Universidad Internacional de la Rioja (UNIR)

Escuela Superior de Ingeniería y Tecnología

Máster Ingeniería Matemática y Computación

Modelos para planificación de rutas de transporte

Trabajo Fin de Máster

Presentado por: Margalida Salom Ramis

Dirigido por: Marc Jorbá Cuscó

Ciudad: Palma de Mallorca

Fecha: 20 de Julio de 2022

Índice de Contenidos

Resumen v

Abstract vi

1. Introducción 1

1.1. Justificación 1

1.2. Planteamiento del trabajo 2

1.3. Estructura 2

2. Contexto y Estado del Arte 4

2.1. Contexto 4

2.2. Estado del arte 8

3. Objetivos concretos y metodología de trabajo 13

3.1. Objetivo general 13

3.2. Objetivos específicos 13

3.3. Metodología del trabajo 14

4. Desarrollo del trabajo 16

4.1. Herramientas de software 16

4.2. Captura y procesado de los datos 17

4.3. Generación de valores aleatorios 21

4.4. Planteamiento del problema 22

4.5. Diseño de algoritmos 24

4.5.1. Algoritmo de programación dinámica 25

4.5.2. Modificación del algoritmo de Prim 27

4.5.3. Algoritmo genético 29

4.6. Comparativa de resultados 31
4.6.1. Comparación gráfica 32
4.6.2. Eficiencia computacional 35
4.6.3. Resultados numéricos 41
4.6.4. Discusión sobre los resultados 45

5. Conclusiones y Trabajo Futuro **47**
5.1. Conclusiones 47
5.2. Líneas de trabajo futuro 48

Referencias **50**

Índice de Ilustraciones

4.1. Ejemplo de ruta 19
4.2. Datos de los núcleos de población de Mallorca 20
4.3. Núcleos de población de Mallorca con latitud y longitud 21
4.4. Mapas de carreteras representados como grafos 23
4.5. Algoritmo de programación dinámica 25
4.6. Árbol de recubrimiento mínimo vs ruta deseada 28
4.7. Principio de la ruleta 30
4.8. Árbol genético 31
4.9. Puntos elegidos para el estudio 32
4.10. Resultado del algoritmo de programación dinámica 33
4.11. Resultado del algoritmo modificado de Prim 34
4.12. Resultado del algoritmo genético 35
4.13. Coste algoritmo de fuerza bruta 36
4.14. Coste modificación algoritmo de Prim 37
4.15. Coste del algoritmo genético con 500 iteraciones 38
4.16. Coste del algoritmo genético según el número de iteraciones 38
4.17. Resultados del algoritmo genético según el número de iteraciones 39
4.18. Coste y resultados del algoritmo genético según el número de iteraciones . . 40
4.19. Distancia y tiempo obtenidos según el algoritmo modificado de Prim 42
4.20. Resultado del algoritmo de Prim con 10 núcleos de población 44

Índice de Tablas

4.1. Resultados obtenidos según cada algoritmo y tiempo de ejecución considerando 11 puntos a visitar. 32
4.2. Resultados obtenidos al visitar 6 núcleos de población 42
4.3. Resultados obtenidos al visitar 10 núcleos de población 43
4.4. Resultados obtenidos al visitar 16 núcleos de población 44

Resumen

El objetivo de este estudio es diseñar y aplicar diferentes modelos para calcular un recorrido eficiente dados ciertos puntos a visitar. Para ello se crearán ciertos puntos aleatorios sobre un mapa y se evaluarán los posibles factores a tener en cuenta para la creación de rutas. Los modelos diseñados tendrán como base diferentes orígenes, los cuáles serán evaluados para conocer las ventajas e inconvenientes de cada uno de ellos. Los resultados obtenidos sugieren que un método de fuerza bruta es el más adecuado para pequeños recorridos mientras que para medianos y grandes visitar el punto más próximo es el más recomendado.

Palabras Clave: problema del viajante, camino más corto, teoría de grafos, ruta.

Abstract

The aim of this work is to design and apply different models to calculate an efficient route given certain points to visit. To do this, random points will be pointed on a map and the possible factors to consider in the creation of routes will be evaluated. The models designed will be based on different origins, which will be evaluated to know the advantages and disadvantages of each of them. The results obtained suggests that a brute force method is the most suitable for small routes, while for medium and large ones visiting the nearest point is the most recommended.

Palabras Clave: travelling salesman problem, shortest path, graph theory, route.

Capítulo 1

Introducción

En este primer capítulo se realizará una introducción al tema planteado y las ideas que han motivado a la realización de este estudio.

1.1. Justificación

En la actualidad los consumidores estamos constantemente comprando productos a través de internet o alguna aplicación móvil, ya sea para solicitar un producto que no encontramos en una tienda física, para comprar un objeto a un precio más competitivo o por simple comodidad. La evolución de las nuevas tecnologías ha ofrecido estas facilidades al consumidor que, en ciertos casos, también ha ocasionado ciertos problemas y surgimiento de nuevas necesidades a las empresas.

Uno de los nuevos retos por parte de los empresarios es proporcionar satisfacción a los clientes manteniendo el control de sus gastos. Entre ellos, uno de los presupuestos que se ha tenido un incremento es el referente a los costes de transporte. Estos gastos han crecido recientemente por el incremento del precio del combustible, pero también por el incremento de los desplazamientos de los últimos años. Esta nueva situación ha proporcionado un auge en las empresas de distribución que, al igual que antes, tienen la necesidad de satisfacer a los clientes al menor coste posible, entre los que se incluyen los gastos de transporte.

Como se ha visto en la asignatura de Optimización y Gestión de la Producción del Máster en Ingeniería Matemática y Computación, el transporte es sólo una de las fases de una cadena de suministro, siendo todas ellas de vital importancia para la satisfacción del cliente final. Entre las diferentes etapas, se ha prestado especial atención en las referentes a la distribución y el transporte, ya que no depende tanto de razones comerciales sino que

tiene un enfoque más técnico y es posible su optimización mediante diferentes técnicas. Un problema que se analizó en detalle en la asignatura citada fue el de asignación de flotas, conocido como *Fleet assignment problem* o *FAP*. De forma muy resumida, este problema consiste en determinar los vehículos y características de los mismos para satisfacer cada uno de los diferentes trayectos que va a realizar un transportista.

Sin embargo, un caso de estudio que se dejó abierto fue el de planificación de rutas de transporte, conocido como *Vehicle routing problem* o *VRP*, motivo por el cuál se decide profundizar en este estudio. El problema planteado consiste en calcular la ruta más rentable para la compañía maximizando la satisfacción del cliente final.

1.2. Planteamiento del trabajo

Para afrontar el problema de planificación de rutas de transporte vamos a suponer que queremos organizar la mejor ruta con un vehículo que dispone de capacidad ilimitada. Por lo que el objetivo será, dados ciertos puntos que debe visitar un transportista a lo largo de su jornada laboral, definir el orden en el que se recomienda realizar el recorrido. Es obligatorio que al finalizar la jornada todos los puntos se hayan visitado.

El problema se plantea como la labor que realizaría un repartidor de paquetería o mensajería, aunque el problema sería extrapolable a otros sectores en los que haya una dualidad con esta profesión, independientemente del contenido del servicio que se realice.

El transportista va a tener siempre como primer y último punto de su recorrido el almacén en el que se guarden los paquetes. Este caso particular al que hacemos frente se conoce como problema del viajante, conocido también como TSP por sus siglas en inglés (*Traveling salesman problem*).

El problema propuesto ha sido de gran investigación durante los últimos años, siendo en muchos casos los algoritmos usados para su resolución privados. Por esta razón, se pretenden diseñar diferentes modelos abiertos que solucionen el problema del viajante, realizar una comparación entre ellos para conocer sus ventajas e inconvenientes y definir cuál sería mejor usar según diferentes escenarios.

1.3. Estructura

Para realizar el estudio, en el Capítulo 2 se resumen las principales fuentes consultadas para la contextualización del caso. En él se presenta cómo surgió el problema de creación

de rutas, su contexto y algunos de los estudios más relevantes del sector. Se introducen también algunos conceptos básicos y ciertas técnicas de resolución de este caso en concreto que se usarán y a las cuáles se hará hará referencia durante el transcurso de la memoria.

En el Capítulo 3 se expone el objetivo general planteado para el estudio y los objetivos específicos necesarios para conseguirlo. En dicha sección se incluye también un apartado sobre la metodología llevada a cabo para lograr los objetivos planteados.

En el Capítulo 4 se encuentra el desarrollo del trabajo. Inicialmente se plantean las herramientas de software usadas y las razones. Posteriormente se capturan y procesan los datos necesarios para realizar el estudio. Una vez se dispone del contexto necesario, se va a plantear el problema a resolver y diferentes modelos para su resolución. Finalmente se aplicarán dichos métodos a ciertos conjuntos de datos y se hará una comparativa de los diferentes resultados obtenidos, comentando las ventajas e inconvenientes de cada modelo.

Para finalizar, en el Capítulo 5 se resumen las principales líneas de trabajo llevadas a cabo, las conclusiones obtenidas durante el estudio y posibles líneas de investigación futuras relacionadas con el sector de la distribución.

Al final del documento se puede encontrar la bibliografía a la que se hace referencia durante la memoria, compuesta por las fuentes que han sido de mayor utilidad para el desarrollo del estudio.

Capítulo 2

Contexto y Estado del Arte

En este capítulo se hace un resumen de cómo surgió el problema de la creación de rutas y la necesidad de su optimización. Se mostrarán también los conceptos básicos y algunas de las técnicas que se aplican para la resolución del problema, además de algunos de los estudios más relevantes del sector.

2.1. Contexto

Una cadena de suministros comprende un gran número de funciones que sólo la correcta cohesión y comunicación entre ellas tendrá como resultado una situación exitosa (Asl-Najafi, Yaghoubi, y Noori, 2021). A continuación se resumen brevemente cada una de las fases de cualquier cadena de suministros, resaltando su importancia.

1. Planificación: esta es una etapa de decisión referente a la estrategia de negocio y operaciones que se van a seguir. En ella se evalúan aspectos como si fabricar un producto u otro, la elección de los proveedores con los que se desea trabajar o el diseño del producto que se desea entregar al cliente. En este paso se evaluarán también aspectos como los costes de fabricación o la externalización del producto o servicio deseado.

2. Fuente: es la fase encargada de la adquisición de las materias primas que se usarán para fabricar el producto deseado. Se incluyen aspectos como las negociaciones con los proveedores, reducción de costes de los mismos o calendario sobre las fechas de entrega.

3. Fabricación: es la etapa encargada de la producción del bien o servicio. En ella se

incluye la programación de la producción que se va a realizar, el embalaje que se va a usar para el producto final (si procede) o las pruebas a realizar sobre el mismo. En esta fase también debe adaptarse el entorno a la normativa existente, tanto en referencia a la protección de los empleados como del producto en sí según el mercado en el que se desee distribuir.

4. Entrega: comprende la solicitud del bien o servicio por parte del cliente, las estrategias de distribución y la entrega del mismo. El inventario, la cantidad y el tipo de vehículos o la gestión del almacén entrarían también en esta fase.

5. Devolución: es la fase encargada de gestionar cualquier tipo de devolución y las gestiones que esto conlleva. Se incluyen tanto las devoluciones por decisión propia del cliente o por producto defectuoso o en mal estado. Algunas de las gestiones en esta etapa pueden ser la sustitución del producto por otro similar, el reembolso o la gestión de la política de devoluciones,

Cabe resaltar que cada una de las fases es imprescindible, ya que en caso de no desarrollarse correctamente impedirían que el producto llegue al cliente final, abarcando al fracaso de la cadena de distribución e, indirectamente, de la empresa. Cada fase tiene un gran estudio por detrás, pudiendo ser objeto de diferentes investigaciones. En este caso nos centraremos en la fase de entrega del producto al cliente.

Años atrás cuando un usuario quería adquirir un producto se desplazaba a un centro comercial o tienda para poder hacerse con él. Esto implicaba que los distribuidores centraran su planificación del transporte entre su centro de producción o almacenamiento y los puntos de venta al cliente. De cada vez son más los usuarios que compran productos de forma telemática (Véliz, 2022), provocando un incremento en los desplazamientos por parte de los transportistas. Actualmente las rutas se centran, en muchos casos, en la distribución de un producto desde el centro de origen hasta el cliente final, incrementando así los transportes al visitar todos los clientes finales que lo deseen. Inevitablemente, esto ha provocando también un auge en el presupuesto destinado a la distribución o la aparición de nuevas empresas de reparto, lo que conlleva una necesidad de redes de transporte que puedan ofrecer mejor nivel de servicio por a la alta demanda.

En este sentido, la planificación de rutas de transporte se ha vuelto un punto crítico para las empresas ya que los costes de este servicio se han visto incrementados en personal contratado, vehículos necesarios u otros gastos derivados de estos, entre otros. Este tipo

de problemas podemos encontrarlos en la literatura como *Vehicle Routing Problems* o, más comúnmente, como VRP por sus siglas en inglés. En el caso de la distribución de productos físicos, se basa en iniciar y finalizar un recorrido en los almacenes o depósitos de cierta compañía y atender la demanda de múltiples clientes, dispersos en diferentes localizaciones (Puchades Cortés, Mula Bru, y Rodríguez Villalobos, 2008).

Otro de los problemas que ha surgido, aunque en menor medida, es el de programación de vehículos, conocido también como *Vehicle Scheduling Problems* o VSP (Frâsinaru y Olariu, 2021). Este problema surge a partir de la alta demanda y se basa en maximizar la cantidad de clientes que son atendidos cuando ha surgido algún imprevisto, ya sea una avería del vehículo, indisposición del trabajador o un retraso importante por razones externas.

El objetivo de la optimización de rutas de transporte se convierte así en reducir el coste del servicio maximizando la satisfacción de los clientes. Esto incluye algunos aspectos como cumplir al máximo con las fechas de entrega, la reducción de los costes de personal y vehículos o la eficiencia en el desplazamiento realizado. Luego, para organizar las rutas será necesario conocer factores como: la jornada laboral de los trabajadores y los tipos de permisos de conducir disponibles; los tipos de vehículos necesarios para los desplazamientos, que pueden requerir conservaciones especiales como neveras o congeladores; el mantenimientos de los vehículos; el tipo de cliente con el que se está trabajando, la frecuencia en sus pedidos o la disponibilidad horaria del mismo. Como se puede observar, el hecho de organizar las rutas de transporte tiene en cuenta tanto optimización de servicios como consideración de restricciones.

Según el contexto de cada empresa de transporte, los objetivos serán diferentes. Para ello es fundamental conocer el tipo de empresa frente a la que estamos y las prioridades de la misma. Por ejemplo, en el caso de estar en el sector de la distribución de comida recién hecha entre sus prioridades principales estará el tiempo transcurrido entre la elaboración del producto y la entrega al cliente. Por otro lado, una empresas de reparto de paquetes probablemente tendrá entre sus prioridades factores como reducir el coste del servicio frente a llegar unos minutos antes o más tarde.

Además de saber a qué tipo de empresa se va a servir, priorizando así unos aspectos u otros, existen una gran cantidad de factores a tener en cuenta. Entre ellos, algunos de los más generalizados según el tipo de empresa son la ubicación del centro de origen o almacén, los medios de transporte a utilizar, el contexto en el que se efectúe el desplazamiento o la

disponibilidad horaria del cliente final, entre otros.

La ubicación del almacén será de vital importancia cuando se requiera la carga y/o descarga de mercancías durante la jornada en numerosas ocasiones. Si se va a cargar el vehículo únicamente una vez al día, este aspecto tendrá menos peso que si se tiene que realizar dicha operación 3 veces al día. Por otra parte, tampoco sería del todo eficiente distribuir productos turísticos y posicionar el centro de origen en un sitio lejano a los clientes, independientemente que se tenga o no que recargar el vehículo.

Los medios de transporte son también otro de los factores a tener en cuenta según el tipo de producto que se distribuya. Si estamos frente a un repartidor de paquetería de tamaño reducido, por ejemplo cartas, probablemente sea suficiente con una furgoneta o coche pequeño. Si transportamos electrodomésticos de gran tamaño, sería más adecuado un camión o similar frente a una furgoneta. Por contra, si el repartidor es un proveedor de helados deberá disponer de una furgoneta con los respectivos congeladores para asegurar la cadena de frío.

El contexto en el que se efectúe el desplazamiento es otro de los puntos a tener en cuenta en la creación de rutas. En caso de necesitar entrar en una gran ciudad, hacerlo en hora punta puede suponer que conlleve mucho más tiempo al haber posibilidad de retenciones, mientras que si se realiza dicho reparto durante un fin de semana se va a tardar el tiempo exclusivamente necesario para el desplazamiento, permitiendo así servir a más clientes.

La disponibilidad horaria del cliente es un aspecto fundamental para no tener que visitarlo nuevamente o, inclusive, que se termine rechazando el producto. Si el cliente indica que sólo estará en casa por las tardes, no tiene sentido programar ese reparto en la franja horaria de mañana ya que muy probablemente no esté en el domicilio y se tenga que asistir otra día en el horario que el cliente esté disponible.

Además de los factores externos al recorrido en sí, algunos de los cuáles han sido citados en los párrafos anteriores, también se deben tener en cuenta otras condiciones de interés para el empresario o repartidor. Entre los elementos a considerar para crear un recorrido es habitual considerar el tiempo total del recorrido, la cantidad de quilómetros que se van a realizar o el coste del desplazamiento realizado. A veces no se trata únicamente de hacer la ruta más corta o más barata, sino que puede haber múltiples fines para la creación del recorrido considerado óptimo, dando lugar a los problemas de objetivos acumulados (Cum-VRP) (Corona-Gutiérrez, Nucamendi-Guillén, y Lalla-Ruiz, 2022). En ellos no se

centra en mejorar una cosa u otra, sino que se desea encontrar un equilibrio entre diferentes objetivos para que la ruta sea eficiente.

2.2. Estado del arte

La necesidad de invertir en rutas de transporte surge cuando las empresas sirven a sus clientes con una demanda superior a la capacidad que tienen de dar servicio, ya sea por restricciones en la flota o por el personal contratado. A grandes rasgos, existen dos categorías en las que se pueden clasificar las soluciones a los problemas de transporte: los métodos óptimos y los métodos heurísticos (Ruiz y Landín, 2003).

Los métodos óptimos, tal y como su nombre indica, persiguen obtener los mejores valores según una función objetivo fijada previamente y que a la vez satisfaga todas las restricciones presentes en el problema. El inconveniente de estos métodos es que la obtención de esa solución deseada conlleva un elevado coste computacional, ya sea por el elevado tiempo de ejecución del algoritmo o por la inexistencia de la misma.

Por otra parte, los métodos heurísticos buscan valores aceptables para la función objetivo y que a la vez satisfagan todas las restricciones. Sin embargo, las soluciones obtenidas habitualmente no son las mejores. Estos métodos carecen de una base matemática y se basan mayoritariamente en la intuición. La ventaja es que permiten obtener rápidamente una solución inicial que permite aproximarse a una solución óptima. Realizando una comparativa con los métodos óptimos, son más eficientes computacionalmente y de gran utilidad cuando la solución óptima es demasiado compleja. Sin embargo, es frecuente que tiendan a soluciones locales frente a la solución óptima del problema.

Los primeros estudios del sector se centran en el desarrollo usando métodos heurísticos (Haimovich y Rinnooy Kan, 1985). Inicialmente se fijaban unos clientes, representados como puntos en el plano y una capacidad determinada del vehículo. Dadas estas condiciones, se consideraba la distancia Euclídea entre los diferentes clientes, realizando pequeñas variaciones y generalizaciones sobre los resultados obtenidos. Como ya se intuye, en la actualidad esta idea presenta carencias ya que habitualmente la distancia entre dos puntos no se corresponde a la distancia euclídea, sino que se deben respetar las órdenes de circulación.

Adicionalmente a los métodos heurísticos, existen también otras formas de entender las rutas de distribución. Una de ellas es como parte de la Teoría de Grafos (Puchades Cortés

y cols., 2008). El inicio de La Teoría de Grafos se establece con el trabajo de Leonhard Euler sobre los Siete Puentes de Königsberg (Euler, 1736). Königsberg fue, hasta el año 1945, una ciudad de Prusia del Este atravesada por el río Pregolia, el cuál fluye a través de la ciudad dividiéndola en cuatro zonas diferentes, conectadas por un total de siete puentes. Euler, uno de los matemáticos de la época, proporcionó una solución para que los habitantes pudieran recorrer todos los puentes de la ciudad sin pasar pasar dos veces por el mismo puente. Además de la solución concreta, proporcionó un método general para poder resolver otros problemas del mismo tipo.

La Teoría de Grafos se aplica a día de hoy en una gran variedad de ámbitos, especialmente para estudiar problemas de flujo de red (Taha, 2004). Entre ellos se destacan algunos de los basados en la programación lineal y que usan implícitamente la Teoría de Grafos:

- Problema de transporte: consiste en minimizar el coste total de enviar ciertos bienes desde uno o más puntos de origen hasta uno o más puntos de destino, satisfaciendo las restricciones de la oferta y la demanda. De este problema pueden surgir nuevas variantes como bloquear ciertas rutas o carreteras, limitar la capacidad de los vehículos o cambiar la función objetivo. A su vez, es posible aplicarlo en otras áreas para llevar a cabo otras funciones como distribución del personal entre las diferentes áreas de una empresa, la organización de los turnos de los empleados, o el control de inventario, entre otros.

- Problema de asignación: es similar al problema de transporte pero se centra en asignar a cada tarea el mejor recurso. En general su resolución es más eficiente frente a los problemas de transporte, ya que la relación suele ser uno a uno.

- Problema de transbordo: se basa en optimizar la carga, por ejemplo de los vehículos, para realizar envíos desde uno o múltiples orígenes hasta uno o múltiples destinos, de tal manera que cada cliente vea satisfecha su demanda al mínimo coste de transporte. En el transcurso de este trayecto se puede pasar por puntos intermedios, en los cuáles se puede cargar o descargar mercancía según se crea oportuno.

- Problema del viajante: tiene como objetivo crear un recorrido de distancia mínima de tal forma que se pase una única vez por cada ciudad o punto a visitar y se regrese de nuevo al punto de origen. Éste es un caso particular del problema de transporte.

Por otra parte, hay una gran diversidad de problemas de optimización de redes, ya que pueden darse muchas situaciones diferentes y áreas de orígenes diversos, como las redes de transporte, las eléctricas o de comunicaciones, entre otras. En este sentido se puede establecer una clasificación de los problemas basados en la optimización de redes (Hillier y Lieberman, 2013).

- Problema de la ruta más corta: tiene como objetivo encontrar la trayectoria que recorra la mínima distancia total desde un origen a un destino fijados, pasando por ciertos puntos intermedios. Para ello desde el punto de origen se identifica de manera sucesiva la ruta más corta a cada uno de los puntos restantes, hasta llegar al punto de destino con el mínimo recorrido posible. Un ejemplo para este caso sería un repartidor, el cuál desea visitar ciertos puntos y volver al origen.

- Problema del árbol de expansión mínima: guarda cierta similitud con el problema anterior ya que su objetivo es también seleccionar una sucesión de puntos con el recorrido total más corto posible. En este caso en concreto, se desea diseñar una red de conexiones para que cada par de puntos esté unidos, ya sea directa o indirectamente, buscando siempre la unión de menor distancia. Un ejemplo para este caso sería una línea de metro o autobús, que va desde un punto a otro sin necesidad de hacer una ruta circular.

- Problema de flujo máximo: tiene como objetivo principal encontrar la ruta que maximice la satisfacción de los clientes. Para ello se desea maximizar el flujo desde un origen al destino, pasando por ciertos puntos intermedios. Este flujo puede ser el número de viajes de una línea de autobús, minimizar la cantidad de las paradas maximizando los pasajeros, o maximizar el número de repartos de un transportista recorriendo la menor distancia posible, entre otros.

- Problema del flujo de coste mínimo: es un problema similar al anterior pero en este caso se persigue el objetivo de minimizar el coste total, teniendo presente que se deben satisfacer ciertas restricciones de flujo. Un ejemplo de este tipo de problema sería una red de producción-distribución, en la cuál se desea satisfacer la demanda de todos los clientes evitando costes como el derivado del almacenamiento de los productos.

A partir de las clasificaciones planteadas, es evidente pensar que además de un proble-

ma de teoría de grafos, el problema de las rutas de transporte puede plantearse también como un problema de optimización (Toyoda, Okamoto, y Koakutsu, 2017). Siguiendo esta idea, se dispone de una función que se desea optimizar y ciertas ecuaciones o inecuaciones planteadas como restricciones del problema. El algoritmo de optimización mínima secuencial, conocido como SMO por sus siglas en ingles (*Sequential minimal optimization*), permite solucionar este tipo de problemas. Inicialmente se eligen dos puntos que serán modificados iterativamente si se encuentra otro par que minimice el coste de la función objetivo.

Plantear el caso como un problema de optimización ofrece a su vez la opción de añadir restricciones, como por ejemplo según diferentes franjas de tiempo (Ma, Wu, y Dai, 2017). La entrega de ciertos productos conlleva en sí mismo una disponibilidad horaria deseada, llegando a ser en algunos casos un requisito para la compra del bien o servicio. Por ejemplo, un repartidor de periódicos no puede realizar la entrega a última hora de la tarde, ya que en este caso el comprador no va a estar interesado en adquirir este producto. De forma análoga, un restaurante que abra sólo para cenas puede no estar disponible para recibir la mercancía con la que servirá a sus clientes a primera hora de la mañana. Por esta razón, en una creación de rutas eficiente deberá también tenerse en cuenta la disponibilidad del cliente final. Un posible método para solucionar este problema es usando la búsqueda de vecinos próximos (Redi, Maghfiroh, y Yu, 2014). Este algoritmo se basa en ordenar a los clientes según el inicio de su ventana de tiempo disponible y, para cada cliente que sea añadido, verificar que el vehículo es capaz de llegar cumpliendo esa restricción temporal. Este tipo de problemas pueden no tener solución al tener la necesidad de visitar a dos clientes muy alejados en distancia en una corta franja de tiempo.

Como extensión del problema clásico sobre la optimización de las capacidades en problemas de rutas, surge el problema sobre objetivos acumulados (Cum-VRP por sus siglas en inglés) (Corona-Gutiérrez y cols., 2022). Este tipo de problemas consideran como múltiples funciones objetivo del problema, como el coste acumulado y la reducción del tiempo de satisfacción a los clientes. Este caso ha sido también de gran interés por parte de los investigadores ya que habitualmente se desea una combinación entre diferentes objetivos. Por ejemplo, en ocasiones no se desea una rápida atención al cliente si eso desencadena un elevado coste, aunque tampoco un mal servicio a cambio de un coste bajo, sino que se persigue un equilibrio entre diferentes objetivos.

Un hecho sorprendente a partir de la bibliografía consultada es la carencia de méto-

dos basados en redes neuronales para la solución de problemas de planificación de rutas (Mandziuk, 2019). Éstos modelos en la actualidad son aplicados a una gran variedad de sectores, ofreciendo los mejores resultados en muchos casos. Desde el año 1985 hay diferentes aplicaciones a problemas de optimización satisfaciendo ciertas restricciones, pero finalmente se han dejado de usar en la creación de rutas de transporte por el hecho de proveer soluciones poco útiles frente a otros métodos más eficientes.

Frente al problema de planificación de rutas de transporte, siempre nos centramos en cuál es el mejor caso que satisface los objetivos deseados. Pero, existen también otros estudios en los que se comparan diferentes algoritmos centrándose en los peores casos (Bertazzi, Golden, y Wang, 2015). Eso es equivalente a plantear la función objetivo como minimización de la ruta más larga o minimización del coste máximo.

Otra forma de plantear el problema del transporte es usando algoritmos genéticos (Hardi, 2015). Estos algoritmos surgieron como una de las líneas de la inteligencia artificial alrededor del año 1970 (Holland, 1975). La idea general de este tipo de algoritmos es estudiar como van evolucionando los individuos cuando son sometidos a ciertas acciones aleatorias que podrían darse en caso de cualquier mutación o selección. En este caso en particular, se trataría de ver cómo van evolucionando la distancia o el tiempo total de una ruta si realizamos ciertas alteraciones sobre una solución inicial.

Capítulo 3

Objetivos concretos y metodología de trabajo

En este capítulo se van a plantear los objetivos generales y específicos del trabajo, así como la metodología llevada a cabo para alcanzarlos.

3.1. Objetivo general

El objetivo principal del estudio es diseñar modelos abiertos que permitan solucionar el problema del viajante usando diferentes técnicas.

3.2. Objetivos específicos

Con la finalidad de conseguir el objetivo principal del trabajo se han planteado una serie de objetivos específicos que deberán conseguirse a lo largo del estudio. Estos objetivos específicos son:

1. Capturar los datos sobre los posibles puntos en los que se realizará un servicio.
2. Obtener ciertos puntos aleatorios sobre un mapa.
3. Identificar y analizar diferentes factores que pueden influir en la creación de rutas.
4. Diseñar diferentes algoritmos de planificación de rutas en función de ciertas variables.
5. Evaluar los resultados obtenidos para comparar los diferentes algoritmos diseñados.

3.3. Metodología del trabajo

En este apartado se describen los pasos llevados a cabo para desarrollar el estudio.

- **Familiarización del contexto y búsqueda de estudios similares.** El objetivo de este estudio es ver cuáles son los estudios existentes del sector para analizar posibles mejoras o diseñar nuevos modelos que proporcionen novedades sobre los ya existentes.

- **Análisis de los principales factores a tener en cuenta en la creación de rutas.** Para optimizar correctamente una ruta de transporte primero tienen que fijarse los objetivos de la misma, motivo por el cuál primero se va a analizar cuáles son los factores a tener en cuenta y las diferentes prioridades que pueden existir en la creación del trayecto.

- **Captura de los datos.** Dado que por protección de datos no se ofrece públicamente información sobre el domicilio de las personas que realizan más compras de servicios de forma telemática, se van a capturar los datos de los diferentes núcleos de población de Mallorca, susceptibles a formar parte de esta creación de rutas.

- **Limpieza y tratamiento de los datos.** Los datos capturados deberán limpiarse para verificar su integridad y tratarse para obtener la información necesaria para aplicarlos al problema deseado.

- **Obtención de los puntos a visitar.** Dados los datos ya procesados para el estudio, se van a elegir ciertos puntos de forma aleatoria que se considerarán como posiciones a visitar por el transportista.

- **Diseño de diferentes modelos.** El objetivo es diseñar diferentes modelos de planificación de rutas en abierto en los que se apliquen técnicas de origen diverso.

- **Aplicación de los modelos.** Una vez se dispone de los modelos deseados y los puntos a visitar, se llevará a cabo una aplicación de los métodos para comprobar sus resultados.

- **Cálculo de la eficiencia de cada modelo.** A partir de los diferentes modelos desarrollados, el objetivo es comparar los resultados de cada uno de ellos y su eficiencia computacional.

- **Análisis y validación de los resultados.** Con los resultados obtenidos se podrá valorar cuál es el mejor algoritmo según cada caso en concreto y sobre qué condiciones es mejor usar uno u otro.

Capítulo 4

Desarrollo del trabajo

En este capítulo se explicarán los pasos llevados a cabo para alcanzar los objetivos específicos definidos en el capítulo anterior. Para ello se explicarán las herramientas usadas, los datos que han sido capturados y los modelos diseñados para optimizar las rutas de transporte.

4.1. Herramientas de software

Para desarrollar este estudio se han usado fundamentalmente las herramientas de software libre Docker y Python. Docker se ha usado para cargar el servicio que permitirá obtener información sobre los mapas y carreteras, mientras que Python se ha decidido usar como lenguaje de programación por su facilidad en el tratamiento de los datos y el gran número de funciones y librerías disponibles.

Docker (Docker, 2022) es una herramienta que fue lanzada en 2013 por Solomon Hykes y en dos años se convirtió en el software más demandado de la industria tecnológica (Agarwal, Jain, y Kumar, 2021). Esta herramienta permite automatizar el despliegue de servicios dentro de contenedores de software, lo que incrementa la productividad de los desarrolladores y reduce los costes operacionales. En este caso en concreto, se ha usado para desplegar la información sobre los mapas y carreteras de Mallorca. Con el uso de este servicio se permite conocer cuál es la distancia entre dos puntos o el tiempo estimado, además de otra información geográfica.

Python (Python Software Foundation, 2022) es una herramienta con licencia de código abierto lanzada en el mercado en el año 1991 por Guido Van Rossum. Es uno de los lenguajes de programación más usados hoy en día, orientado a objetos y con semántica dinámica,

que permite desarrollar todo tipo de aplicaciones (LUCA, 2020). Python dispone también de librerías científicas que facilitan las operaciones, como NumPy, Pandas o Matplotlib, entre otras. Este software es el que se ha usado para procesar conectarse al servicio de Docker, procesar los datos necesarios y ejecutar los algoritmos que previamente se han desarrollado con este lenguaje. Para el desarrollo del estudio se ha usado la versión 3 de Python, la cuál presenta ciertas modificaciones sobre versiones anteriores proporcionando resultados ligeramente diferentes en algunas operaciones.

4.2. Captura y procesado de los datos

Para empezar con el desarrollo del trabajo ha sido necesario en primera instancia disponer de los datos geográficos con los que se ha trabajado. Para ello se han descargado los mapas de la isla de Mallorca, situada en las Islas Baleares, España (Geofabrik, 2022). Esta información se ha desplegado en Docker para disponer de un servicio que permita obtener la información geográfica.

Una vez se ha tenido disponible toda la información geográfica desplegada, con la ayuda de la librería *folium* de python se han implementado ciertas funciones que permiten obtener información sobre el mapa, como distancia o trayecto entre dos puntos dados o representaciones gráficas sobre los recorridos deseados sobre el mapa, entre otros. La API del servicio desplegado permite, usando peticiones HTTP de OSRM (OSRM, 2012) conocer información como la distancia entre dos puntos, el nombre de la localidad o tiempo estimado del trayecto en función del medio de transporte, entre otros.

La ventaja que proporciona esta librería frente a otras es que no calcula únicamente la distancia en línea recta entre dos puntos, sino que tiene en cuenta la distancia del semiverseno. La distancia del semiverseno (Cotter, 1974), conocida en inglés como *Haversine*, calcula la distancia angular entre dos puntos de la superficie terrestre. Sea (x_1, y_1) la longitud y latitud de un punto en la superficie terrestre y (x_2, y_2) otro de los puntos. La fórmula del semiverseno indica cuál es la distancia entre estos dos puntos, la cuál viene dada por la ecuación 4.1.

$$D(x,y) = 2 \cdot arcsin\left(\sqrt{sin^2\left(\frac{x_1 - y_1}{2}\right) + cos(x_1) \cdot cos(y_1) \cdot sin^2\left(\frac{x_2 - y_1}{2}\right)}\right). \quad (4.1)$$

Por ejemplo, si queremos obtener la ruta desde el pueblo de Sineu (longitud 2.666300

y latitud 39.615615) al pueblo de Son Sardina (longitud 3.0288 y latitud 39.6380), una vez tenemos desplegado el servicio, se llamaría a él usando la URL:

```
http://IP/route/v1/driving/2.666300,39.615615;3.0288,39.6380
```

Esta URL se puede modificar en función de la información que se desee obtener. En este caso en concreto:

- IP: sería la IP del servidor en el cuál se ha desplegado el servicio. Puede ser *localhost* si es en el escritorio local o la dirección en caso de una IP externa.
- *route*: tipo de servicio al que se está llamando. En este ejemplo nos interesa una ruta, la cuál tiene como resultado los pasos a llevar a cabo para ir desde el origen hasta el destino.
- *v1*: versión del código usada, actualmente sólo disponible la 1a versión.
- *driving*: indica el modo de transporte. En este caso queremos ir con un vehículo motorizado, pero existen otras alternativas como *bike* si se va a realizar el desplazamiento en bicicleta o *foot* si va a ser andando.
- Coordenadas: en esta versión se tienen que indicar las coordenadas en formato JSON, teniendo el formato (longitud origen, latitud origen; longitud destino, latitud destino).

La respuesta que se obtiene es en formato JSON, de la cuál se muestra una parte a continuación.

```
{ 'code': 'Ok',
        'routes': [{
                'geometry': 'uuxpFusgOkFz[}e@w@~hBhf@|{@pCljA{nAu|Ai~
                    @qkDy_GqLanCwj@{zAaC{vBqk@czAq}A{vBygAkaCwbA{_Dyj@
                    }{@arCy~I~l@wl@dkAke@bnCoyCdgBk`GnGg~@tgAmfA{Is}@p[nY
                    ',
                'distance': 47297.1,
                'duration': 2331,
                'weight_name': 'routability',
                'weight': 2331
        }]
```

```
}
```

La primera línea nos indica si la petición ha sido correcta o no, mientras que después se muestran algunas medidas referente a la ruta que se ha creado. En este caso en concreto, se tardaría 2331 segundos para recorrer los 47297 metros que separan ambos puntos. El resultado que aparece bajo *geometry* sería, en lenguaje codificado, las indicaciones que se tienen que seguir para realizar el recorrido del punto inicial al punto final. Por otra parte, el *weight* sería el coste de realizar la ruta, en este caso se ha calculado como coste el tiempo total del trayecto. Estos valores son modificables acorde a la información que interese en cada momento.

Usando la librería *polyline* se puede decodificar la ruta obtenida y realizar una representación gráfica marcando los diferentes puntos del trayecto sobre el mapa de la zona en la que estamos trabajando, la cual se puede observar en la Figura 4.1.

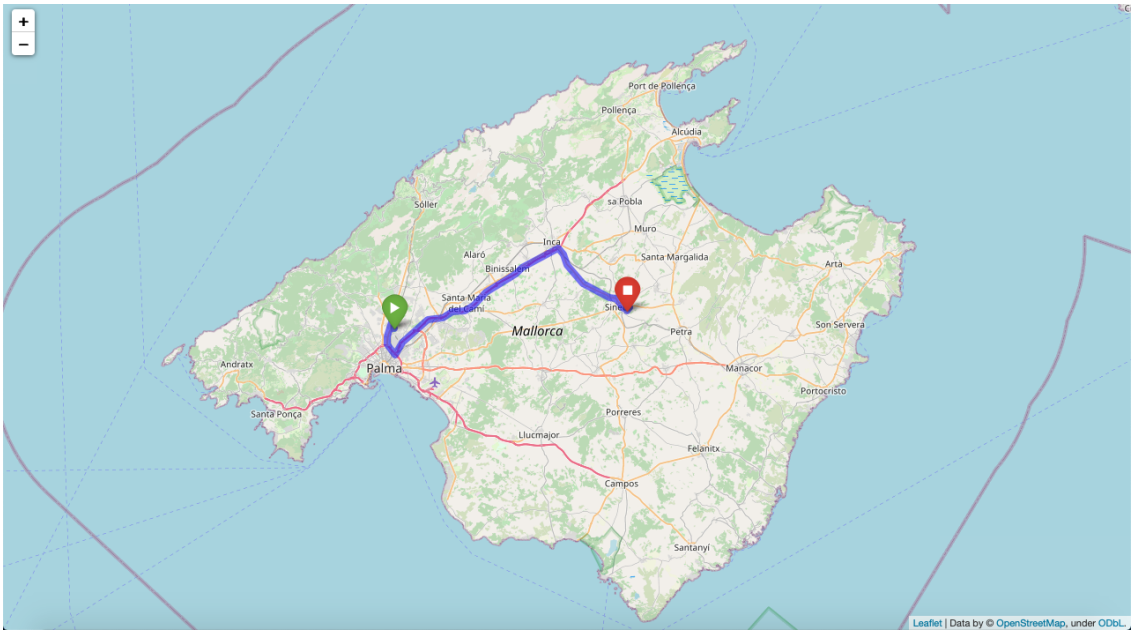


Figura 4.1: Ejemplo de ruta. Fuente: elaboración propia.

Una vez se dispone de la información geográfica correspondiente, se han descargado todos los núcleos de población de Mallorca con el objetivo de poder simular rutas entre ellos. Según el Instituto Nacional de Estadística (*Instituto Nacional de Estadística*, s.f.), se considera como núcleo de población un conjunto de al menos diez edificaciones, que formen calles, plazas y otras vías urbanas. Los datos disponibles son los publicados por el Gobierno de las Islas Baleares (Govern de les Illes Balears, 2022), por lo que se corresponden a los

datos de núcleos de toda la comunidad autónoma. Dado que el estudio se centra en la isla de Mallorca, ha sido necesario depurar esos datos para eliminar los de las islas de Menorca, Ibiza y Formentera. En la Figura 4.2 se puede observar una muestra de los datos descargados y su formato.

NOM	Nom INE	Municipi	Codi INE Municipi
Planera	000105 PLANERA-SES TREMPES-SON DAVIU	Marratxí	7036
Son Granada (Les Palmeres)	001001 PALMERES (LES)	Llucmajor	7031
Calonge	000401 CALONGE	Santanyí	7057
Sol de Mallorca	001601 SOL DE MALLORCA	Calvià	7011
Sa Cabaneta	000101 CABANETA (SA)	Marratxí	7036
Biniagual	000101 BINIAGUAL	Binissalem	7008

Figura 4.2: Muestra de los datos de los núcleos de población de Mallorca. Fuente: Gobierno de las Islas Baleares (Govern de les Illes Balears, 2022).

Para poder calcular las distancias entre diferentes núcleos de población, necesitamos disponer de la longitud y latitud de cada uno de ellos, ya que como hemos visto esta es la información de entrada que se usa para conectar con el servicio que nos proporcionará la ruta. En nuestro caso, los datos descargados contienen información como el nombre de la población, el municipio o el código postal, pero no longitud ni latitud. Para conseguir estas unidades, dentro de la librería *geopy*, ya implementada en Python, existe una función llamada *geocode* a la cuál se le pasa como parámetro el nombre de la ubicación deseada y retorna dichas métricas. Para mejorar la precisión de los resultados obtenidos, se ha usado como entrada de la función el nombre de la zona concatenado con el municipio.

Dado que estamos frente a un conjunto de datos relativamente pequeño (242 núcleos de población) se ha revisado manualmente que la dirección completa de la zona localizada por la librería es la misma que estábamos buscando, sobre el cuál se han requerido pequeñas correcciones manuales al estar frente a pueblos con nombres parecidos en diferentes ubicaciones. Finalmente la información de la longitud y latitud se ha añadido como dos columnas extra a los datos proporcionados por el gobierno. En la Figura 4.3 se puede observar una muestra de los resultados obtenidos, cuyos datos han sido usados para realizar el estudio.

Para el desarrollo del estudio se ha considerado también que el centro de distribución está situado en Inca, al ser ésta la ciudad más céntrica de la Isla. Notemos sin embargo que éste podría ser otro problema de estudio en el que se de deberían tener en cuenta

NOM	Nom INE	Municipi	Latitude	Longitude
Planera	000105 PLANERA-SES TREMPES-SON DAVIU	Marratxí	39.6411269	2.707566428826291
Les Palmeres	001001 PALMERES (LES)	Llucmajor	39.4720157	2.7490457
Calonge	000401 CALONGE	Santanyí	39.4007378	3.2039972
Sol de Mallorca	001601 SOL DE MALLORCA	Calvià	39.4824894	2.5283191
Sa Cabaneta	000101 CABANETA (SA)	Marratxí	39.6216062	2.7502292

Figura 4.3: Muestra de los datos de los núcleos de población de Mallorca con su longitud y latitud. Fuente: Elaboración propia.

muchos otros factores como el coste de la infraestructura en las diferentes ubicaciones o la cantidad de pedidos que realizan los habitantes de cada zona, entre otros.

4.3. Generación de valores aleatorios

La muestra de datos obtenida de los diferentes núcleos de población de Mallorca está compuesta por un total de 242 núcleos, por lo que se ha decidido seleccionar una muestra aleatoria de este conjunto, la cuál se entenderá como los núcleos a visitar durante la jornada laboral de un repartidor.

Una secuencia de valores aleatorios es aquella que cumple obligatoriamente con dos condiciones:

1. Los valores están uniformemente distribuidos en un conjunto o intervalo,
2. No es posible predecir valores futuros conociendo los valores pasados y el presente.

Existen multitud de investigaciones sobre los diferentes métodos de generación de valores aleatorios y la complejidad de los mismos (Kuss, Griffiths, Binder, y Street, 2013). En Python existen diferentes funciones ya implementadas que permiten la generación de valores aleatorios, una de las más usadas es la llamada *randint()* (Brownlee, 2018). Debido a que estamos frente a números aleatorios, téngase en cuenta que cada vez que ejecutemos la función el resultado será diferente. Esto puede ocasionar ciertos inconvenientes en el momento, por ejemplo, de comparar diferentes modelos, ya que si se usa la generación de números aleatorios la entrada del algoritmo será diferente por lo que también su salida, imposibilitando así su comparación. Este hecho puede solucionarse fijando una semilla inicial, por lo que cuando se inicie la ejecución los valores se generarán en función de esa semilla y el orden de los resultados obtenidos será siempre el mismo, manteniéndose la pseudo-aleatoriedad.

La generación de valores aleatorios puede proporcionar valores repetidos, especialmente si se trabaja con un conjunto reducido de valores o se desean muestras muy grandes. En este caso en concreto, se sobreentiende que si el repartidor tiene dos pedidos en el mismo núcleo de población los va a repartir de forma consecutiva, por lo que para la elaboración de la ruta final se considerará como un único núcleo a visitar. Para evitar estas repeticiones en la generación de valores aleatorios, existe otra función en Python que proporciona dichos valores sin reemplazamiento. Esto quiere decir que un valor que ya se ha seleccionado no se volverá a a tener en cuenta, evitando su repetición. Esta función se llama *sample()* y tiene dos entradas: la lista de valores posibles y la cantidad de valores que se desean extraer. En este caso en concreto, se ha considerado como lista de posibles valores la posición del índice de cada núcleo de población (valor entero) y una muestra de tamaño 10.

```
seed(1)
sequence = [i for i in range(mapMallorca.index.min(),mapMallorca.index.max(),1)]
sample(sequence, 10)
>> [115, 120, 166, 97, 201, 53, 24, 124, 7, 228]
```

4.4. Planteamiento del problema

Una vez introducidas las condiciones del estudio, vamos a plantear en detalle el problema que se pretende afrontar. A nivel conceptual, la idea es optimizar la ruta que debería realizar un transportista fijados ciertos puntos que tiene que visitar durante su jornada laboral.

Imaginemos que tenemos ciertos puntos marcados en un mapa de carreteras, el cuál podría representarse usando grafos como se muestra en la Figura 4.4. En este caso en concreto, se ha marcado en color verde el centro de distribución, desde el cuál saldría el repartidor al iniciar su jornada laboral y volvería al finalizarla. Los puntos a visitar se han marcado en color negro y una letra, mientras que las aristas indican si dos puntos están conectados entre ellos y su distancia en kilómetros. En la figura se muestran dos posibles casuísticas, la A y la B.

En el caso de la situación A, puede verse bastante claro que habría dos rutas igualmente óptimas en cuanto al número de kilómetros totales, la que visita los puntos en el orden $A \rightarrow B \rightarrow C$ y la que realiza el recorrido en orden $C \rightarrow B \rightarrow A$. En ambos casos, la distancia total recorrida por el transportista sería 29 kilómetros. Visitar la ciudad

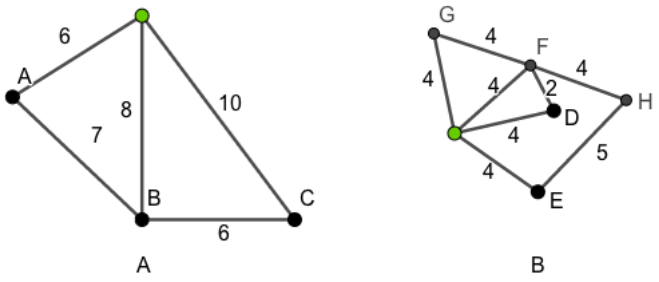


Figura 4.4: Representación de un mapa de carreteras usando grafos. Fuente: Elaboración propia.

B en primera instancia sería una opción descartada ya que esto implicaría quedar mal posicionado respecto al resto de puntos a visitar.

En el caso de la situación B, en cambio, no se vería tan claro a primera vista cuál sería la solución óptima para recorrer todos los puntos. A priori se podría pensar que seguir una técnica similar a la de la situación A pero quedaría un punto central D que habría que determinar cuál es el mejor momento para visitarlo. En el caso de disponer de únicamente 5 puntos a visitar, evaluar todas las posibles opciones y quedarse con la mejor podría ser una solución. Sin embargo, esta opción dejaría de ser válida si estamos frente a una gran cantidad de puntos a visitar.

En este caso se ha planteado como ejemplo el problema del viajante considerando que se desea optimizar la distancia total de la ruta, pero a lo largo del estudio se tendrán en cuenta diferentes factores como la distancia o el tiempo, entre otros.

Planteemos el problema formalmente usando teoría de grafos. Sea $G = (V, A)$ un grafo donde V son los núcleos de población a visitar, $V = \{0, \ldots, n\}$ con 0 el punto de salida del transportista y $A = \{(i, j) : i, j \in V, i \neq j\}$ las conexiones entre los puntos, correspondientes a las carreteras existentes. Supongamos sin pérdida de generalidad que siempre se podrá llegar desde un punto a otro sea el trayecto más o menos costoso, i.e. el grafo es completo. En caso de puntos aislados, en los cuáles no se puede llegar a ellos a través de ningún medio de transporte, se sobreentiende que no se ofrecerá envío a dichos puntos, teniendo dicha arista un coste infinito a efectos computacionales.

Sea W la matriz de costes donde w_{ij} representa el coste para ir desde el punto i hasta el punto j, ya sea en tiempo, distancia o combinación de diferentes factores. Esta matriz tendrá valores infinitos en la diagonal ya que nunca iremos desde un punto a sí mismo,

por lo que su coste se considera infinito a fin de ser una opción siempre descartada por los modelos. Dado que las carreteras son bidireccionales, ésta será una matriz simétrica similar a la de la Ecuación 4.2, en la que se muestra una posible situación con 3 ciudades a visitar. El coste de viajar de la ciudad 1 a la ciudad 2 sería 4, mientras que el coste de ir de 1 a 3 sería 7.5 y así sucesivamente.

$$\begin{pmatrix} \infty & 4 & 7{,}5 \\ 4 & \infty & 3 \\ 7{,}5 & 3 & \infty \end{pmatrix}. \tag{4.2}$$

Definamos también Y como la matriz de tamaño $(n+1) \times (n+1)$ que identificará la ruta, de tal manera que si y_{ij} es 1 indica que el transportista irá desdes el punto i hasta el punto j y 0 en caso contrario.

El problema de optimización planteado es:

$$\text{minimizar} \qquad \sum_{i=1}^{n} \sum_{j=0}^{n} w_{ij} \cdot y_{ij}, \tag{4.3}$$

$$\text{sujeto a:} \quad \sum_{i=0}^{n} y_{ij} + \sum_{j=0}^{n} y_{ij} = 2 \quad \forall i \in \{0, \ldots, n\}, \tag{4.4}$$

$$y_{ij} \in \{0, 1\} \qquad \forall (i,j) \in A, \tag{4.5}$$

$$w_{ij} \geqslant 0 \qquad \forall (i,j) \in A. \tag{4.6}$$

La ecuación (4.3) es la función objetivo de la que queremos hallar la solución óptima, la cuál representa el coste total del trayecto realizado por el transportista según la medida elegida. las ecuaciones (4.5) y (4.6) definen el comportamiento de cada variable del sistema, siendo en el caso de y_{ij} 1 si el trayecto forma parte del recorrido total o 0 en caso contrario, mientras que w_{ij} será siempre positivo ya que siempre existe un coste para desplazarse desde un punto a otro. Así pues, una combinación de las ecuaciones (4.4) y (4.5) nos indican que cada punto será visitado una única vez, teniendo así cada nodo grado 2.

4.5. Diseño de algoritmos

En esta sección se van a desarrollar diferentes algoritmos que proporcionen una solución al problema planteado anteriormente.

4.5.1. Algoritmo de programación dinámica

Como hemos visto anteriormente, una forma de obtener la solución óptima del problema de optimización es calcular todas las posibles rutas y quedarse con la que ofrece menor coste. Esta técnica se conoce también como algoritmo de fuerza bruta. Este tipo de algoritmo, consisten pues en consisten en calcular todas las posibles combinaciones que podría realizar el repartidor y finalmente quedarse con la que haya obtenido mejores resultados según la función objetivo planteada.

Siguiendo con la situación A planteada en la Figura 4.4, una posible forma de solución sería crear el árbol que se muestra en la Figura 4.5. En dicho grafo se muestra cómo se crearían las posibles rutas para llegar a la solución óptima.

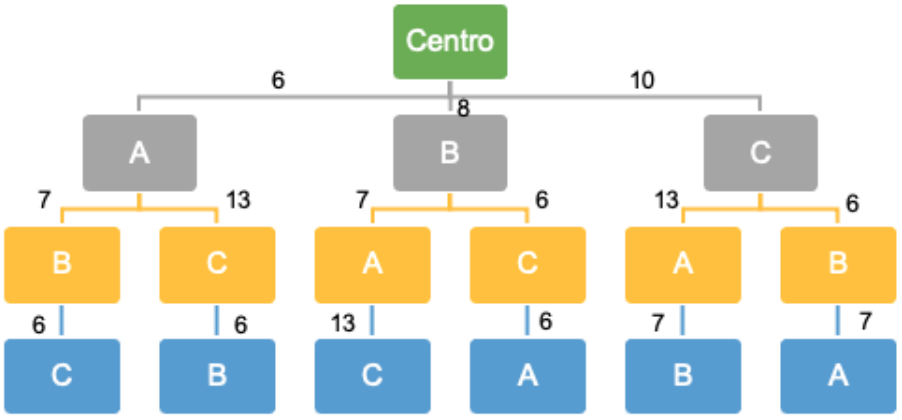


Figura 4.5: Pasos del algoritmo de programación dinámica. Fuente: Elaboración propia.

Inicialmente se sale desde el centro de distribución y se evalúan las posibles opciones para la primera parte del trayecto. Según cuál sea esta opción, después habrá dos posibles opciones de puntos a visitar, correspondientes a los dos todavía pendientes. Al realizar esta segunda parte del trayecto, finalmente sólo quedará un punto para visitar. Para cada trayecto, se indica en color negro el coste del mismo siendo la ruta óptima la que tenga suma total inferior. Por claridad no se muestra en el grafo el retorno al centro de distribución.

Como ya se veía en la Figura 4.4, los trayectos de menor coste serían $A \rightarrow B \rightarrow C$ y $C \rightarrow B \rightarrow A$, teniendo en cuenta que en la figura no se muestra el retorno al centro de distribución.

Para el caso concreto de tres ciudades a visitar planteado, se puede observar como existen 6 posibles combinaciones de caminos, los cuáles se evaluarán para determinar el

óptimo. Usando esta técnica, ya se puede intuir que cuántas más ciudades haya por visitar la evaluación de todas las posibles combinaciones puede ser muy elevado.

Formalmente, los pasos que seguiría el algoritmo de fuerza bruta para obtener la solución son:

1. Calcular todas las posibles permutaciones entre los diferentes núcleos de población a visitar.
2. Calcular el coste total de la permutación según la métrica deseada.
3. Si la permutación obtenida es mejor que la encontrada hasta el momento, se guarda como mejor.
4. Se repiten los pasos anteriores para todas las permutaciones.

Obsérvese que al tener calcular y evaluar todas las posibles permutaciones, dados n puntos se podrían llegar a tener hasta $n!$ rutas diferentes (Chang y Xue, 2022) en el caso que todas las ciudades estén conectadas entre ellas..

Esta metodología es, por ejemplo, la que usaría el problema clásico del viajero. Una posible implementación del mismo es la que se muestra a continuación.

```
def travellingSalesmanProblem(W, s=0):
    Q = [] # points to visit
    for i in range(len(W)):
        if i != s:
            Q.append(i)
    # save minimum route
    min_path = maxsize
    best_route = ()
    next_permutation=permutations(Q)
    # iterate over all permutations
    for i in next_permutation:
        #compute weight
        current_pathweight = 0
        u = s
        for j in i:
            current_pathweight += W[u][j]
```

```
            u = j
        current_pathweight += W[u][s]

        # update minimum
        if current_pathweight < min_path:
            min_path = current_pathweight
            best_route = i

    best_route = (s,) + best_route[0:] + (s,) #Add first and end point
    return min_path, list(best_route)
```

Las entradas de la función serían la matriz de pesos, W, con el coste de la medida considerada y el punto de inicio, denotado por s e inicializado en el punto 0, correspondiente al centro de distribución. Posteriormente se guardaría en una lista, Q, los puntos que se tienen que visitar, por lo que inicialmente sería una lista con todos los puntos a excepción del inicial. Después se prosigue con los puntos anteriormente comentados: se calculan todas las posibles combinaciones y para cada una de ellas el coste según la matriz de pesos. En caso que el coste total encontrado para un recorrido sea mejor que el encontrado hasta el momento, se actualiza la mejor ruta, correspondiente a la variable *best_route*. Para finalizar se añade a la ruta óptima el primer y último punto, correspondientes al centro de distribución.

El gran inconveniente que surge en su uso es el coste computacional, ya que al evaluar todas las posibles combinaciones éste crece de forma muy pronunciada cuando se incrementan los puntos. En caso que el repartidor tenga que visitar muy pocos núcleos de población, éste podría ser un buen algoritmo al ofrecer una solución óptima. Sin embargo, cuando es necesario visitar varios núcleos o, inclusive, aplicar este mismo algoritmo para todos los repartidores que puede tener una misma empresa, se vuelve insostenible su uso debido al coste computacional.

4.5.2. Modificación del algoritmo de Prim

El Algoritmo de Prim tiene como objetivo encontrar un árbol de recubrimiento mínimo en un grafo conexo y no dirigido, en el cuál cada arista tiene un peso (Prim, 1957). Éste es el caso en el que nos encontramos ya que en Mallorca todos los núcleos de población,

que serían los nodos del grafo, son accesibles por alguna carretera, que serían las aristas del grafo, por lo que sería conexo. Por otra parte, al ser todas las carreteras doble sentido, también se puede considerar como grafo no dirigido. Finalmente, el peso de las aristas podría ser cualquier medida cuantitativa que cuantifique el transporte entre un núcleo de población y otro, ya sea la distancia en tiempo, en kilómetros o el coste en combustible, entre otros.

La idea general del algoritmo es inicializar con un único vértice, que será el origen (en nuestro caso el almacén) e ir incrementando el árbol de tal forma que cada nodo se una a él usando el punto más próximo. Este objetivo difiere del planteado en este trabajo ya que esto implicaría que el repartidor vuelva hacía atrás para proveer otro punto. En la Figura 4.6 se muestra una comparativa entre el resultado del algoritmo de Prim y el deseado.

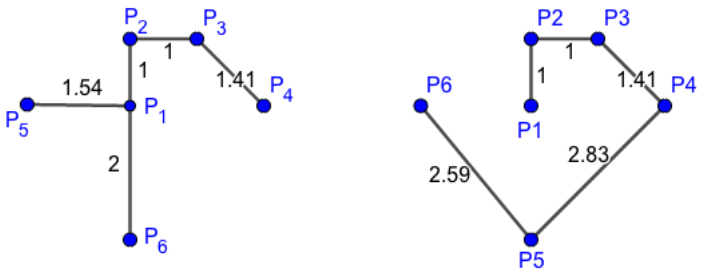


Figura 4.6: Ejemplo de árbol de recubrimiento mínimo (izquierda) comparado con el resultado que se desea obtener (derecha). Fuente: Elaboración propia.

Por esta razón, se ha modificado ligeramente el algoritmo para simular que el repartidor irá siempre al núcleo de población que le quede más próximo a su posición, obviando los ya visitados. Luego los pasos del algoritmo serán:

1. Inicializar el árbol con un único punto, correspondiente a la posición del centro de distribución o almacén.

2. Desplazarse al núcleo de población más próximo.

3. Actualizar la lista de puntos restantes eliminando el inicial y los ya visitados.

4. Repetir los pasos anteriores hasta que no queden núcleos pendientes de visitar.

A continuación se muestra el código implementado en Python:

```
def prim(W, s=0):
    P, Q = {}, [(0, None, s)] #P = visited points, Q = pending points
    while Q:
        w, p, u = heappop(Q) # weight, parent, point
        if u in P: continue
        P[u] = w
        for v in range(len(W[u])):
            heappush(Q, (W[u][v], u, v))
        Q = list(filter(lambda x: x[1] != p, Q))

    best_route = list(P.keys())
    best_route.insert(len(P), s) #Add destination point
    min_path = sum(P.values()) + W[best_route[len(best_route)-2]][s] #Total Km
    return min_path, best_route
```

Para el desarrollo del algoritmo se ha tenido en cuenta la misma notación que en el método anterior. W sería la matriz de pesos, s el punto de inicio del trayecto, Q los puntos pendientes a visitar y P, una nueva variable en este algoritmo, los puntos ya visitados. Mientras haya puntos pendientes a visitar, se extrae del listado el que tenga menor coste. Una vez visitado un nuevo punto, se actualiza la lista de puntos accesibles excluyendo los ya visitados. Finalmente se añade el punto inicial y final a la ruta obtenida para poder calcular su coste total.

4.5.3. Algoritmo genético

Un algoritmo genético tiene como objetivo buscar la mejor solución al problema considerando un espectro de posibles soluciones. Para ello se empieza con ciertos estados iniciales sobre los cuáles se van aplicando ciertas alteraciones, con el objetivo de mejorar la solución a medida que vayan desarrollándose más iteraciones.

Existen diferentes técnicas para alterar la solución en función de cuál sea el objetivo final. En este caso se usará el principio de selección de la ruleta (Lipowski y Lipowska, 2012). Con este método la probabilidad de selección de cada núcleo de población depende del coste proporcional que tiene respecto al coste total de la ruta. Es decir, si hay un punto que esté muy alejado de todo el resto, tiene mayor probabilidad de ser uno de los elegidos

en la alteración de las posibles soluciones. En la Figura 4.7 se muestra una representación gráfica de un escenario con cinco ciudades diferentes, estando una de ellas al doble de distancia del resto.

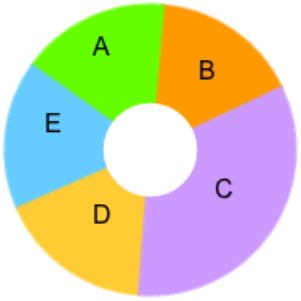


Figura 4.7: Principio de la ruleta para cinco ciudades, suponiendo una de ellas al doble de distancia. Fuente: Elaboración propia.

Los pasos a llevar a cabo para la implementación del algoritmo genético son:

1. Crear el estado inicial con ciertas soluciones creadas forma aleatoria y calcular su coste con la medida deseada. Por simplicidad en el primer paso se crearán tantas soluciones como puntos totales a visitar en el recorrido.

2. Seleccionar un nuevo conjunto de progenitores. En este caso se usará la selección de la rueda de la ruleta.

3. Para cada par de padres obtenido, generar un par de descendientes. Para evitar la repetición, se copiará un fragmento aleatorio de un progenitor y se llenarán los espacios en blanco con el otro progenitor.

4. Para cada elemento de la nueva población, agregar un posible intercambio de forma aleatoria.

5. Repetir los pasos anteriores hasta cumplir cierto criterio de parada, habitualmente fijado como máximo de iteraciones.

En la Figura 4.8 se muestra un ejemplo de cómo podrían ser los cruces para un caso simplificado en el que se tengan 5 ciudades: A, B, C, D y E.

Este tipo de algoritmos no siempre tienen como resultado la solución óptima, pero sí pueden proporcionar buenos resultados al estar frente a muchos núcleos de población a

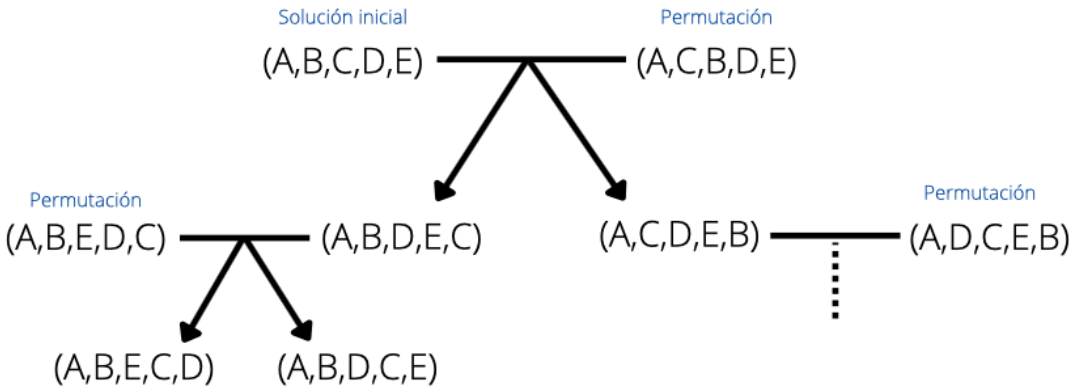


Figura 4.8: Ejemplo de árbol genético dadas cinco ciudades. Fuente: Elaboración propia.

visitar y ser inviable calcular todas las posibles rutas. La precisión del método dependerá también del número máximo de iteraciones fijado al inicio del problema, teniendo en cuenta que a cuantas más iteraciones, más cerca se estará de la solución óptima pero también mayor será el coste computacional.

4.6. Comparativa de resultados

Para poder comparar los diferentes algoritmos entre sí, se han elegido diferentes subconjuntos aleatorios generados con semillas diferentes y variando el número total de ciudades. Uno de ellos ha sido usando el valor 3 como semilla y 10 ciudades diferentes, además del centro de distribución que siempre será el punto inicial y final del recorrido. En la Figura 4.9 se muestra este primer subconjunto de núcleos de población a visitar.

En la Tabla 4.1 se muestran los resultados obtenidos considerando los 11 núcleos de población citados en la Figura 4.9. Para realizar una comparativa se ha ejecutado dos veces cada algoritmo, la primera de ellas considerando como matriz de costes la distancia entre los diferentes puntos (en kilómetros) y en la segunda ocasión considerando el coste como el tiempo de desplazamiento (en minutos).

El algoritmo con mejores resultados ha sido el de programación dinámica o fuerza bruta para ambas mediciones, pero es también el que presenta mayor coste computacional, con mucha diferencia respecto al resto. El algoritmo de Prim modificado ha sido probablemente el que ha ofrecido más equilibrio en este sentido. Ha sido el que ha tardado menos tiempo en ejecutarse y el incremento, ya sea en distancia o en tiempo, no supera el 1 % al finalizar

NOM	Address	Latitude	Longitude
Maioris Decima	Maioris Dècima, Migjorn, Illes Balears, 07609, España	39.454433	2.7513978
Es Mal Pas / Bonaire	es Mal Pas - Bonaire, camí de la Victòria, Mal Pas, Alcúdia, Raiguer, Illes Balears, 07410, España	39.8659449	3.1458309
Son Serra Perera	Son Serra Perera, es Secar de la Real, Palma, Illes Balears, 07010, España	39.5994453	2.6397526
Cala d'Or	Cala d'Or, Migjorn, Illes Balears, 07660, España	39.3748968	3.2302905
Valldemossa	Valldemossa, Serra de Tramuntana, Illes Balears, 07170, España	39.7108465	2.6230264
Alcúdia	Alcúdia, Raiguer, Illes Balears, 07400, España	39.8532473	3.1212505
Cala Santanyí	Cala Santanyí, Migjorn, Illes Balears, 07690, España	39.3305962	3.1450895
S'Empeltada	s'Empeltada (Deià), carretera de la Cala, Deià, Serra de Tramuntana, Illes Balears, 07179, España	39.7544533	2.6475571
Santanyí	Santanyí, 7, plaça de la Constitució, Santanyí, Migjorn, Illes Balears, 07650, España	39.3553499	3.1291638
Biniali	Biniali, Sencelles, Pla de Mallorca, Illes Balears, 07140, España	39.6403898	2.8599118

Figura 4.9: Puntos elegidos aleatoriamente para comparar los resultados de los diferentes algoritmos. Fuente: Elaboración propia.

Algoritmo	Distancia total (Km)	Tiempo ejecución (s)	Tiempo total (min)	Tiempo ejecución (s)
Fuerza bruta	269.7	25	286	25
Prim modificado	281.5	3	295	3
Genético (500 iter)	291	4	304	5
Genético (2000 iter)	285	15	300	12

Tabla 4.1: Resultados obtenidos según cada algoritmo y tiempo de ejecución considerando 11 puntos a visitar.

la jornada.

A partir de las ejecuciones con el algoritmo genético, puede verse como los resultados no son malos comparando con el algoritmo de Prim, sin embargo tiene un mayor coste computacional y no proporciona beneficios en el resultado final. También se puede observar como, para esta ejecución concreto que se ha llevado a cabo, no se nota una diferencia significativa en los resultados obtenidos al cuadruplicar las iteraciones del algoritmo, viéndose considerablemente afectado el tiempo computacional.

4.6.1. Comparación gráfica

Otro factor clave para la elección de los algoritmos ha sido la interpretación gráfica de las soluciones obtenidas, ya que con un número reducido de puntos se puede ver donde presentan más deficiencias cada uno de los algoritmos implementados. En la Figura 4.10 se muestra una representación gráfica de la ruta óptima. Como puede verse en este caso en concreto los 10 puntos aleatorios que han salido están repartidos a lo largo de la isla, lo

cuál implica que un algoritmo poco preciso podría suponer la realización de muchos más kilómetros de los necesarios.

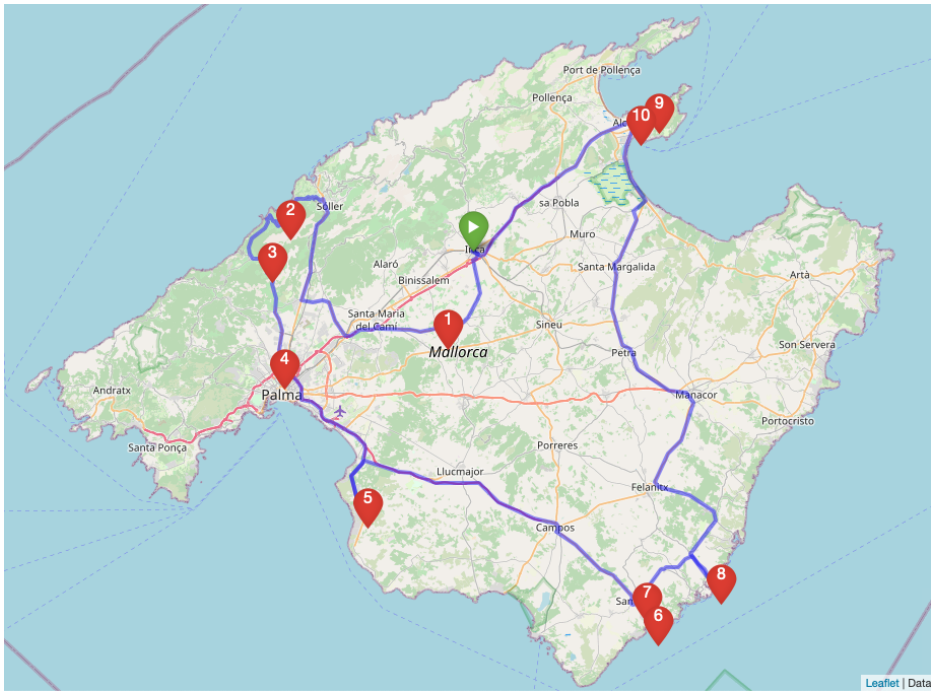


Figura 4.10: Resultado del algoritmo de programación dinámica. Fuente: Elaboración propia.

En la Figura 4.11 puede verse el resultado obtenido, para esos mismos 10 puntos, con el algoritmo modificado de Prim. Para este caso en concreto se presenta una gran similitud con el resultado obtenido con el algoritmo de programación dinámica.

Sin embargo, al ir siempre de cada punto al más cercano, se están buscando mínimos locales para cada caso, lo que provoca centrarse en un problema muy concreto que pierde eficiencia a nivel general en la ruta. Muestra de ello es que después del punto 1 se ha visitado el 2 por ser el más cercano, sin embargo para ir del 4 al 5 se ha pasado nuevamente por el punto 2 al estar en dicho camino. Este hecho se podría haber evitado si en lugar de en cada paso buscar un mínimo local se buscara el mínimo de un subconjunto mayor.

Para intentar minimizar el impacto de esta casuística, que puede afectar muy negativamente según los puntos aleatorios que se obtengan inicialmente, se desarrolló otro algoritmo que basaba el recorrido en sus coordenadas geográficas. Sin embargo, este algoritmo fue descartado inicialmente por los malos resultados obtenidos, ya que si dos puntos se encontraban en la misma latitud pero longitudes muy diferentes podía salir como mejor opción

frente a otros puntos muy cercanos con latitud y longitud diferentes ya que la diferencia total entre ambas medidas era mayor. Este algoritmo no se muestra en detalle a lo largo de la memoria al no presentar ningún beneficio respecto a los elegidos.

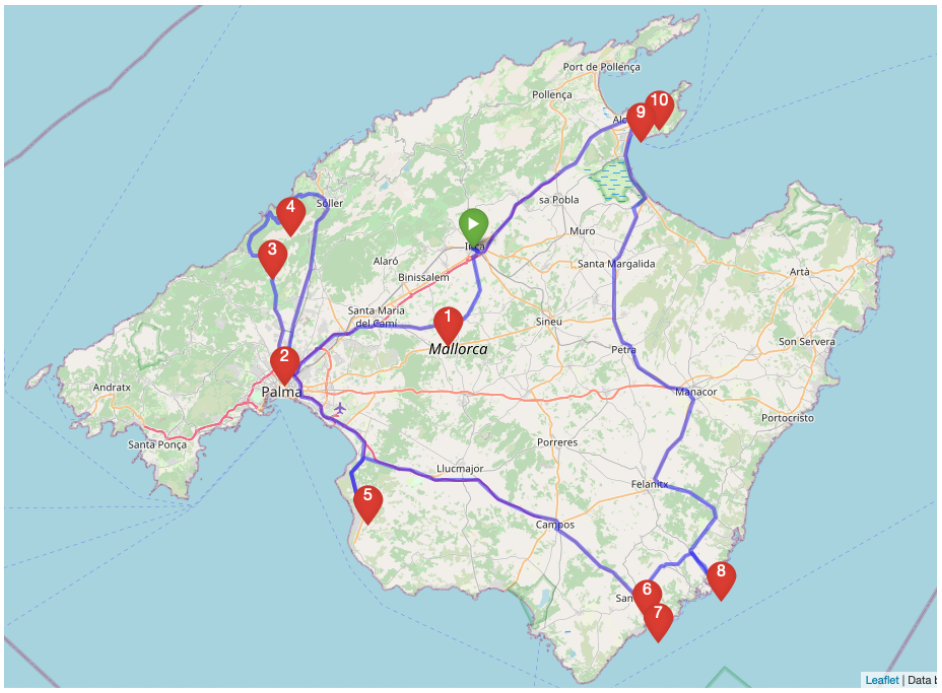


Figura 4.11: Resultado del algoritmo de Prim modificado. Fuente: Elaboración propia.

Para el caso del algoritmo genético, cuyo resultado se muestra en la Figura 4.12, se puede observar que la solución obtenida es muy similar a la obtenida con el algoritmo de programación dinámica, destacando principalmente dos puntos.

El primero de ellos es que prácticamente se invierte el orden de todos los puntos, es decir, el punto que había sido el primero ahora es el último, mientras que los últimos ahora son los primeros. Notemos que este hecho no afecta a nivel de eficiencia del algoritmo ya que, al ser todas las carreteras bidireccionales y ser una ruta circular, la variación total sería mínima o nula.

El segundo punto donde se pierde eficiencia es, desde el punto 2, visitar el punto etiquetado como 3 y no como 4. Como hemos visto anteriormente, desde el punto 2 hasta el 4 existe otra carretera directa que acortaría la distancia. Obsérvese que por el funcionamiento del algoritmo genético, el intercambio de los puntos 3 y 4 podría haber sido otra permutación válida que, al tener el algoritmo un gran comportamiento aleatorio, probablemente se habría dado esa casuística al incrementar el número de iteraciones.

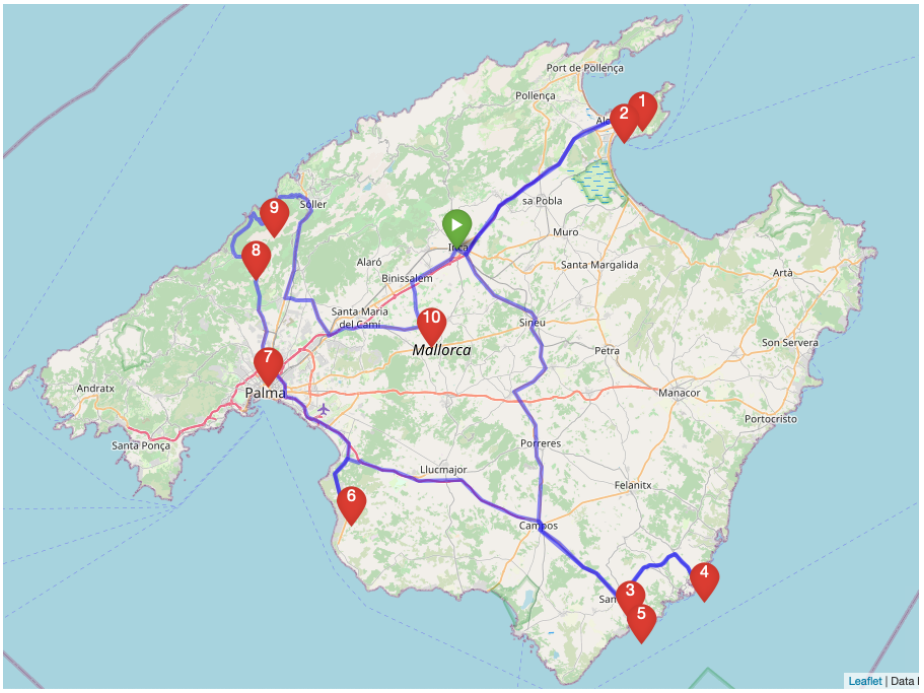


Figura 4.12: Resultado del algoritmo genético. Fuente: Elaboración propia.

4.6.2. Eficiencia computacional

Aunque a priori pueda parecer que la eficiencia computacional sea un factor poco relevante en la creación de rutas, este es también un factor clave para las empresas del sector, especialmente según la cantidad de transportistas de los que dispongan. Si se van a organizar rutas para 2 o 3 transportistas, puede ser un tema secundario el tiempo que tarde el algoritmo considerando que en pocos minutos cada uno puede disponer de su ruta con un coste reducido. Sin embargo, si se desean organizar rutas para decenas o inclusive cientos de transportistas no pueden esperar media jornada para saber cuál es la ruta más eficiente, ya que en este caso lo que se gane en un aspecto se perderá en otro.

Luego se puede considerar que la eficiencia computacional es también un factor clave en la creación de rutas. Estudiemos cuál es el comportamiento de cada uno de los algoritmos en este sentido. Por simplicidad a partir de ahora se realizará el estudio considerando como función objetivo la que minimiza la distancia del recorrido, siendo extrapolable a tiempo u otros factores.

Recordemos que el algoritmo de programación dinámica calcula todas las posibles permutaciones de los diferentes puntos a visitar para después contabilizar el coste total del trayecto resultante. Ya se puede intuir que a más puntos a visitar, el coste computacional

será mucho mayor. En la Figura 4.13 se muestra el tiempo de ejecución del algoritmo variando la cantidad de núcleos de población por los cuales va a pasar el recorrido.

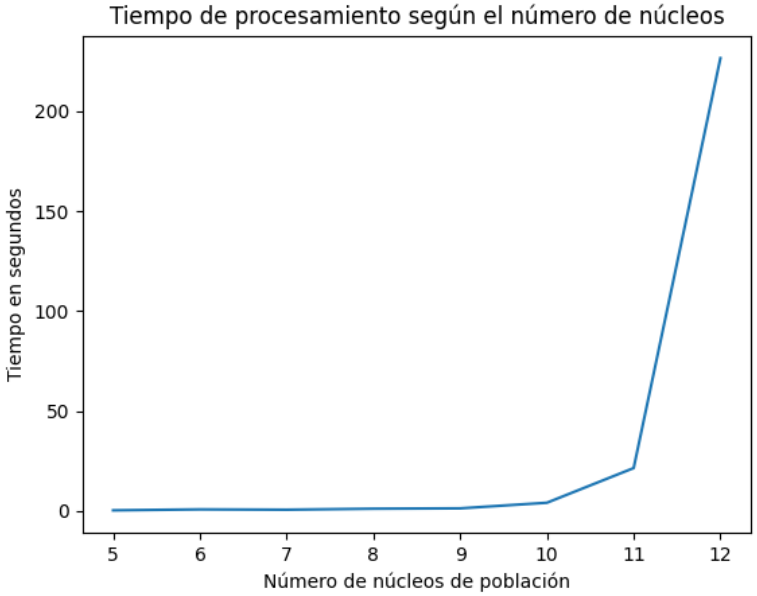


Figura 4.13: Tiempo computacional del método de fuerza bruta según el número de núcleos de población. Fuente: Elaboración propia.

En la gráfica puede verse como hasta 10 núcleos de población el tiempo de ejecución es de muy pocos segundos. Inclusive hasta con 11 puntos el tiempo sería de alrededor de 25 segundos, algo que podría aplicarse en el caso de casi cualquier compañía.

Sin embargo, cuando disponemos de 12 puntos el tiempo de ejecución ya es de más de 3 minutos, creciendo enormemente si hubiera más núcleos de población, motivo por el cuál se vuelve un método computacionalmente inviable para usar en casos prácticos. Por esta razón, a pesar que este algoritmo proporciona unos resultados óptimos, no será posible su uso cuando el transportista tenga que visitar 12 o más puntos.

En la Figura 4.14 se muestra el mismo comportamiento cuando el método usado para la creación de la ruta es el método modificado de Prim. Es decir, se pude observar el tiempo de ejecución del algoritmo según el número de núcleos de población a visitar.

Como se puede observar, usando este algoritmo el tiempo computacional crece de una forma más constante que con el algoritmo de programación dinámica. Hasta con 16 núcleos de población a visitar el tiempo computacional es de 6 segundos, lo cuál entraría en unos valores esperados para la resolución del problema. Por las características del

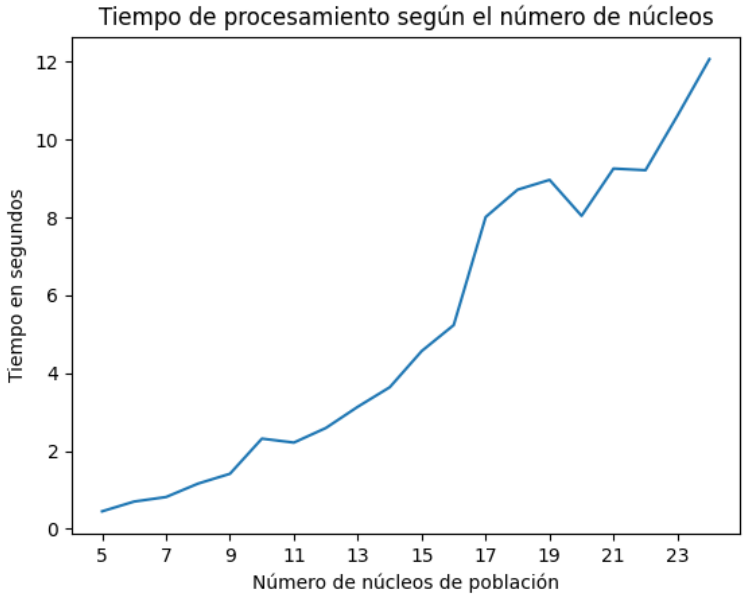


Figura 4.14: Tiempo computacional de la modificación del algoritmo de Prim según el número de núcleos de población. Fuente: Elaboración propia.

algoritmo, la operación más compleja que realizará es calcular la distancia del punto inicial al resto de ciudades, viéndose reducido este cálculo en próximas iteraciones ya que no se vuelven a evaluar ciudades ya visitadas. Por esta razón se puede ver como a medida que se van incrementando los núcleos de población el tiempo computacional no incrementa exponencialmente como en el caso anterior, sino que es un incremento suave pero constante.

Otro efecto a destacar es que en este caso la gráfica no es estrictamente creciente. Este hecho puede ser debido a la disposición inicial de los puntos y el carácter de aleatoriedad en las condiciones iniciales.

Si realizamos este mismo estudio con el algoritmo genético y 500 iteraciones se obtienen los resultados de la Figura 4.15.

En este caso estamos frente a un algoritmo que no es tan eficiente como el modificado de Prim pero, a diferencia del algoritmo de programación dinámica, es aplicable cuando se deben visitar grandes cantidades de puntos.

Como hemos visto uno de los componentes a tener en cuenta cuando se usa este algoritmo es la cantidad de iteraciones que se llevan a cabo, ya que cuantas más veces se alteren los progenitores más probabilidad hay de encontrar una solución óptima o más cerca de esta. Veamos en la Figura 4.16 cuál es el comportamiento del algoritmo al variar

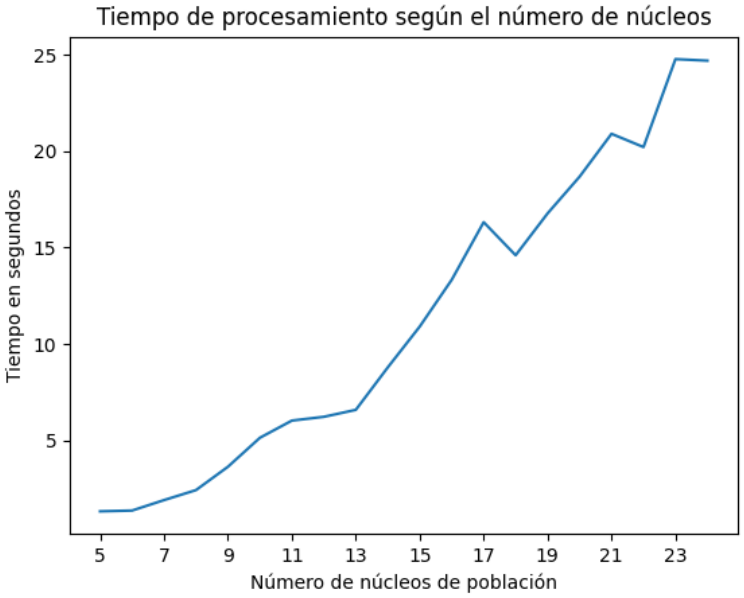


Figura 4.15: Tiempo computacional del algoritmo genético según el número de núcleos de población. Fuente: Elaboración propia.

la cantidad de iteraciones sobre el intercambio de progenitores.

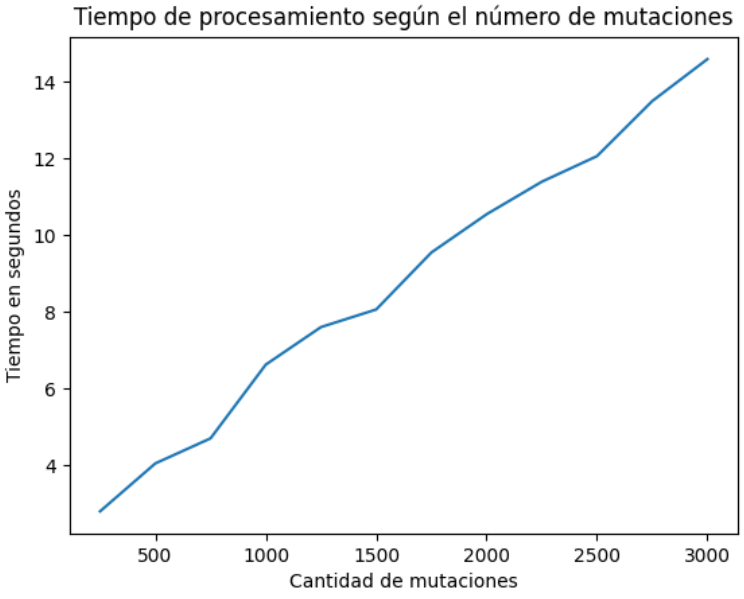


Figura 4.16: Tiempo computacional del algoritmo genético según el número de iteraciones. Fuente: Elaboración propia.

Para este caso en concreto, se ha ejecutado el algoritmo oscilando la cantidad de iteraciones entre progenitores entre 500 y 3000 y considerando intervalos de 250 entre ellos. Como era de esperar, el tiempo de ejecución incrementa progresivamente a medida que suben el número de iteraciones.

Sin embargo, en la Figura 4.17 se puede observar como los resultados no son los esperados. A priori pensaríamos que cuantas más iteraciones mejor debería ser el resultado del algoritmo, pero en la figura se muestra como para estos valores no se cumple.

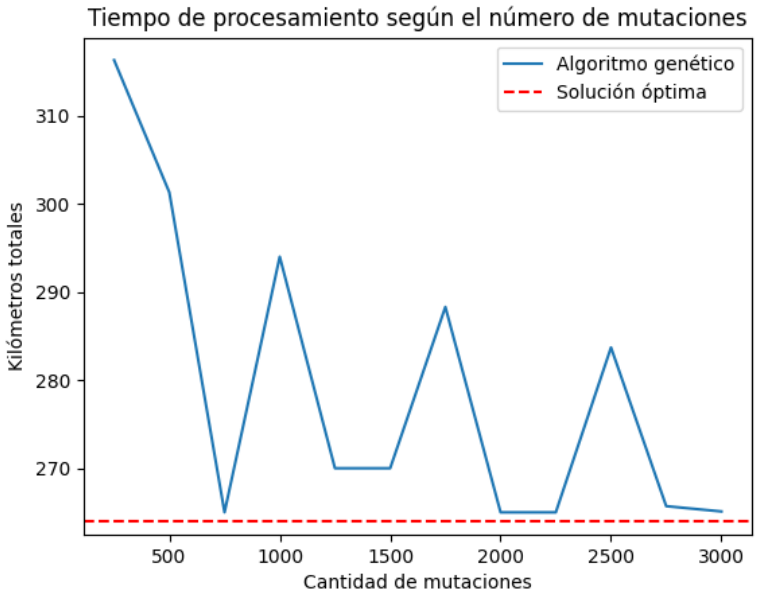


Figura 4.17: Kilómetros totales de la ruta variando el número de iteraciones en el algoritmo genético. Fuente: Elaboración propia.

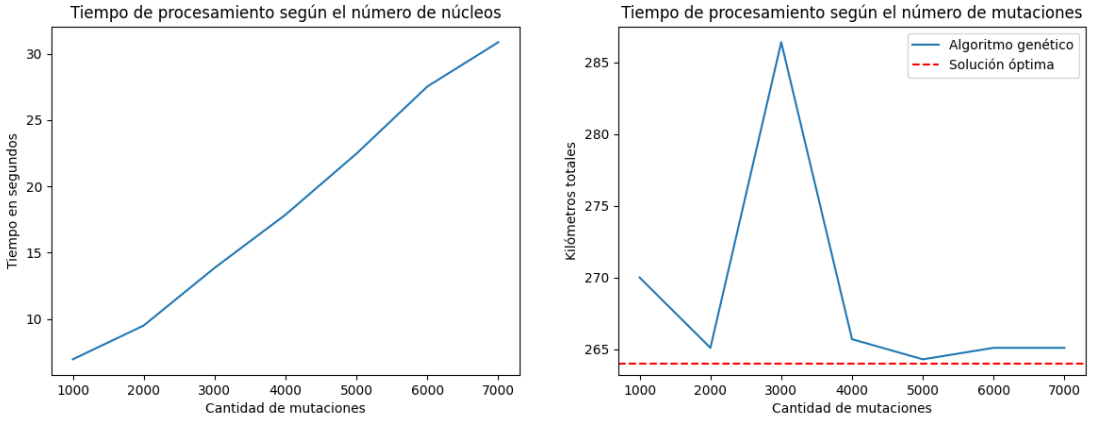
Este hecho es debido a que el algoritmo genético tiene un alto carácter aleatorio, provocando que en cada iteración se crucen los progenitores de forma totalmente aleatoria. De esta manera, podemos concluir que a más iteraciones no necesariamente la solución sea más cercana a la óptima, aunque sí incrementa la probabilidad que así sea. Para solucionar este inconveniente existen dos posibles opciones:

- Realizar múltiples iteraciones con la misma cantidad de mutaciones y quedarse con el mejor resultado. Este caso se basaría en aplicar el algoritmo múltiples veces bajos las mismas condiciones y quedarse con la iteración que haya devuelto mejores resultados, reduciendo notablemente así el carácter de aleatoriedad del algoritmo. En este caso el coste computacional se multiplicaría por tantas veces como iteraciones hagamos del

algoritmo bajo ciertas circunstancias, algo que a nivel de eficiencia no proporcionaría un buen resultado. Por ejemplo, para el caso de 2000 mutaciones se ha tardado unos 10 segundos en realizar la ejecución. Si repetimos este mismo caso 5 veces para eliminar el carácter aleatorio, el tiempo total de ejecución sería de 50 segundos, lo que convertiría el método en un algoritmo poco eficiente con grandes cantidades de puntos.

- Incrementar la cantidad de mutaciones a realizar. Inicialmente se han tomado entre 500 y 3000 mutaciones, pudiendo éstos no ser unos valores adecuados. A continuación se va a analizar cuál es el impacto de variar la cantidad de mutaciones a realizar en la ejecución del algoritmo.

En la Figura 4.18 se muestra el resultado, tanto en coste computacional (Figura 4.18a) como en distancia total a recorrer por el transportista (Figura 4.18b), variando el número de mutaciones del algoritmo.



(a) Tiempo de ejecución (b) Kilómetros totales

Figura 4.18: Tiempo computacional y kilómetros totales de la ruta variando el número de iteraciones en el algoritmo genético. Fuente: Elaboración propia.

En este caso sí se puede ver como a partir de 4000 mutaciones diferentes se mantiene cierta estabilidad en los resultados obtenidos mediante este algoritmo. Obviamente esta cantidad de mutaciones dependerá también de la cantidad de núcleos de población a visitar. Recordemos que en este caso nos habíamos centrado en un recorrido de 10 puntos, además del centro de distribución. A partir de esta representación se pueden elegir 4000 o 5000 iteraciones como medida para proporcionar un buen resultado, pero esto podría varia en

función de la cantidad de puntos a recorrer, ya que cuantos más puntos se tengan que recorrer más combinaciones podrá tener el algoritmo.

4.6.3. Resultados numéricos

Los resultados mostrados hasta ahora se corresponden al aplicar los diferentes algoritmos sobre un mismo conjunto de datos, el presentado en la Figura 4.9. Pero ya hemos visto que debido a las particularidades de cada algoritmo la eficiencia global de la ruta obtenida variará en función de los datos inicialmente introducidos. Por esta razón se mostrará una comparativa de los resultados obtenidos al aplicar los algoritmos a diferentes casuísticas.

A priori se podría pensar que existe una gran relación entre la cantidad de kilómetros a recorrer comparado con el tiempo que va a tardar el repartidor. Con el objetivo de probar la eficiencia de los algoritmos en diferentes entornos, vamos a ver si hay una relación entre la distancia y el tiempo. Para ello se ha elegido la modificación del algoritmo de Prim al ser el más eficiente según el caso hasta ahora estudiado.

Para evidenciar si hay una relación entre la distancia y tiempo de un recorrido se han realizado diferentes iteraciones considerando una ruta de 10 ciudades diferentes, las cuales se han generado aleatoriamente y sin semilla. En la Figura 4.19 se muestran los resultados obtenidos para cada una de las 10 iteraciones llevadas a cabo.

Como se puede ver, los resultados obtenidos considerando la variable tiempo o la variable distancia siguen aproximadamente la misma tendencia. Sin embargo, no podemos decir que siempre haya una relación entre ambas curvas, ya que observando en detalle las iteraciones 4 y 5 tienen sus resultados invertidos. Es decir, la iteración 4 consta de un total de casi 650 kilómetros y 800 minutos mientras que la iteración 5 tiene unos 660 kilómetros y sin embargo podría realizarse en 780 minutos.

Debido a esta razón, con el objetivo de realizar más comparaciones entre los diferentes algoritmos y descartar factores relacionados con el azar u otras particularidades de los mismos, tendremos que analizar los resultados tanto en tiempo como en distancia totales.

A continuación, en formato tabla, se van a mostrar los resultados obtenidos, tanto en tiempo como en distancia, al aplicar los diferentes algoritmos bajo diferentes situaciones. Para ello se van a modificar la cantidad de núcleos de población a visitar, la semilla inicial (para que sea totalmente aleatoria) y la cantidad de mutaciones en el caso del algoritmo genético. Todos los algoritmos han sido aplicados al mismo entorno (mismas ciudades) para cada caso para realizar una comparativa fiable entre ellos.

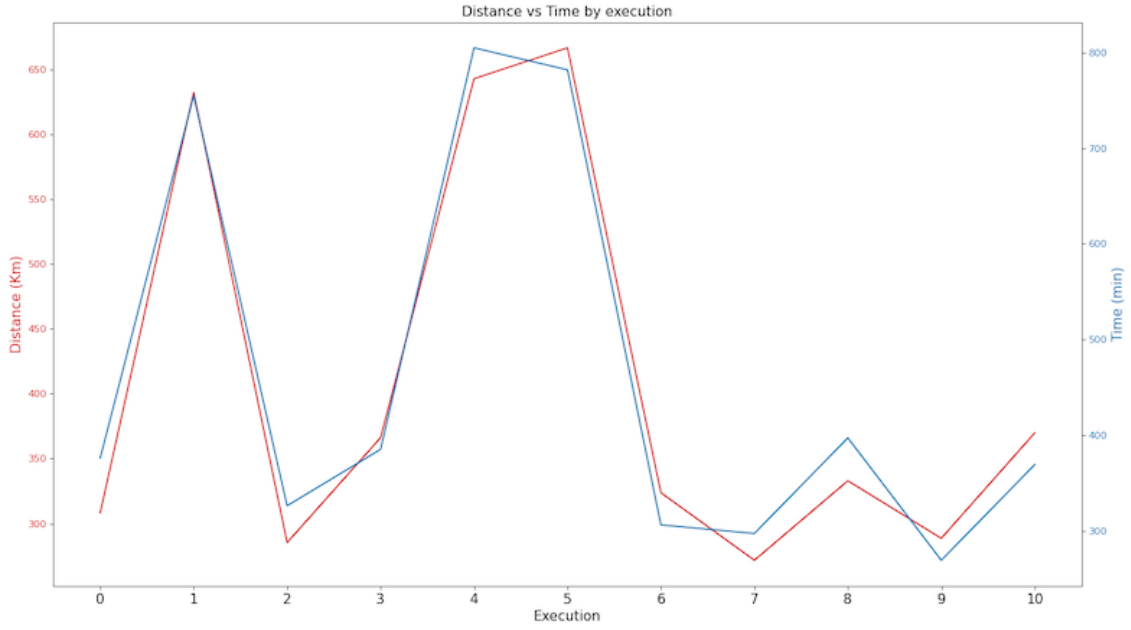


Figura 4.19: Tiempo y distancia total del recorrido según el algoritmo modificado de Prim considerando 10 ciudades a visitar. Fuente: Elaboración propia.

Algoritmo	Distancia total (Km)	Tiempo ejecución (s)	Tiempo total (min)	Tiempo ejecución (s)
Fuerza bruta	285.6	0.85	271	0.82
Prim modificado	286.4	0.90	272	0.78
Genético (500 iter)	285.6	1.89	271	3.32
Genético (2000 iter)	285.6	5.54	271	5.52

Tabla 4.2: Resultados obtenidos según cada algoritmo y tiempo de ejecución considerando rutas de 6 núcleos de población a visitar.

En la Tabla 4.2 se muestran los resultados sobre un caso en el que se visitan 6 núcleos de población. Como se puede observar, cuando el número de puntos de la ruta es relativamente bajo, como es este caso, los resultados obtenidos entre los diferentes algoritmos son muy pequeños. Para este caso el algoritmo de fuerza bruta ha sido el más eficiente, tardando menos de 1 segundo a proveer los resultados. A pesar de ser el más eficiente computacionalmente, está muy cerca el algoritmo modificado de Prim, que tiene unos resultados prácticamente iguales ya que al considerar un kilómetro más o un minuto más sobre una ruta de más de 200 kilómetros o minutos resulta despreciable la diferencia. El caso del algoritmo genético aplicado a esta situación ofrece los mismos resultados que el algoritmo óptimo, sin embargo es menos eficiente computacionalmente, aunque la diferencia

resulta algo totalmente asumible en una situación práctica.

Algoritmo	Distancia total (Km)	Tiempo ejecución (s)	Tiempo total (min)	Tiempo ejecución (s)
Fuerza bruta	282.9	19.4	309	20.50
Prim modificado	342.5	2.17	361	2.10
Genético (500 iter)	309.6	4.64	336	4.39
Genético (2000 iter)	307.1	13.8	310	12.94
Genético (4500 iter)	287.5	25	328	23.19

Tabla 4.3: Resultados obtenidos según cada algoritmo y tiempo de ejecución considerando rutas de 10 núcleos de población a visitar.

En la Tabla 4.3 se muestran los resultados sobre un caso en el que se visitan 10 núcleos de población. Como se puede observar, en este caso sí se evidencia una diferencia, en los diferentes aspectos, entre los diferentes algoritmos. Si el tiempo computacional no es un problema y se tienen que visitar 10 núcleos de población, la mejor opción sigue siendo el algoritmo de fuerza bruta. En caso que se desee un equilibrio entre el tiempo computacional y la distancia o tiempo total del recorrido, el algoritmo genético puede ser una buena opción. Aunque el coste de la ruta será ligeramente superior, siendo dicha diferencia menos de un 1 %, el tiempo que tardará el algoritmo es de sólo 4 segundos. Con dichos resultados se evidencia también que para este caso en concreto un incremento del número de iteraciones del algoritmo genético no incrementa proporcionalmente los resultados, por lo que en caso de necesitar muchas iteraciones sería más adecuado usar el algoritmo de programación dinámica o fuerza bruta.

Este es un claro ejemplo que el algoritmo modificado de Prim puede no proveer buenos resultados según el contexto. En la Figura 4.20 se muestra la representación gráfica de la mejor ruta obtenida usando este algoritmo para la situación de la Tabla 4.3.

Como se puede observar, el trayecto final seleccionado a través de este algoritmo es totalmente ineficiente e ilógico. Algunos de los pasos pueden ser una buena opción según la perspectiva con la que se mire. Sin embargo, otros como la secuencia de visita de los puntos 6, 7, 8 y 9 no tiene ninguna justificación. A pesar que ese sea el orden en el que visitar un punto al siguiente más próximo, a excepción del punto 7 que podría ser un poco más céntrico los otros 3 están situados uno en cada lado de la isla. Al usar esa sucesión para realizar el recorrido, provoca que se pase en múltiples ocasiones por la misma carretera cuando sería posible evitarlo usando otra secuencia con mejores resultados.

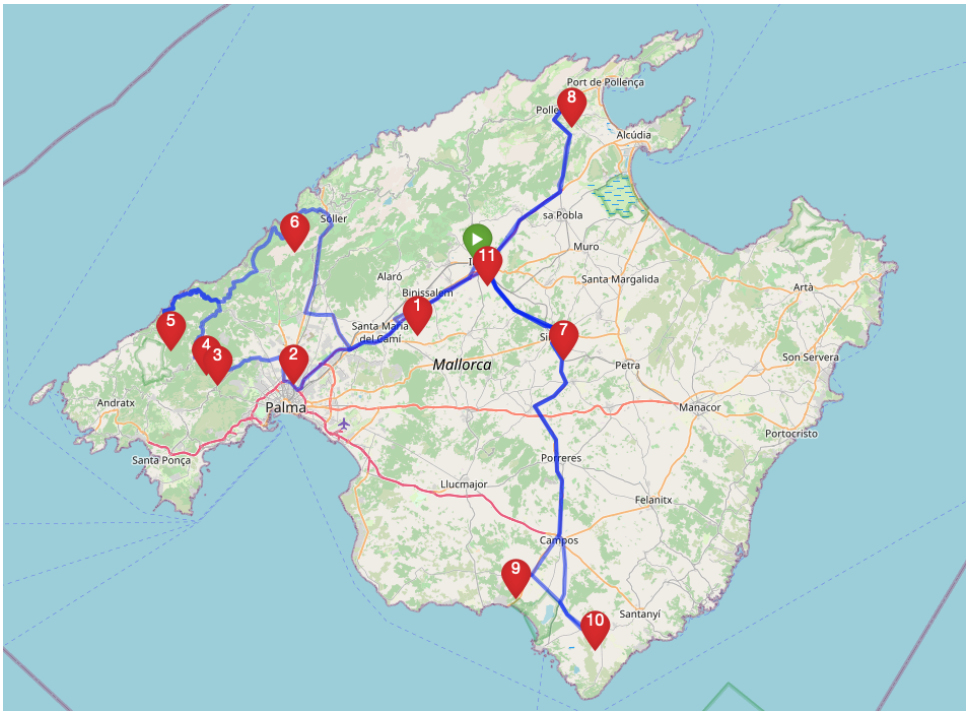


Figura 4.20: Resultado del algoritmo modificado de Prim considerando 10 núcleos de población a visitar. Fuente: Elaboración propia.

Algoritmo	Distancia total (Km)	Tiempo ejecución (s)	Tiempo total (min)	Tiempo ejecución (s)
Prim modificado	449.7	7.14	523	7
Genético (2000 iter)	558.6	28.4	586	32.85
Genético (5000 iter)	548.9	64	595	64
Genético (8000 iter)	528.8	92.6	576	93.18

Tabla 4.4: Resultados obtenidos según cada algoritmo y tiempo de ejecución considerando rutas de 16 núcleos de población a visitar.

En la Figura 4.4 se muestran los resultados obtenidos considerando que se deben visitar 16 núcleos de población. Se ha obviado la ejecución del algoritmo de programación dinámica al ser computacionalmente inviable la resolución a través de este método. En este caso, por contra de los resultados vistos anteriormente, el algoritmo que proporciona mejores resultados es el algoritmo de Prim. Además de ser el más eficiente computacionalmente, es también el que menos kilómetros totales recorre y el que lo hace en un menor intervalo de tiempo.

A partir de los resultados del algoritmo genético y diferentes pruebas que se han rea-

lizado, se puede observar como a pesar que el número de iteraciones sea mayor, eso no siempre proporciona mejores resultados y sí incrementa notablemente el coste computacional. Para que los resultados sean próximos al algoritmo modificado de Prim se deberían realizar muchas más iteraciones para tal cantidad de puntos provocando que el algoritmo sea también totalmente ineficiente para su uso práctico.

4.6.4. Discusión sobre los resultados

A partir de los casos estudiados y entornos en los que se han aplicado los diferentes algoritmos se pueden extraer ciertas conclusiones sobre las ventajas e inconvenientes de cada uno de ellos, así que casos de uso más apropiados.

Algoritmo de programación dinámica o fuerza bruta:

Ventajas

- Algoritmo simple que evalúa todas las posibles soluciones.
- Proporciona una solución óptima.

Inconvenientes

- Necesita conocer todas las distancias entre cada par de puntos.
- Elevado coste computacional, convirtiéndose en inviable su aplicación cuando se tengan que elaborar recorridos de más de 12 puntos.

Algoritmo de Prim modificado:

Ventajas

- Aplicable independientemente de la cantidad de puntos a visitar.
- Poco costoso computacionalmente.
- Proporciona una solución aceptable en la mayoría de los casos.

Inconvenientes

- Sensible a la dispersión de los puntos, puesto que según su posición los resultados obtenidos pueden no ser aceptables.
- Habitualmente no proporcionará la mejor solución.

Algoritmo genético:

Ventajas

- Aplicable independientemente de la cantidad de puntos a visitar.
- En la mayoría de los casos mejora iterativamente
- Las mutaciones y cruces intentan evitar los óptimos locales.

Inconvenientes

- Cuando la cantidad de mutaciones y cruces posibles es muy alta no proporcionan buenos resultados.
- La solución obtenida puede tardar mucho en converger a la solución óptima o incluso no converger.
- Cambio de configuración dependiendo de la cantidad de núcleos de población que se deseen visitar, número de mutaciones, etc.

Además de las ventajas e inconvenientes de cada algoritmo de forma particular, en base a los resultados obtenidos se pueden extraer unas conclusiones frente a las recomendaciones sobre los casos de uso de cada uno.

- Cuando se dispone de pocos puntos a visitar (alrededor de 6 núcleos de población) cualquier algoritmo podría ser válido, recomendando usar el de programación dinámica o fuerza bruta por ofrecer resultados óptimos y ser también el más eficiente.
- Cuando se disponen de alrededor de 10 núcleos de población se recomienda también el algoritmo de programación dinámica excepto si el tiempo de computación fuera un elemento crucial, en cuyo caso se recomendaría el algoritmo genético con 500 o 1000 iteraciones.
- Cuando se disponen de más puntos a visitar se recomienda el algoritmo de Prim modificado. Dado que el algoritmo de fuerza bruta es inviable y se disponen de muchos puntos repartidos en un espacio relativamente reducido, la ineficiencia del algoritmo queda en segundo plano frente a los malos resultados del algoritmo genético.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Conclusiones

Los recorridos realizados por los transportistas es un aspecto fundamental que afecta directamente a los costes totales de una empresa de distribución. El buen servicio o el coste del recorrido son factores a los que se desea encontrar un equilibrio para que los transportistas realicen las rutas de la forma más eficiente posible.

Este trabajo se ha centrado en desarrollar y aplicar diferentes algoritmos para la planificación de rutas de transporte, lo cuál se corresponde con el objetivo específico 4 fijado al inicio del estudio. Para ello se han elegido tres métodos basados en técnicas totalmente diferentes.

El primero de ellos usa técnicas de programación dinámica o fuerza bruta, ya que evalúa todas las posibles soluciones del recorrido para quedarse finalmente con la solución óptima. El segundo ha sido una modificación siguiendo la idea del algoritmo de Prim, el cuál se basa en la teoría de grafos y desde cada punto del recorrido se ha usado el siguiente punto más cercano para elaborar el recorrido. Finalmente se ha usado un algoritmo de programación genética basado en los principios de la evolución biológica combinando los diferentes puntos a visitar como si se tratara de individuos de una población y mutaciones entre ellos.

Para realizar el trabajo, según lo fijado en el objetivo específico 3, previamente se ha realizado un estudio sobre los diferentes factores a tener en cuenta en la creación de rutas. A partir de datos capturados sobre los diferentes núcleos de población de Mallorca, objetivo específico 1, se han creado diferentes subconjuntos de datos aleatorios para simular la ubicación de personas que puedan requerir del servicio de distribución, correspondiente al

objetivo específico 2.

Se ha variado el tamaño de los subconjuntos de datos creados para aplicarlo a los diferentes algoritmos y realizar una comparación entre ellos para conseguir el objetivo específico 5. Se han extraído ciertas ventajas e inconvenientes de cada algoritmo y se ha definido bajo qué situaciones es mejor el uso de uno u otro.

En general se ha podido observar como el algoritmo de programación dinámica o fuerza bruta es el único que proporciona una solución óptima y se recomienda su aplicación cuando se tengan que visitar menos de 10 o 11 núcleos de población. Según los experimentos realizados, en el caso de disponer de más núcleos de población, se recomendaría el uso del algoritmo modificado de Prim puesto que la relación entre los resultados obtenidos y coste computacional es el más equilibrado.

5.2. Líneas de trabajo futuro

En este caso el estudio se ha centrado puramente en la planificación de rutas dados ciertos puntos a visitar que formarán parte del recorrido. Sin embargo hay muchos puntos de la cadena de distribución que podrían optimizarse y formar parte de otros estudios consecutivos a este.

Definir cuál es el punto óptimo para situar el centro de distribución desde el cuál saldrán y finalizarán todos los recorridos es un factor clave que también afectará al coste total del recorrido. En este caso en concreto se ha centrado un punto en el centro de la isla, pero esta posición podría estudiarse a partir de datos reales en función de cuáles son los núcleos de población que realizan más pedidos, la frecuencia o el tiempo de entrega esperado, entre otros.

Otro problema que se puede afrontar según el tipo de productos que se estén distribuyendo es optimizar las rutas en función de ventanas de tiempos. Puede darse el caso que algunos de los puntos que se tengan que visitar de la ruta tengan cierta disponibilidad horaria, provocando que esto altere el recorrido del transportista. En algunas ocasiones se podrá satisfacer esta demanda modificando la jornada laboral del trabajador o reorganizando el recorrido pero en otros casos puede ser un problema sin solución en caso que se tengan que visitar puntos muy distantes en una misma franja horaria muy reducida.

Durante el estudio se ha supuesto que el recorrido se realizaría con un vehículo de capacidad ilimitada y que tuviera las condiciones necesarias para realizar el recorrido. Otra

propuesta de estudio podría ser optimizar las rutas en función de la flota de la que disponga la empresa. Entre esta flota puede haber vehículos más grandes y más pequeños, para los cuáles se debería optimizar el recorrido de tal forma que entre todos ellos satisfagan toda la demanda. Además, podría haber también vehículos con características especiales, como de transporte de animales o refrigerados, pudiéndose combinar entre ellos y ajustando en cada caso el coste de su movilización para optimizar el recorrido final.

Referencias

Agarwal, S., Jain, S., y Kumar, A. (2021). GUI Docker Implementation: Run Common Graphics User Applications Inside Docker Container. *IEEE*, 424–427.

Asl-Najafi, J., Yaghoubi, S., y Noori, S. (2021). Customization of incentive mechanisms based on product life-cycle phases for an efficient product-service supply chain coordination. *Computers in Industry*, *135*, 103582.

Bertazzi, L., Golden, B., y Wang, X. (2015). Min-Max vs. Min-Sum Vehicle Routing: A worst-case analysis. *European Journal of Operational Research*, *240*(2), 372–381.

Brownlee, J. (2018). *How to Generate Random Numbers in Python.* Descargado 10 Mayo 2022, de `https://machinelearningmastery.com/how-to-generate-random-numbers-in-python/`

Chang, Z., y Xue, J. (2022). Enumerations of universal cycles for k -permutations. *Discrete Mathematics*, *345*(9), 112975. Descargado de `https://doi.org/10.1016/j.disc.2022.112975` doi: 10.1016/j.disc.2022.112975

Corona-Gutiérrez, K., Nucamendi-Guillén, S., y Lalla-Ruiz, E. (2022). Vehicle routing with cumulative objectives: a state of the art and analysis. *Computers & Industrial Engineering*, 108054.

Cotter, C. H. (1974). Sines, versines and haversines in nautical astronomy. *Journal of Navigation*, *27*(4), 536–541. doi: 10.1017/S0373463300029337

Docker, I. (2022). *Docker.* Descargado 10 Abril 2022, de `https://www.docker.com`

Euler, L. (1736). Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientarum Imperialis Petropolitanae*, *8*, 128–140.

Frâsinaru, C., y Olariu, E. F. (2021). Generating multi depot vehicle scheduling problems with known optimal tours. *Procedia Computer Science*, *192*, 776–785.

Geofabrik. (2022). *OpenStreetMap.* Descargado 5 Abril 2022, de `http://download.geofabrik.de/europe/spain.html`

Govern de les Illes Balears. (2022). *Catàleg Dades Obertes GOIB.* Descargado 15 March 2022, de `https://catalegdades.caib.cat/Demografia/Nuclis-INE/jiwj-fyx4/data`

Haimovich, M., y Rinnooy Kan, A. H. (1985). Bounds and Heuristics for Capacitated Routing Problems. *Mathematics of Operations Research*, *10*(4), 527–542.

Hardi, R. (2015). Genetic algorithm in solving the TSP on these mineral water. *2015 International Seminar on Intelligent Technology and Its Applications*, 369–372.

Hillier, F. S., y Lieberman, G. J. (2013). *Introducción a la investigación de operaciones* (Vol. 53) (n.º 9).

Holland, J. H. (1975). Adaptation in natural and artificial systems..

Instituto Nacional de Estadística. (s.f.). Descargado 28 Mayo 2022, de `https://www.ine.es/index.htm`

Kuss, D. J., Griffiths, M. D., Binder, J. F., y Street, B. (2013). *Random Number Generation: Types and Techniques* (Tesis Doctoral no publicada). Liberty University.

Lipowski, A., y Lipowska, D. (2012). Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, *391*(6), 2193–2196.

LUCA. (2020). *¿Qué es Python?* Descargado 10 Abril 2022, de `https://web.archive.org/web/20200224120525/https://luca-d3.com/es/data-speaks/diccionario-tecnologico/python-lenguaje`

Ma, Z. J., Wu, Y., y Dai, Y. (2017). A combined order selection and time-dependent vehicle routing problem with time widows for perishable product delivery. *Computers and Industrial Engineering*, *114*(September), 101–113.

Mandziuk, J. (2019). New Shades of the Vehicle Routing Problem: Emerging Problem Formulations and Computational Intelligence Solution Methods. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *3*(3), 230–244.

OSRM, P. (2012). *Osrm api documentation.* Descargado 10 March 2022, de `http://project-osrm.org/docs/v5.7.0/api/?language=Python#general-options`

Prim, R. C. (1957). Shortest Connection Networks And Some Generalizations. *Bell System Technical Journal*, *36*(6), 1389–1401.

Puchades Cortés, V., Mula Bru, J., y Rodríguez Villalobos, A. (2008). Aplicación de la teoría de grafos para mejorar la planificación de rutas de trabajo de una empresa del sector de la distribución automática. *Revista de Metodos Cuantitativos para la Economia y la Empresa*, *6*(6), 7–22.

Python Software Foundation. (2022). *Python.* Descargado 10 Abril 2022, de `https://www.python.org`

Redi, A. A., Maghfiroh, M. F., y Yu, V. F. (2014). An improved variable neighborhood search for the open vehicle routing problem with time windows. *IEEE International Conference on Industrial Engineering and Engineering Management*, 1641–1645.

Ruiz, F. L., y Landín, G. A. (2003). Nuevos Algoritmos en el Problema de Transporte. *V Congreso de Ingeniería de Organización*(Valladolid-Burgos, 4-5 Septiembre 2003), 4–5.

Taha, H. A. (2004). *Investigación de operaciones* (7a ed.).

Toyoda, K., Okamoto, T., y Koakutsu, S. (2017). An optimal routing search method on the network routing problem using the sequential minimal optimization. *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan, SICE 2017*, *2017-Novem*, 805–810.

Véliz, D. (2022). *El 47,43 % de los españoles compra online de forma mensual... y reiterada.* Descargado de `https://marketing4ecommerce.net/el-4743-de-los-espanoles-compra-online-de-forma-mensual-y-reiterada/`