ventanas de tiempo (TSPTW) usando*

métodos heurísticos de búsqueda local. Recuperado 8 de marzo de 2023, de https://core.ac.uk/reader/211096990

Optimización de rutas de transporte. (2020, julio 7). *NovaTrans®*. https://www.novatrans.es/blog/optimizacion-rutas-transporte/

P (clase de complejidad) P Y NPHARD - Wikipedia, la enciclopedia libre. (s. f.). Recuperado 8 de marzo de 2023, de https://es.wikipedia.org/wiki/P_(clase_de_complejidad)_P_Y_NPHARD

Proyecto_762.pdf. (s. f.). Recuperado 8 de marzo de 2023, de http://eio.usc.es/pub/mte/descargas/proyectosfinmaster/proyecto_762.pdf

¿Qué son los algoritmos heurísticos? (s. f.). Quora. Recuperado 8 de marzo de 2023, de https://es.quora.com/Qué-son-los-algoritmos-heurísticos

R: The R Project for Statistical Computing. (s. f.). Recuperado 8 de marzo de 2023, de https://www.r-project.org/

[Resuelta] algorithm | ¿Cuál es la diferencia entre un. (s. f.). Recuperado 8 de marzo de 2023, de https://www.iteramos.com/pregunta/21531/cual-es-la-diferencia-entre-un-heuristico-y-un-algoritmo

RStudio. (2023). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=RStudio&oldid=149496820

Ruiz, E. M. (2021, diciembre 17). La importancia de la optimización de rutas. *Programa Transporte*. https://programadetransporte.es/la-importancia-de-la-optimizacion-de-rutas/

Shiny. (s. f.). Recuperado 8 de marzo de 2023, de https://shiny.rstudio.com/

Spain, S. (2020, octubre 26). Optimizar Rutas de Reparto para mejorar los resultados. *GRUPO T.I.* https://transporte-inmediato.com/optimizar-rutas-de-reparto/

TFG-I-467.pdf. (s. f.). Recuperado 8 de marzo de 2023, de https://uvadoc.uva.es/bitstream/handle/10324/18552/TFG-I-467.pdf?sequence=1&isAllowed=y

Universitat Politècnica de València - UPV (Director). (2013, febrero 22). *S6.12- El problema del viajante (peso bajo) | | UPV*. https://www.youtube.com/watch?v=MplYrY3PXSQ

Usar heurísticas (artículo) | Algoritmos | Khan Academy. (s. f.). Recuperado 8 de marzo de 2023, de https://es.khanacademy.org/_render

Relación de documentos

(X) Memoria 82 páginas

(_) Anexos 22 páginas

La Almunia, a 06 de junio de 2023

Firmado: Iván Sevilla Antón