Trabajo Fin de Máster

Máster en Ingeniería Industrial

Diseño y optimización de una distribución coordinada de suministros en colegios públicos de Sevilla

Autor: Miguel Novoa Contreras

Tutor: Luis Miguel Romero Pérez

Dpto. Ingeniería y Ciencia de los Materiales y del Transporte
Escuela Técnica Superior de Ingeniería

Sevilla, 2018

Trabajo Fin de Máster
Máster en Ingeniería Industrial

Diseño y optimización de una distribución coordinada de suministros en colegios públicos de Sevilla

Autor:
Miguel Novoa Contreras

Tutor:
Luis Miguel Romero Pérez

Área de conocimiento: Ingeniería e Infraestructura de los Transportes
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Trabajo Fin de Máster: Diseño y optimización de una distribución coordinada de suministros en colegios públicos de Sevilla

Autor: Miguel Novoa Contreras

Tutor: Luis Miguel Romero Pérez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

A mi familia por su apoyo.

A mi tutor Luis Miguel Romero por su colaboración.

Resumen

Es tan amplio el abanico de posibilidades, de situaciones logísticas y de distribución de mercancías, y cada una de estas posibilidades tan distintas entre sí, que los problemas de optimización de ruta como el del problema del viajante, son muy importantes en la actualidad. Es tal el ahorro y la eficiencia que se pueden alcanzar con estos métodos, que con apenas inversión se puede conseguir unos niveles de competitividad muy altos, algo que es muy importante en este sector.

Este proyecto trata de diseñar y optimizar la ruta de reparto de material escolar a todos los centros educativos localizados en la ciudad de Sevilla. Como se explicará durante el proyecto, la idea es que desde un mismo centro de distribución se reparta todo el material necesario para los centros educativos públicos de Sevilla. De esta manera se disminuirá los kilómetros recorridos y el número de transportes que van a los centros educativos, consiguiendo también un menor tráfico y una mayor seguridad cerca de los centros educativos. Para ello, se ha investigado y estudiado los problemas de optimización de rutas, y cuales son la mejor forma de resolución.

A lo largo del proyecto, se presentará una base teórica sobre los problemas de optimización de rutas y posibles métodos de resolución. Además, se aplicarán algunos de los métodos explicados, cuyos resultados serán analizados y comparados. Se finalizará con la explicación y el análisis de la mejor ruta encontrada, y con posibles mejoras que aplicar.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

TSP	Problema del agente viajero
ACO	Colonia de hormigas
VRP	Problema de enrutamiento de vehículos
ADN	Ácido desoxirribonucleico
APP	Aplicación

1 INTRODUCCIÓN

1.1 Objetivo del proyecto

En el presente proyecto se va a diseñar y optimizar la distribución del material escolar en todos los centros educativos públicos de la ciudad de Sevilla. Para llevar a cabo el diseño de la distribución, se van a implementar distintos algoritmos en el programa Matlab. Estos algoritmos proporcionarán diferentes rutas, las cuales reflejarán el orden de reparto en los centros educativos. Se implementarán varios algoritmos para poder comparar sus soluciones, y analizar cuál de estas es la solución idónea a implantar en la distribución.

Es importante destacar que, en esta idea la distribución del material escolar se llevará a cabo por un distribuidor único, a diferencia de como se está realizando actualmente con varios distribuidores, como se comentará a lo largo del proyecto.

1.2 Antecedentes del proyecto

Actualmente, este reparto de material es realizado por diferentes empresas, y cada una de estas empresas a diferentes centros educativos. Esta forma de distribución del material escolar es muy ineficiente y se obtienen muchas consecuencias negativas. Al contar cada centro educativo con diferentes empresas de aprovisionamiento, y dependiendo del material necesario la distribución la realiza una empresa u otra, se origina un excesivo tráfico de mercancías alrededor de los centros educativos.

Esta forma de distribución no solo origina un excesivo tráfico, sino que provoca otras consecuencias nada deseables. El coste y el consumo de combustible de toda la distribución también son muy elevados, debido al gran número de kilómetros que tienen que realizar el conjunto de las empresas para completar el reparto de todo el material escolar a todos los centros educativos. También provoca situaciones mejorables como en el ámbito de la seguridad vial y del medio ambiente, donde al existir tanto tráfico de mercancías se producen un mayor riesgo de accidente y unas elevadas emisiones de CO_2 en el entorno de los centros educativos.

Es importante disminuir ese excesivo tráfico que existe con la distribución actual. El excesivo tráfico de mercancías es un tema muy notable y de mucho debate en la actualidad. La distribución de mercancías es el principal culpable de la congestión urbana y por lo tanto de los perjuicios de esta. Son numerosas las medidas que se están tomando a nivel europeo y nacional para intentar reducir este tráfico. De entre los aspectos negativos que tiene el excesivo tráfico de mercancías en la ciudad se pueden destacar:

- Ambiental: Contribución al efecto invernadero y a la existencia de una mala calidad del aire. Así como excesivas situaciones de intenso ruido y vibraciones.

- Seguridad: Alta tasa de accidentes por el excesivo tráfico, y la sensación de peligro al circular peatones y pequeños vehículos junto a grandes camiones pesados.

- Costes de operación: Al haber una gran congestión del tráfico, el tiempo y el coste destinado a la distribución se incrementa, lo que al final repercute en el coste final de los productos.

- Operaciones urbanas: Existe gran cantidad de vehículos destinados al transporte de mercancías que tienen que cargar y descargar en la vía pública, lo que disminuye el espacio de circulación para peatones y para el resto de vehículos. Además, este espacio de circulación se ve reducido cuando los vehículos se estacionan ilegalmente para su carga y descarga. En este sentido destacar que el tráfico de mercancías supone el 75% de las operaciones de carga y descarga, mientras que el 25% restante es para uso doméstico o para el sector servicios.

En Muñuzuri,2003, se puede ampliar la información sobre los impactos negativos de la congestión del tráfico tanto para la logística urbana de mercancías como para la sociedad. Así como se pueden observar algunas soluciones a este problema como el uso del carril bus, repartos nocturnos o repartos conjuntos desde un mismo centro de distribución.

Toda esta problemática adquiere un valor todavía más importante, si estas situaciones se dan en el entorno de los centros educativos, donde se debe tener especial cuidado por la presencia de un gran número de niños pequeños.

Para evitar estas situaciones, en este proyecto se estudia la posibilidad de que un mismo distribuidor se encargue de todo el aprovisionamiento de material escolar que necesitan los centros educativos, esto provocará una disminución del número de kilómetros y de vehículos destinados a dicha causa.

Con esta propuesta se disminuirá dicho tráfico y se aumentará la seguridad alrededor de los centros educativos de la ciudad. Además, se disminuirá un coste y un número importante de kilómetros recorridos. Otro aspecto importante que se mejorará, será la disminución de las emisiones de CO_2 en el entorno de los centros educativos de la ciudad debido al menor tráfico.

La idea general es la de recibir todo el material escolar en un punto de la ciudad de Sevilla, y desde ese punto distribuirlo a todos los centros educativos. Para que esa distribución sea lo más óptima posible se han diseñado las posibles rutas que se pueden seguir. En el capítulo de resultados experimentales se presentarán las distintas rutas y las elegidas como mejores rutas a implantar. La siguiente imagen puede resumir el cambio que se va a estudiar en este proyecto de pasar de una distribución individual a una distribución coordinada.

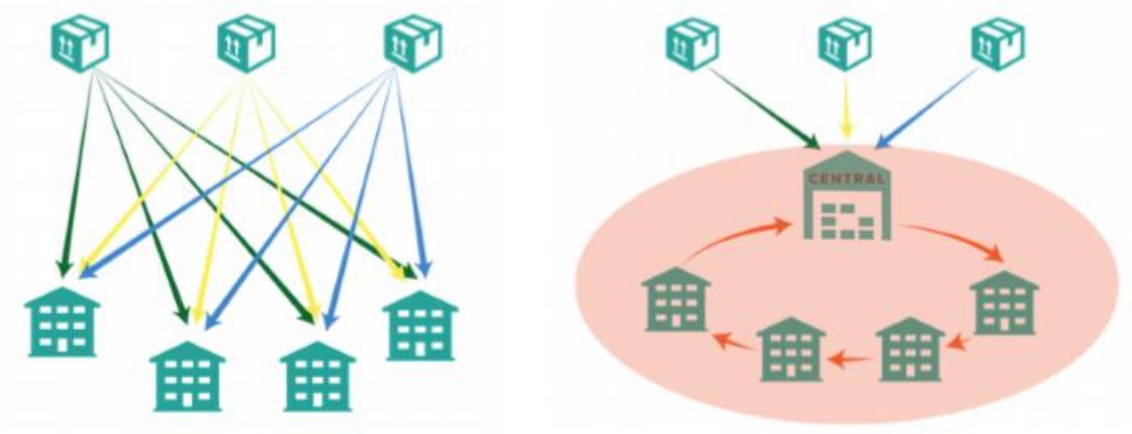

Figura 1. Distribución individual a la izquierda y distribución coordinada a la derecha. (Junta de Andalucía. Consejería de Educación.)

Como bien se ha comentado, el objetivo principal es implementar la ruta óptima desde un centro de distribución, que completen el reparto a todos los centros educativos públicos, para así disminuir el número de kilómetros y el número de transportes. Para esto es necesario hacer un inventario de todos los destinos, es decir, de todos los centros educativos a los que se entregan los bienes.

Con esta propuesta se eliminan las entregas individuales, lo que ocasiona 4 principales beneficios que se pueden resumir en:

- Medio ambiente: Se disminuye las emisiones de CO2 tanto en la ciudad como en el entorno de los centros educativos.

- Seguridad del tráfico: Al disminuir el tráfico de camiones, se aumentará la seguridad del tráfico en la cercanía de los centros educativos.

- Ambiente de trabajo: Con el diseño de las rutas se ahorra tiempos de entrega, lo que aumenta la competitividad y mejora el clima de trabajo. Además, se pueden fijar los mismos días y horas para la entrega de la mercancía.

- Coordinación municipal: Se podrá realizar con mayor facilidad un inventario del material escolar. Con una distribución coordinada se puede llevar un mayor control del material que se necesita en los centros educativos. Además, con este método se podrá solventar de manera más eficiente y eficaz cualquier contratiempo que ocurra en el reparto del material.

Esta idea de distribución coordinada se ha implantado en algunas regiones de Suecia, donde los resultados fueron impresionantemente buenos. Los datos que se obtuvieron tras la implantación de esta idea en esas regiones hablan de una mejora importante, tanto en kilómetros recorridos como en emisiones de CO2.

- Kilómetros recorridos: Se ahorró de media 2.800 kilómetros en el transporte de mercancías.
- Envíos: Se disminuyeron los envíos en un 62%.
- Emisiones de CO2: Se produjo una gran disminución de emisiones de CO2, reduciéndose en un 73% con respecto al modelo anterior de distribución.

Viendo los resultados que se obtuvieron en este proyecto sueco, se puede pensar que es una muy buena medida a implantar en el modelo de distribución de material escolar en los centros educativos públicos de Sevilla.

2 FUNDAMENTOS TEÓRICOS

2.1 Problema del agente viajero

2.1.1 Introducción al problema del agente viajero

A lo largo de todo este proyecto, se va a acometer el estudio y el análisis de la ruta que debe seguir el reparto de material escolar a todos los centros educativos de la ciudad de Sevilla. El objetivo principal es conseguir que el tiempo que se necesite para llevar a cabo dicha ruta sea el menor posible. Es importante comentar que en el reparto de material escolar se deben visitar todos los centros educativos, y no repetir el paso por ningún centro, ya que esto aumentaría el tiempo del reparto.

La definición del objetivo de este proyecto, es muy similar a la de uno de los grandes problemas de la optimización combinatoria como es el problema del viajante. Debido a esta gran similitud, en este proyecto se va a estudiar todo lo relacionado con el problema del viajante, como sus características, tipos de problemas, algoritmos de resolución, etc. Por ello se pretende que, a través del problema del viajante se dé solución al problema del reparto de material escolar antes explicado.

En primer lugar, y antes de introducir que es el problema del viajante, es necesario saber que tipos de problemas de rutas existen en la actualidad. Existen dos grandes grupos de problemas de rutas, los problemas por vértices y los problemas por arcos.

El primero de ellos tiene como objetivo visitar los vértices, es decir, visitar los nodos de un grafo, estos nodos pueden ser visitados por un vehículo, TSP, o por varios vehículos, VRP (Vehicle Routing Problem).

El segundo grupo de problemas se centra en pasar por las aristas de un grafo, en este grupo se puede encontrar un tipo de problemas parecido al TSP, y es el CPP (Chinese Postman Problem). En este, en vez de recorrer todos los nodos, se tienen que recorrer todas las aristas una sola vez.

Por profundizar un poco más en el mundo de los problemas por vértices, como es el caso del problema que se va a tratar en este trabajo, se pueden dar distintos tipos de problemas como los presentados a continuación:

1. Max-TSP → Su objetivo es contrario al caso estudiado, es decir, se busca la ruta con coste máximo.

2. TSP con cuello de botella → Busca una ruta, de tal forma que se minimice el mayor coste de las aristas que pertenecen a dicha ruta.

3. TSP → El problema tratado en este trabajo, busca el mínimo coste de una ruta que visita una vez todos los nodos.

4. TSP generalizado → Los nodos están agrupados y el objetivo de este problema es la búsqueda de una ruta de coste mínimo que visite una sola vez un nodo de cada grupo.

El problema del agente viajero, o problema del viajante, es comúnmente conocido bajo las siglas TSP (Travelling Salesman Problem). El objetivo de este problema es encontrar la ruta óptima que debe realizar el agente, partiendo desde el origen, recorriendo todos los puntos y volviendo a dicho origen. Esta ruta óptima debe ser la de menor distancia o menor tiempo. Esta ruta que contiene todos los nodos es conocida como ciclo hamiltoniano, cuyo nombre es en honor a William Rowan Hamilton.

A priori, parece que el problema es sencillo, pero si se tiene en cuenta el número de nodos, las distancias o los tiempos entre ellos y demás restricciones que se pueden incorporar, el problema adquiere una dificultad elevada. Es decir, se está ante un problema que no es difícil de modelar, pero sí lo es de resolver.

En la actualidad, se utilizan algoritmos que proporcionan una solución aproximada en un tiempo razonable. Un algoritmo que proporciona la solución exacta, es comprobar todas las posibles soluciones del problema, pero esto es inviable debido al tiempo computacional requerido, ya que, el tiempo necesario para encontrar la solución aumenta de manera exponencial a medida que aumenta el tamaño del problema. Es decir, para un problema con n nodos, donde los tiempos son simétricos (tiempo de A-B es el mismo que el tiempo de B-A) existen $(n-1!)/2$ rutas posibles. Por ello este tipo de problemas se enmarca dentro de los problemas NP-HARD, como se explica en Bernal et al, 2015.

Este tipo de problema tiene muchas aplicaciones que se explicarán posteriormente, pero una de las aplicaciones más importantes es la logística y el transporte de mercancías, debido a la alta competitividad que tienen las empresas del sector. Por ello, se utilizan nuevas herramientas de gestión, para así optimizar los recursos y poder obtener mayores márgenes de beneficio.

Una de estas nuevas herramientas es la resolución de los problemas de optimización combinatoria, entre los que se incluye el TSP, mediante algoritmos o heurísticas. Para su resolución es habitual la ejecución de varios algoritmos y estrategias, para valorar sus soluciones y ver cuál es la mejor a implementar. Es decir, dependiendo del tiempo computacional, de la calidad de la solución, de la simplicidad o del cumplimiento de las restricciones.

Figura 2. Ejemplo ruta TSP (López et al, 2014)

2.1.2 Fundamentos históricos

En esta sección se va a detallar como se ha ido formulando el problema del viajante desde su origen, el cual no está del todo claro, hasta la actualidad. Para profundizar más en el siguiente recorrido histórico, sobre el proceso de definición del problema del viajante y de las diferentes metodologías para su resolución, se puede visitar multitud de referencias bibliográficas, como por ejemplo K. Menger, 1931; Dantzig et al, 1954; Gomory, 1958; entre otras muchas.

El primer gran avance en este campo, se debe gracias a la aparición de los circuitos hamiltonianos, nombrados anteriormente en este trabajo. Aunque, antes de conocer estos circuitos con este nombre, Euler y Vandermonde investigaron estos circuitos para dar solución al problema del juego del salto del caballo, ambos presentaron una solución a dicho problema, la de Leonhard Euler, de 1735, está recogida en Núñez Valdés et al,2004 y la de Alexandre-Théophile Vandermonde, en su artículo *"Remarques sûr des problèmes de situation"* publicado en 1771 por la Academia Real de las Ciencias de París.

(Vandermonde, 1771)

Ambos investigadores trataron de encontrar un circuito, el cual tiene que simular el recorrido de un caballo, que debe pasar una sola vez por cada uno de los 64 cuadrados del tablero de ajedrez. Aunque, como se ha dicho al principio el primer gran avance no fue el de estos dos investigadores, fue la aparición de los circuitos hamiltonianos. En 1857 William Donald Hamilton desarrolló un juego cuyo objetivo era obtener un circuito que pase por las aristas de un dodecaedro, de manera que se visite cada una de las 20 esquinas de este poliedro una sola vez, y en el que el punto de partida sea también el punto final del recorrido. El nombre de este recorrido se da en honor a Hamilton, pero 2 años antes, el matemático británico Thomas Penyngton Kirkman ya planteó el mismo problema en un documento llamado *"dado un grafo de un poliedro, ¿existe un ciclo que pase una vez por cada vértice?"*.

(Kirkman, 1855)

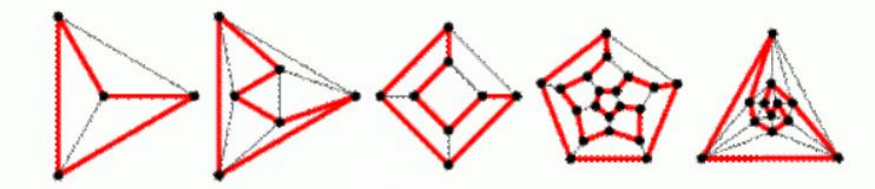

Figura 3. Juego Icosaédrico. (Aranda et al, 2007)

A pesar de que estos fueron los primeros planteamientos del problema, el primer libro donde aparece explícitamente el problema del viajante fue impreso en Alemania en 1832 y cuyo nombre fue "El viajante de comercio: cómo debe ser y qué debe hacer para conseguir comisiones y triunfar en el negocio. Por un viajante de comercio veterano".

(B. Fr. Voigt, 1832)

Es cierto que en al final del libro se empieza a hablar del problema, donde argumenta que con una buena ruta se puede ahorrar tiempo y se pueden visitar más ciudades sin repetirlas, pero no habla de ningún modelo o algoritmo matemático.

Hay muchos rumores sobre a quién atribuirle el nombre de TSP (Travelling Salesman Problem), lo único que se sabe a ciencia cierta, es que Merril Flood en 1984 relató que Albert William Tucker en 1937 le habló sobre un matemático americano llamado, Hassler Whitney, que sobre los años 30 ya le dio nombre al problema, TSP. Durante estos años 30, se empezó a investigar y a trabajar mucho en este problema, el propio Merril Flood comenzó la búsqueda del mejor camino que debía trazar un autobús escolar.

En las mismas fechas que Merril Flood, en Austria, el matemático Karl Menger estaba investigando en un problema que, denominó como el problema del mensajero, cuyo objetivo es hallar la ruta más corta que uniera un grupo de nodos. Es decir, la solución del problema no busca un ciclo, si no un camino que contenga todos esos nodos. Aparte de definir dicho problema, este matemático austriaco, dio con una propiedad del TSP, dicha propiedad se caracterizaba por decir que el TSP era un problema NP-HARD, esta clase de problema se explica posteriormente en este trabajo. No solo logró llegar a estas dos investigaciones tan importantes, si no que elaboró un algoritmo de resolución, el cual actualmente, es conocido por el algoritmo del vecino más próximo.

A partir de los años 50, el interés en el problema del viajante se incrementó, y muchos científicos e investigadores dieron una gran popularidad a este problema. Los científicos asemejaban el TSP al problema de asignación y al problema del transporte, aspecto que también influyó en el incremento de popularidad. George Dantzig fue otro matemático con gran influencia en este campo, en 1947 presentó el método simplex para resolver un problema de programación lineal, años más tarde se incorporó a la corporación RAND, un centro intelectual cuyo auge coincidió con la incorporación de Merril Flood, nombrado anteriormente. A partir del método simplex desarrollado por Dantzig, en 1953, se desarrollaron nuevos algoritmos basados en dicho método, para resolver los famosos problemas de asignación y del transporte. Uno de estos algoritmos fue el método húngaro, presentado en 1955 por Harold Kuhn.

Un hito muy importante para el campo de la optimización combinatoria fue la publicación de "Soluciones de un problema del viajante de gran tamaño", elaborado en 1954 por George Dantzig, Delbert Ray Fulkerson y Selmer Martin Johnson. En este artículo se resolvía un problema del viajante para 49 ciudades. En el mismo año Dantzig y Fulkerson publicaron otro artículo de gran relevancia para el sector, en él, se resolvía un problema cuyo objetivo era buscar el número óptimo de buques necesarios para cumplir ciertas restricciones horarias.

En cuanto al primer artículo, tuvo una gran repercusión, ya que, fue la primera vez que se resolvió un problema de tamaño aceptable, estos investigadores conseguían resolverlo mediante técnicas de programación lineal, pero no solo eso, dedujeron que, si encontraban una ruta cercana a la solución óptima, a esta ruta se le podía añadir restricciones para llegar al óptimo del problema. Dicho método, se conoció como el método de los planos de cortes. Al principio de la investigación, predijeron que para las 49 ciudades serían necesarios unos 25 cortes, y estuvieron muy acertados ya que, finalmente fueron 26 los cortes. Se puede decir que fueron grandes años en el desarrollo y en la investigación de estos problemas.

Esta interacción de grande investigadores y científicos dio lugar años más tarde a la elaboración de un algoritmo primal-dual para la resolución de problemas lineales, para el desarrollo de dicho algoritmo también fue muy importante Lester Randolph Ford, matemático americano.

Volviendo al método anteriormente mencionado, el de los planos de cortes, a lo largo de los años fue evolucionando hasta el conocido actualmente como el método de ramificación y acotación, o método Branch and Bound. Un algoritmo que se elaboró a partir del de los planos de cortes, fue el método de Gomory, del propio Ralph Gomory en 1958.

A partir de todos estos hechos, fueron apareciendo nuevos algoritmos y el número de ciudades a resolver cada vez era mayor, algunos investigadores pensaron en la aplicación de la programación dinámica a estos problemas, pero la programación dinámica es aplicable a problemas de un tamaño reducido, ya que, para grandes problemas, la elevada cantidad de condiciones haría ineficiente su resolución. A pesar de esto, M. Held y R. Karp, aplicaron la programación dinámica.

En los años 80, se consiguió resolver un problema de 2.392 ciudades, gracias a los investigadores Grotschel, Padberg y Rinaldi, entre otros.

En la década de 1990, se desarrolló un programa, llamado Concorde, y con el cual se logró resolver un problema de 85.900 ciudades en 2006, gracias a W. Cook y otros investigadores, el problema que resolvieron fue para el diseño de un microchip. Es impresionante la evolución del tamaño del problema desde sus orígenes hasta la actualidad. A forma de resumen, a continuación, se puede observar la evolución del tamaño de los problemas resueltos, desde el ya comentado problema de las 49 ciudades de George Dantzig, Delbert Ray Fulkerson y Selmer Martin Johnson, hasta este último de 85.900 ciudades.

1954	G. Dantzig, R. Fulkerson, S. Johnson	49 ciudades
1971	M. Held, R. Karp	57 ciudades
1971	M. Held, R. Karp	64 ciudades
1975	P.M. Camerini, L. Fratta, F. Maffioli	67 ciudades
1975	P. Miliotis	80 ciudades
1977	Martin Grötschel	120 ciudades
1980	H. Crowder, M.W. Padberg	318 ciudades
1987	M. Padberg, Rinaldi	532 ciudades
1987	Grötschel, O. Holland	666 ciudades
1987	M. Padberg, Rinaldi	1002 ciudades
1987	M. Padberg, Rinaldi	2392 ciudades
2005	W. Cook, Espinoza, Goycoolea	33810 ciudades
2006	W. Cook	85900 ciudades

Tabla 1. Evolución del tamaño del TSP. (Applegate et al, 2006)

2.1.3 Formulación y variantes

Como ya se ha comentado anteriormente, el problema del agente viajero es fácil de modelar o formular, pero su resolución no es tan sencilla.

Cabe destacar que el modelo que se presenta en esta sección es una formulación básica del problema, a esta, se le puede añadir las restricciones que cada problema en particular necesite. Antes de mostrar el modelo del problema, se va a presentar y detallar cada una de las variables y de los datos del problema:

- $C = (c_{ij}) \rightarrow$ Matriz de costes del problema, en la mayoría, suelen ser tiempos o distancias entre todos los puntos a visitar.

- $x_{ij} \rightarrow$ Variable binaria, será 1 si se visita el nodo j desde el nodo i.

- $n \rightarrow$ Número de nodos a visitar.

La función objetivo del problema será el coste total de visitar todos los nodos, es decir, el sumatorio de los costes de aquellos x_{ij} con valor 1. En cuanto a las restricciones, deben garantizar que cada nodo sea visitado una sola vez, y que este a su vez tenga solo un camino de salida hacia un nodo no visitado. Por todo esto, este tipo de problemas se asemeja a un problema de asignación, con la diferencia de que en el problema del viajante no pueden existir subcircuitos. Es decir, solo puede existir una ruta que recorra todos los nodos. A continuación, se observa dicho modelo:

$$Min\ Z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$sujeto\ a: \quad \sum_{i=1}^{n} x_{ij} = 1, \qquad \forall\, j = 1,2,\dots.n$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad \forall\, i = 1,2,\dots.n$$

$$x_{ij} \in \{0|1\}, \qquad \forall\, i,j = 1,2,\dots.n$$

Este es el modelo del problema presentado, sin embargo, como se ha explicado anteriormente es necesario evitar posibles subcircuitos, por ello se añaden un par de restricciones, que son las siguientes:

$$u_i - u_j + n * x_{ij} = 1, \qquad \forall\, 2 \leq i \neq j \leq n$$

$$u_i \geq 0, \qquad \forall\ 1 \leq i \leq n$$

En estas dos restricciones, aparecen unas variables ficticias, u_i, las cuales están asociadas a cada nodo. Se puede comprobar fácilmente en Bernal et al, 2015, como estas restricciones impiden la formación de ciclos.

Una vez explicado el modelo del problema del viajante, se va a comentar las distintas variantes que puede presentar dicho problema, las variantes dependen del tipo de matriz de costes.

- Problema del viajante simétrico → Este problema se caracteriza porque la matriz de costes es simétrica. Es decir, el coste de ir desde el nodo i hasta el nodo j, es el mismo que el coste de ir desde el nodo j hasta el nodo i.

$$c_{ij} = c_{ji} \qquad \forall (i,j)$$

- Problema del viajante asimétrico → Es el caso del problema tratado en este trabajo, ya que el coste de ir desde i hasta j, es distinto del coste de ir desde j hasta i.

$$c_{ij} \neq c_{ji} \qquad \forall (i,j)$$

- Problema del viajante triangular → En estos problemas se cumplen para todas las aristas del grafo que:

$$c_{ik} \leq c_{ij} + c_{jk}$$

2.1.4 Problema NP-HARD

En anteriores secciones, se ha catalogado el problema del viajante como NP-HARD; en los años 30 ya se había etiquetado el problema como NP-HARD. Para entender el problema del viajante, hay que entender que características presentan este tipo de problemas y en que se distinguen de otros. Por lo que, en este capítulo, se va a repasar un poco sobre las distintas clases de problemas y explicar en profundidad los catalogados como NP-HARD.

Antes de entrar en detalle en este tipo de problema es necesario conocer la definición de problema. Se conoce como problema a una cuestión, en la que existen un conjunto de datos de partidas, los cuales pueden permanecer fijos o variar a lo largo de este. En la resolución del problema se deben realizar una serie de transformaciones y operaciones, hasta llegar a una o varias soluciones que deben cumplir unos criterios establecidos previamente.

Hay dos grandes clases de problemas, los computacionales o los de decisión; el primero se caracteriza por ser una relación entre el conjunto de datos de entrada, también llamados instancias, y el conjunto de soluciones que se dan en la salida. El segundo tipo, el de decisión, es un problema donde las soluciones son del tipo sí o no. Ambos tipos de problemas son complementarios, ya que, en un problema computacional puede haber varios problemas de decisión.

En el problema del viajante, es muy importante conocer lo que es el término instancia, una instancia hace referencia a los valores que toman los parámetros del problema.

Como se ha comentado al principio de la sección, para la resolución del problema es necesario una serie de transformaciones y operaciones, por ello es muy importante la aplicación de algoritmos. Un algoritmo es un conjunto de procedimientos, que deben poder aplicarse a cualquier instancia del problema, llegando a una solución concreta. Para la elección de un algoritmo u otro hay que tener en cuenta varios factores, el más importante es que llegue a la mejor solución posible, pero también debe ser eficiente y rápido.

Dependiendo de la complejidad del problema, se pueden distinguir diferentes clases:

- Clase P: Un problema es de clase P, si se puede resolver en un tiempo polinómico, a partir de una entrada, por una máquina de Turing determinista. Una máquina de Turing, es similar a un algoritmo, realiza una lectura de una entrada de datos y genera una salida. Resolver un problema en tiempo polinómico es cuando el tiempo que tarda un algoritmo en ejecutarse es menor que un valor que depende del tamaño de la entrada, utilizándose procedimientos polinómicos.

- Clase NP: Un problema pertenece a esta clase de complejidad, si se puede resolver en tiempo polinómico por una máquina de Turing no determinista.

En la actualidad, hay una gran interrogación sobre si estas dos clases de problemas se pueden diferenciar totalmente o no. Es decir, la pregunta es si P=NP. Para muchos investigadores, hay un conjunto de problemas NP que, sí son totalmente diferentes a los de clase P, y eso son los problemas NP-COMPLETO. Este conjunto de problemas de decisión se caracteriza por ser los que presentan mayor dificultad de resolución, por lo que son los que más difieren de los problemas de clase P. Se puede ampliar la información sobre el origen de estos problemas en Cook, 1971, aunque Stephen Cook no se refirió a ellos con el mismo nombre.

Como se bien se ha comentado al principio de la sección, el problema del viajante se enmarca en la clase de problemas NP-HARD. Esta clase de problemas, se caracteriza por ser, como mínimo, tan compleja como la clase NP-COMPETO. Es más, los problemas NP-COMPLETO, representan la intersección entre NP-HARD y NP.

Para entender un poco mejor esta clasificación, se puede observar la siguiente imagen:

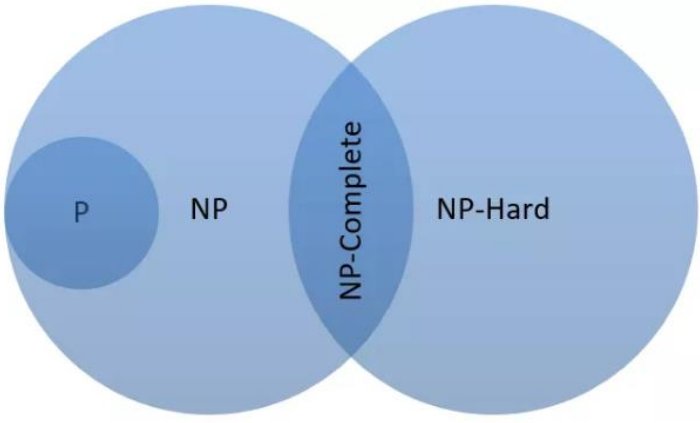

Figura 4. Clasificación de problemas

Para acabar, durante toda esta sección del trabajo se ha enmarcado al problema del viajante en los problemas NP-HARD, pero debe destacarse que pertenece a la parte de la intersección entre NP y NP-HARD, es decir, NP-COMPLETO. Algunos investigadores catalogan al problema del viajante, como NP-HARD y otros como NP-COMPLETO.

2.1.5 Aplicaciones

El problema del viajante cuenta con numerosas y variadas aplicaciones en la actualidad, ya que, es una buena herramienta para la búsqueda de la máxima eficiencia. Como se ha comentado en la introducción, para pequeñas empresas que tienen unos recursos muy limitados, puede ser una herramienta muy a tener en cuenta, debido a que se puede obtener una gran ventaja competitiva con apenas incrementar los recursos necesarios por la empresa.

A continuación, se va presentar algunas de las aplicaciones más destacadas del TSP:

- Logística: Campo donde la utilización del problema del viajante es muy necesaria y muy recomendable. Realizar una ruta de reparto óptima, y que minimice tiempos y costes, influirá positivamente en la empresa, y puede haber grandes diferencias entre trazar dicha ruta y trazar una cualquiera. Pues gracias al TSP se puede obtener dicha ruta óptima o una cercana a la óptima, dependiendo del tamaño del problema y el algoritmo utilizado. Por ejemplo, en este trabajo, el cual trata del reparto de material escolar, conviene la utilización de esta herramienta.
- Circuitos electrónicos: La utilización del TSP en este campo ya comenzó en los años 70, cuando Lin y Kernighan empezaron a tratarlo. Gracias a la resolución de este problema, se puede llegar a ahorrar en tiempo de taladrado de las placas, a taladrar en un orden correcto y en la cantidad de cable utilizado. Para profundizar más en este tema consultar Lin & Kernighan, 1973.
- Industria: Quizás un campo donde no se utiliza tanto, pero puede llegar a ser muy importante la resolución del TSP. Para el conjunto de tareas que se debe llevar a cabo en una industria, la aplicación del TSP puede llegar a ser muy interesante. Las actividades a realizar serían los nodos del problema, y la matriz de costes sería el tiempo que transcurre entre que se acaba una actividad y empieza otra.
- Biología: Mundo novedoso en la aplicación del TSP, pero en el que se están haciendo grandes avances. Una aplicación real en este campo puede ser la secuenciación de ADN.
- Astronomía: Para la optimización de rutas por cuerpos celestes o para establecer un orden de observaciones en telescopios.

Estos campos pueden ser algunos de los más novedosos y donde más se está trabajando el problema del viajante, pero existen muchas más aplicaciones en el entorno real como la optimización de rutas para el tráfico aéreo, para programar unas vacaciones con distintos sitios a visitar, para los servicios de emergencia, etc. Es importante comentar que, a la hora de tomar la decisión de una ruta no solo debe contar la información que nos proporcione en el TSP. Por ejemplo, en el caso de los servicios de emergencias, otras variables son más importantes.

2.2 Algoritmos de resolución

En esta sección se va presentar y explicar los algoritmos de resolución del problema del viajante, no se nombrarán todos, ya que, existe una gran diversidad de algoritmos y heurísticas. De los presentados en esta sección se eligieron algunos para la implementación de un código que resolviera el problema.

Se debe destacar que, como ya se ha comentado en secciones anteriores, el problema es NP-HARD, por lo que el tiempo necesario para llegar a la solución óptima puede llegar a ser muy ineficiente. Por ello, con los algoritmos que se describen, se intenta llegar a una solución los más cercana posible al óptimo.

En primer lugar, se pueden diferenciar dos grandes grupos de algoritmos, aquellos que buscan una solución que, de no ser la solución óptima, se puede saber cuánto difiere del óptimo. Y otro gran grupo que, se caracteriza por ser eficiente en tiempo polinomial, aunque no se sabe cuánto difiere la solución obtenida de la óptima. La mayoría de los algoritmos presentados en este proyecto pertenecen a este gran grupo.

Hay una manera de obtener la solución exacta para este problema, pero su tiempo de resolución no es nada eficiente. Sería obtener todas las posibles rutas y para cada una de ellas calcular el tiempo total, la ruta con menor tiempo será la óptima. Pero esta forma de resolución es totalmente inviable, por ejemplo, para un pequeño problema de 10 ciudades, existen 362.880 rutas posibles.

2.2.1 Algoritmo del vecino más próximo

También denominado, algoritmo codicioso o algoritmo voraz. Quizás, a priori, el algoritmo más fácil de implementar, y que normalmente no proporciona la solución óptima. Hay veces que sí proporciona una muy buena solución del problema, dependiendo de la localización de los nodos y del tamaño del problema. Este algoritmo realiza los siguientes pasos:

1. Se inicia buscando el nodo más cercano al nodo de inicio, cuando se encuentre dicho nodo, se añade a la ruta, quedando dicha ruta primero con el nodo inicio, seguido del nodo más cercano a este.

2. Para este último nodo incorporado a la ruta, se busca el nodo más cercano, exceptuando los que ya están añadidos a la ruta.

3. El algoritmo finaliza cuando se incorporan todos los nodos a la ruta, en ese caso, se incorporará al final de la ruta el nodo inicial, ya que, la ruta debe salir y debe finalizar en dicho nodo.

Una vez obtenida la ruta, se puede calcular el tiempo total invertido, sumando cada uno de los tiempos de ir de un nodo a otro, en el orden que nos diga la solución obtenida.

2.2.2 Algoritmo de Christofides

Algoritmo diseñado, como bien indica su nombre, por el matemático chipriota Nicos Christofides, en 1976. Este algoritmo se basa en otro, llamado el minimum spanning tree o árbol de expansión mínima, básicamente el algoritmo trata de encontrar un árbol mínimo, y, una vez encontrado, se duplican sus ramas y aplicando algunas técnicas se puede llegar a una muy buena solución del problema. Además, también utiliza el concepto de grafo euleriano, el cual se caracteriza por ser un grafo conexo donde los nodos tienen grado par. Para entender un poco más el algoritmo se presentan los siguientes pasos:

1. Llegar a un árbol de expansión mínima, esto es, una colección de n-1 ramas que unan los n nodos sin formar ciclo.

2. Una vez se obtiene dicho árbol, se aplica un matching o emparejamiento mínimo para todos los nodos de grado impar, es decir, se unen entre sí aquellas parejas de nodos que sean de grado impar, donde el número de ramas que salen y entran al nodo es impar, de esta manera se convierte el nodo en par.

3. Una vez terminado dicho proceso, se obtendrá un grafo euleriano, al cual se le aplicará la técnica de shortcuts, la cual consiste en recorrer el grafo y cuando se llegue a un nodo sin ramas de salida o cuyas ramas de salida ya han sido explorados, unirlo con el siguiente nodo del grafo que no ha sido visitado.

En Christofides, 1976, se puede ver todo lo explicado en este apartado y se puede indagar con mucho más detalle sobre el Algoritmo de Christofides.

Para entender este algoritmo y modo de ejemplo se presentan la siguiente imagen, en la que se puede ver en primer lugar un árbol de expansión mínima, seguido de un grafo euleriano y llegando a la ruta final mediante la aplicación de shortcuts.

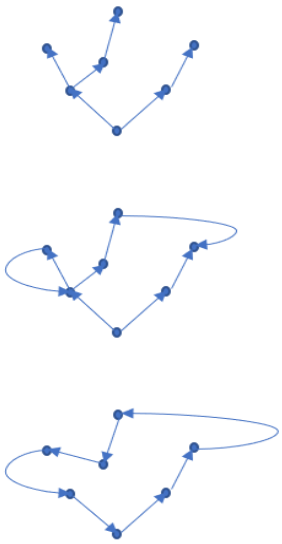

Figura 5. Ejemplo del Algoritmo de Christofides

2.2.3 Método de Branch and Bound

En castellano, método de ramificación y acotación, es una evolución del método de los planos de cortes, que se diseñó en 1954, como se ha mencionado en los fundamentos históricos de este proyecto.

En términos generales, consiste en crear un árbol de soluciones, el propio algoritmo detecta si una rama ya no puede ser óptima y la corta, es decir, no buscará más soluciones en esa rama. La eficiencia de estos algoritmos depende mucho del número de nodos que tenga el problema.

El algoritmo se basa en una técnica, llamada backtracking, similar a la que se utiliza en los árboles de expansión mínima, consiste en volver a un nodo anterior para examinar sus posibles descendientes. Y otro criterio muy importante para la aplicación de dicho algoritmo es la de podar aquella rama cuyo coste ya no permite que sea óptima, para así no perder tiempo investigando sus descendientes. Cada hoja será un subproblema, donde se decidirá si una ciudad concreta entrará en la ruta en dicha posición o no.

Para asimilar el problema un poco mejor se presentan los siguientes pasos, donde se observa de forma más intuitiva la manera de trabajar del algoritmo:

1. La primera hoja será el nodo inicial del problema.

2. A cada fila de la matriz de costo original se le resta el menor valor que aparezca en su fila,

posteriormente a cada columna de esta matriz que se obtiene se le realiza el proceso similar. Es decir, es simplemente aplicar el teorema de la matriz reducida a la matriz de costes original.

3. El coste total de la ruta será la suma de los valores que se han restado en las filas y en las columnas. Este coste se irá incrementando según avance el algoritmo.

4. Una vez se tiene la nueva matriz de costes, se elige aleatoriamente una casilla que tenga valor cero, y de aquí saldrán dos subproblemas, uno eligiendo esa casilla como óptima (por lo que se eliminará su fila y su columna de la matriz de coste modificada) y otro eligiendo esa casilla como no óptima (en cuyo caso habrá que elegir otro cero de la matriz y repetir este proceso)

5. Para cada vértice que se alcanza realizar el proceso del punto 4, cuando se llegue a un vértice cuyo coste sea mayor del coste total actualizado, cortar su progresión y hacer un backtracking al vértice anterior que no haya sido explorado. También se realizará el backtracking si el vértice ya no puede tener más ramificación.

6. La ruta óptima será la del subproblema que tenga un menor coste.

Ahora sí se puede entender mejor la dificultad que tiene este algoritmo para problemas del viajante con un número elevado de nodos a visitar. En la siguiente imagen se puede observar el árbol al que se llegaría para un ejemplo de problema de solo 4 nodos. Se puede apreciar que para tan solo 4 nodos hay un gran número de subproblemas.

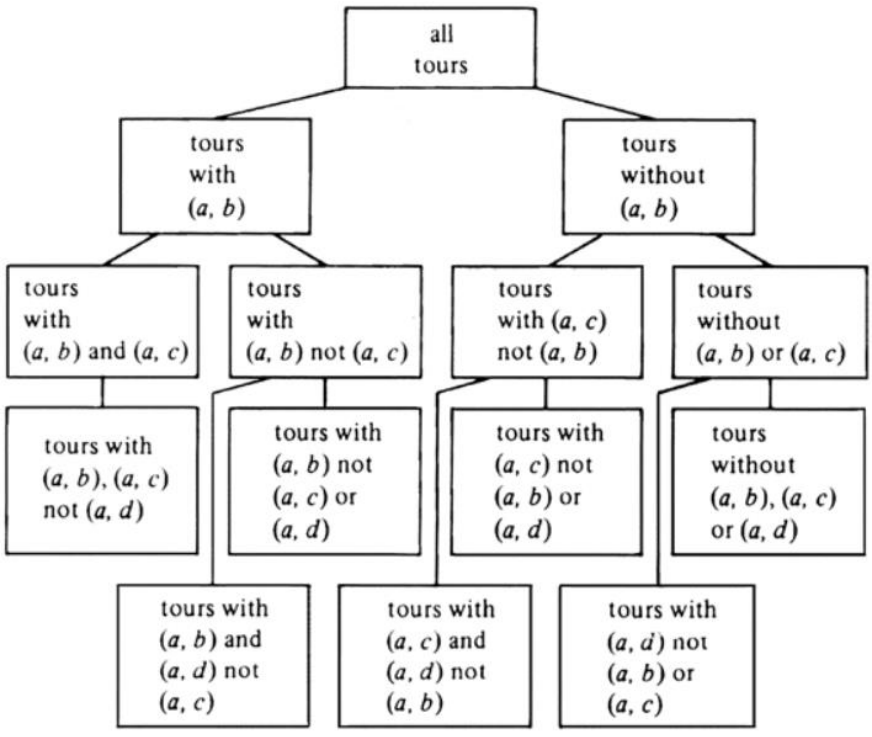

Figura 6. Ejemplo árbol de soluciones (Alfred, John, & Jeffrey, 1987)

2.2.4 Clustering

Este método de resolución del problema del viajante se basa en la agrupación de los nodos que se tienen que visitar en el problema. Básicamente trata de buscar el óptimo de cada clúster, para así llegar al óptimo del problema. Para más información sobre esta herramienta visitar Laporte, 2002.

Esta técnica por sí sola, no llega a ninguna solución, debe de estar acompañada de algún algoritmo extra, es decir, cuando se aplica clustering el resultado es una agrupación de los puntos a visitar, según unas variables, como distancia, tiempo, coste, etc. A estos grupos hay que aplicarle una heurística para obtener el óptimo de cada uno de ellos. Una ventaja de esto, es que permite hacer el problema escalable, ya que dependiendo del número de grupos y del tamaño del problema, se puede resolver muchos problemas de poco tamaño, en vez de un problema de gran tamaño.

Los pasos a seguir para llegar a una correcta agrupación de los nodos serían los presentados a continuación:

1. Establecer el número de grupos que se desean obtener, puede establecerse de manera aleatoria o según un criterio. Al número de grupos se llamará k, por ejemplo.

2. Seleccionar k nodos, uno por cada número de grupos, que se utilizarán como centroides de cada grupo. Estos deben estar lo más alejado posibles unos de otros.

3. Para el resto de nodos, calcular la distancia a los centroides y asignar al clúster cuyo centroide sea el más cercano.

4. Cuando se asigna un nodo a un clúster, se recalcula su centroide.

5. Una vez se tienen todos los grupos, aplicar alguna heurística de resolución a cada uno de ellos.

6. Cuando se obtiene la ruta de cada clúster, unir los grupos, se calcula la distancia entre los grupos y se unen los más cercanos entre sí, uniéndose nodo final de un grupo, con el nodo inicial de otro.

7. Se llega a la ruta final.

Para ver gráficamente como se resuelve el problema del viajante, empleando una herramienta de clustering se presentan a continuación esta imagen.

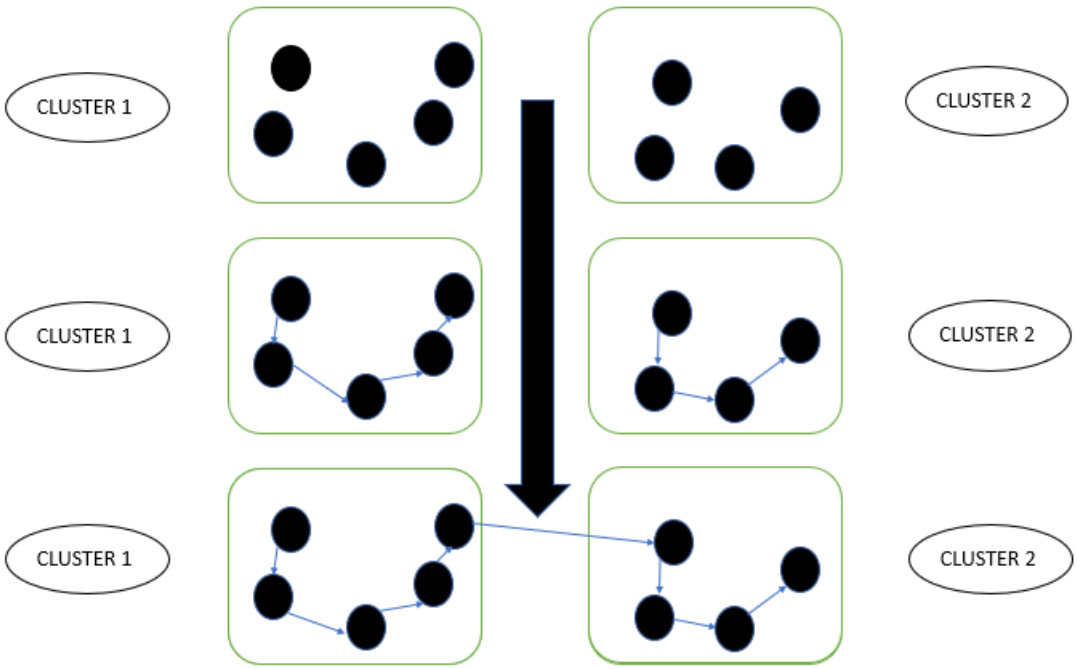

Figura 7. Ejemplo de clustering, Anaya et al. (2012)

2.2.5 K-opt

Esta heurística, desarrollada por Lin-Kernighan, necesita de la aplicación anterior de otra heurística, es decir, K-opt pertenece al grupo de heurísticas de mejoras. A partir de la solución de otro algoritmo, utiliza dicha ruta obtenida para aplicarle mejoras locales e intentar mejorar dicha ruta. En este proyecto se ha utilizado en concreto la heurística 2-opt que, fue propuesta y desarrollada por Croes en 1958.

Es un algoritmo en el que interviene demasiado el azar, ya que se eligen k nodos aleatoriamente para intercambiarlos y comprobar si el tiempo de la ruta es menor. El algoritmo se puede comprender mejor con las siguientes instrucciones:

1. Obtener una solución factible mediante otra heurística. En el caso de que la solución sea óptima, el algoritmo de mejora no proporcionará ninguna solución mejor.

2. Elegir aleatoriamente k nodos de la ruta, por ejemplo, se elige 2 nodos al azar.

3. Se cambia la posición en la ruta de un nodo por la del otro.

4. El tramo de ruta que hay entre los dos nodos se invierte.

5. Se obtiene una nueva ruta, se comprueba el tiempo que se tardaría en recorrer esta nueva ruta.

6. Si este tiempo es menor que el obtenido al principio, este será el mejor tiempo encontrado y su ruta la mejor.

7. Los pasos del 2 al 6 se pueden realizar de manera iterativa, estableciendo una condición de parada o un número máximo de iteraciones.

Se puede concluir diciendo que, puede ser una buena heurística para mejorar la solución obtenida por otro método, pero quizás es demasiado aleatoria, por lo que depende demasiado del número de iteraciones.

2.2.6 Colonia de hormigas

Como bien indica su nombre, este algoritmo refleja la vida de las hormigas: las hormigas, en la búsqueda de un objetivo donde ir, trazan primero algunos recorridos aleatorios; las hormigas que regresan antes serán las que alcancen el camino más corto, el resto de hormigas seguirá el rastro de feromonas que han dejado las hormigas que han ido por ese camino corto. Cuantas más hormigas pasen por este camino, mayor será la concentración de feromonas. Mientras que las feromonas de otras rutas menos transitadas se irán evaporando.

El algoritmo presentado aquí es un método heurístico, que se caracteriza por seguir un proceso similar. Se crean una serie de caminos aleatorios; aquellos caminos más cortos tendrán un mayor rastro de feromonas, es decir, una mayor probabilidad de ser elegido de nuevo, mientras que los más largos tendrán una probabilidad casi nula. Una explicación más detallada del algoritmo puede encontrarse en Chura et al, 2015.

Este algoritmo fue desarrollado por Marco Dorigo en 1999 como se recoge en Dorigo et al, 2004; con este método pretendían hallar la ruta más corta saliendo de su ciudad natal, dirigiéndose a otras ciudades sin repetir ninguna, y volver de nuevo a la ciudad natal.

El algoritmo está basado en proceso iterativo, en el que, en cada iteración, cada hormiga artificial realiza un camino; en el proceso de búsqueda del camino por parte de cada hormiga, se llegará a una ciudad u otra según dos variables de decisión:

- Matriz heurística: Matriz que no se modifica de una iteración a otra, siempre es constante, y está relacionada con el coste de ir de un nodo a otro.

- Matriz rastro de feromonas: Esta información se actualiza iteración a iteración, ya que depende de la concentración de feromonas que tenga cada camino, es decir, si un camino es recorrido por más hormigas tendrá una probabilidad mayor de que las siguientes hormigas también lo utilicen, como se puede apreciar en la siguiente imagen.

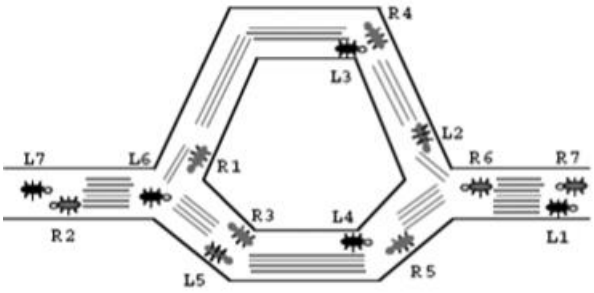

Figura 8. Ejemplo rastro de feromona (Robles Algarín, 2010)

Para concluir la explicación de este algoritmo, se va a esquematizar los pasos principales del método, desde la inicialización de los parámetros hasta la solución final.

1. Inicialización de los parámetros para la matriz heurística y el rastro de feromonas, así como el número de hormigas a crear y el número de iteraciones.

2. Mientras no se llegue al número de iteraciones o no se cumple una condición de parada, realizar el siguiente proceso iterativo.

3. Se crean las hormigas, y según la decisión de la hormiga, que depende de la matriz heurística y el rastro de feromonas, ir de ciudad en ciudad, sin repetir ciudad, hasta llegar al final.

4. Comprobar el resultado de todas las hormigas, actualizar las feromonas, para que los caminos más cortos tengas más probabilidad de ser elegidos por las hormigas de la siguiente iteración.

5. Realizar tantas iteraciones hasta las establecidas o hasta que se cumpla una condición de parada.

6. Cuando se termine el bucle, se obtendrá la solución del algoritmo.

2.2.7 Algoritmo genético

Uno de los algoritmos más utilizados para resolver este tipo de problemas de optimización, se basan en la evolución biológica de las especies. Fue ideado en 1975 por John Holland como un algoritmo innovador y revolucionario en el ámbito de la inteligencia artificial. A continuación, se presenta una breve explicación de la metodología; más detalles pueden encontrarse en Holland, 1992.

De forma general, este algoritmo se puede explicar como la creación de una población de individuos, esta población evolucionará, creándose nuevos individuos que mejorarán a los anteriores. Esta evolución se caracteriza por la mutación y el cruce de individuos, y por ser una heurística probabilística, es decir, a la capacidad de mutación y a la de cruce, se le asignarán una probabilidad.

Para entender mejor el algoritmo es necesario conocer unos conceptos fundamentales:

- Población: Conjunto de individuos que se crean para dar solución al problema, puede ser una población aleatoria o presentar ya alguna ruta alcanzada por otro algoritmo.

- Individuo: Posible solución del problema, en el caso del problema del viajante, sería una posible ruta.

- Descendientes: Son aquellos individuos que ganan en un torneo que se realiza entre los propios individuos anteriores, es decir los que tienen mejor función objetivo.

- Cruce: También denominado recombinación, como su propio nombre indica, cruza los descendientes entre sí, es decir, coge los mejores individuos y los cruza por si se crea un individuo mejor.

- Mutación: Para un individuo elegido al azar, se elige un punto arbitrario de la ruta y se intercambian sus nodos predecesores por los posteriores.

- Evolución: Tras realizar los procesos de cruce y mutación, los individuos más fuertes, es decir, los mejores, sobrevivirán y seguirán en el proceso.

Conociendo estos términos, se va a proceder a detallar los pasos que sigue el algoritmo genético en el proceso de resolución del problema del viajante:

1. Crear una población de individuos de manera aleatoria.

2. Para cada individuo de la población, se calcula el tiempo de sus rutas.

3. Realizar el enfrentamiento entre los individuos, generando unos individuos ganadores, los de menor tiempo de ruta.

4. Para estos ganadores, crear nuevos individuos, mediante la recombinación entre ellos.

5. Aplicar la mutación a un individuo y ver si mejora y puede evolucionar a otro mejor.

6. Ver los tiempos de todos estos individuos que se han creado en los pasos 4 y 5, y si algún individuo tiene un tiempo de ruta menor que el mínimo de todos los individuos anteriores, establecer como mejor tiempo y mejor individuo.

7. Repetir los pasos del 2 al 6, hasta llegar a una condición de parada o al número de iteraciones máximo.

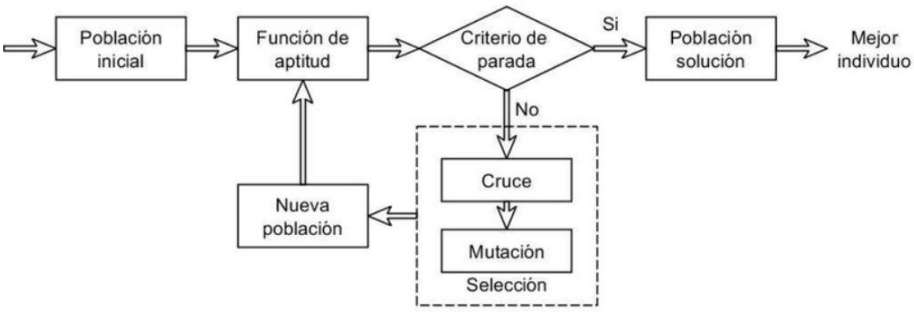

Figura 9. Algoritmo genético (Bonelli & Begliardo, 2016)

Aparte de estos pasos, es importante establecer una serie de parámetros antes de iniciar el algoritmo, como son la probabilidad de mutación, la probabilidad de cruce, el tamaño de la población, la condición de parada del bucle o el número máximo de iteraciones.

Algunas desventajas de este algoritmo podría ser la alta complejidad del método, que puede depender demasiado de los parámetros elegidos y que para el diseño del algoritmo se necesita conocimiento y experiencia en este campo. Otro punto negativo del algoritmo es que, para un número elevado de nodos, el problema no converge en un tiempo eficiente, o converge a una solución lejana del óptimo, esto último suele pasar al llegar a un óptimo local.

Los algoritmos y las heurísticas que se han explicado en esta sección son algunos de los muchos métodos que existen para resolver el problema del viajante, existen otros como búsqueda tabú (Glover & Laguna, 1997), recocido simulado (Kirkpatrick, Gelatt, & Vecchi, 1983), optimización por enjambre de partículas (Kennedy & Eberhart, 1995), etc.

En secciones posteriores se expondrá como se han utilizado algunas de las heurísticas explicadas en esta sección, los resultados obtenidos y las conclusiones sacadas. A continuación, se explica el software en el que se han implementado los algoritmos, así como otros programas empleados para la resolución del problema del viajante.

3 SOFTWARE

3.1 Matlab

Matlab es un software perfecto para realizar análisis iterativos y procesos de diseño, su creador fue Cleve Moler en 1984. Puede utilizarse en varios sistemas operativos como Microsoft Windows, Mac OS o Linux. Es de licencia privada, su propietario es MathWorks, aunque tiene una versión gratuita para 30 días.

Es de los principales softwares utilizados en universidades y muchos investigadores lo utilizan para sus estudios. Ya que, aparte de ser un software muy potente, posee una interfaz gráfica muy fácil de utilizar y cuenta con numerosas herramientas de apoyo.

Utiliza un lenguaje propio, basado en Java y C, el cual permite trabajar con vectores y matrices, programación orientada a objetos, visualizar datos en 2D y en 3D. Por ejemplo, en este trabajo han sido muy útiles una herramienta que, este software ha incorporado recientemente, como es la de clustering y redes neuronales.

Es un software que está en constante desarrollo, y es muy utilizado para el desarrollo de aplicaciones, el análisis de datos y la visualización de estos; simulación de prototipos, cálculos numéricos, etc.

Matlab cuenta con numerosas ventajas para su utilización, la nomenclatura de las funciones es fácil de recordar, ya que, es un programa orientado a científicos e ingenieros, permite escalar los problemas a tratar y posee una gran herramienta de simulación como es Simulink.

En conclusión, Matlab puede ser una muy buena opción para trabajar el problema del viajante, debido a que, permitiría una fácil programación, una ejecución del código rápida y unos resultados en los que se pueden confiar. Además, a la hora de elegir este software se valoró muy positivamente, el gran conocimiento y la experiencia que se tenía del programa, al haber trabajado anteriormente con él.

Figura 10. MATLAB. (MathWorks.)

3.2 Otro software

En el entorno real, existen muchos otros softwares a parte del ya comentado Matlab. En esta sección del proyecto se va a explicar algunos de ellos:

- Concorde: Este solucionador del problema del viajante fue desarrollado por William Cook, como se ha detallado en los fundamentos históricos de este proyecto. Gracias a él se pudo resolver el problema del viajante de mayor tamaño, 85.900 ciudades. Su biblioteca incluye más de 700 funciones encaminadas a resolver TSP y problemas de optimización de una red. Se caracteriza principalmente por ser libre y académico.

- TSP Solver and Generator: Otro gran software para la resolución del problema del viajante, se caracteriza por tener una interfaz fácil de manejar. El programa está basado en el algoritmo de ramificación y acotamiento. El lenguaje de programación que utiliza es C++ y tiene licencia GPL.

- WinQSB: Gran herramienta que permite dar solución a una gran variedad de problemas, ya que, cuenta con una gran gama de módulos, los cuales permiten solucionar problemas desde el ámbito administrativo hasta de optimización combinatoria.

Estos son algunos de los softwares más utilizados para la resolución del problema del transporte, la mayoría de ellos solo cuentan con apenas un algoritmo de resolución, por lo que no son apropiados para comparar diferentes métodos. Los comentados anteriormente son de los más potentes del mercado, por su amplia gama de algoritmos ya implementados.

Todos ellos son muy válidos para su utilización en este proyecto, aunque al final se empleó el programa Matlab, por lo comentado anteriormente, ya que se conocía bien el programa.

Con la presentación de estos softwares, se termina la parte más teórica del proyecto, en las siguientes secciones se va a detallar el caso de estudio, los procedimientos llevados a cabo, los resultados obtenidos y que conclusiones se han sacado a partir de estos resultados.

4 CASO DE ESTUDIO

4.1 Planteamiento del caso de estudio

En este capítulo, se va a presentar el planteamiento del caso de estudio, así como los datos de partida que se utilizaron para llegar a los resultados finales.

La misión del proyecto es provisionar a los centros educativos del material docente necesario para el correcto funcionamiento del curso educativo mediante una distribución coordinada. Los centros educativos se reparten por toda la ciudad de Sevilla, y el objetivo es encontrar la ruta más corta que debe realizar el transportista, pasando por todos los centros educativos sin repetir ninguno. De esta manera se conseguirá una mejora de la actual forma de distribución del material escolar en Sevilla, con los beneficios que esto implica y que fueron detallados al principio del presente proyecto.

Es importante comentar, que, a parte de los tiempos obtenidos en la ruta, hay que tener en cuenta los tiempos de carga de material y los de descarga en los centros educativos. Por ello se han estimado los siguientes tiempos:

- Tiempo de carga: 60 minutos
- Tiempo de descarga: 3 minutos por centro educativo

En la siguiente tabla se puede observar los centros educativos públicos de Sevilla, con el índice que se le asocia para la aplicación de los algoritmos.

1	Colegio de Educación Infantil y Primaria	Adriano
2	Colegio de Educación Infantil y Primaria	Adriano del Valle
3	Colegio de Educación Infantil y Primaria	Al-Andalus
4	Colegio de Educación Infantil y Primaria	Alfares
5	Colegio de Educación Infantil y Primaria	Almotamid
6	Colegio de Educación Infantil y Primaria	Andalucía
7	Colegio de Educación Infantil y Primaria	Ángel Ganivet
8	Colegio de Educación Infantil y Primaria	Aníbal González
9	Colegio de Educación Infantil y Primaria	Arias Montano
10	Colegio de Educación Infantil y Primaria	Arrayanes
11	Colegio de Educación Infantil y Primaria	Azahares
12	Colegio de Educación Infantil y Primaria	Baltasar de Alcázar
13	Colegio de Educación Infantil y Primaria	Blas Infante
14	Colegio de Educación Infantil y Primaria	Borbolla
15	Colegio de Educación Infantil y Primaria	Calvo Sotelo
16	Colegio de Educación Infantil y Primaria	Capitán General Julio Coloma Gallegos
17	Colegio de Educación Infantil y Primaria	Carlos V
18	Colegio de Educación Infantil y Primaria	Carmen Benítez
19	Colegio de Educación Infantil y Primaria	Cristóbal Colón
20	Colegio de Educación Infantil y Primaria	El Manantial

21	Colegio de Educación Infantil y Primaria	Emilio Prados
22	Colegio de Educación Infantil y Primaria	Escritor Alfonso Grosso
23	Colegio de Educación Infantil y Primaria	España
24	Colegio de Educación Infantil y Primaria	Federico García Lorca
25	Colegio de Educación Infantil y Primaria	Fernán Caballero
26	Colegio de Educación Infantil y Primaria	Fray Bartolomé de las Casas
27	Colegio de Educación Infantil y Primaria	Hermanos Machado
28	Colegio de Educación Infantil y Primaria	Híspalis
29	Colegio de Educación Infantil y Primaria	Huerta de Santa Marina
30	Colegio de Educación Infantil y Primaria	Huerta del Carmen
31	Colegio de Educación Infantil y Primaria	Ignacio Sánchez Mejías
32	Colegio de Educación Infantil y Primaria	Isbilya
33	Colegio de Educación Infantil y Primaria	Jacarandá
34	Colegio de Educación Infantil y Primaria	Jardines del Valle
35	Colegio de Educación Infantil y Primaria	Concepción Estevarena
36	Colegio de Educación Infantil y Primaria	Joaquín Turina
37	Colegio de Educación Infantil y Primaria	Jorge Juan y Antonio Ulloa
38	Colegio de Educación Infantil y Primaria	José María del Campo
39	Colegio de Educación Infantil y Primaria	José Sebastián y Bandarán
40	Colegio de Educación Infantil y Primaria	Juan de la Cueva
41	Colegio de Educación Infantil y Primaria	Juan de Mairena
42	Colegio de Educación Infantil y Primaria	Juan Ramón Jiménez
43	Colegio de Educación Infantil y Primaria	Juan Sebastián Elcano
44	Colegio de Educación Infantil y Primaria	Juan XXIII
45	Colegio de Educación Infantil y Primaria	La Candelaria
46	Colegio de Educación Infantil y Primaria	La Raza
47	Colegio de Educación Infantil y Primaria	Lope de Rueda
48	Colegio de Educación Infantil y Primaria	Lora Tamayo
49	Colegio de Educación Infantil y Primaria	Macarena
50	Colegio de Educación Infantil y Primaria	Maestra Isabel Álvarez
51	Colegio de Educación Infantil y Primaria	Maestro José Fuentes
52	Colegio de Educación Infantil y Primaria	Manuel Altolaguirre
53	Colegio de Educación Infantil y Primaria	Manuel Canela
54	Colegio de Educación Infantil y Primaria	Manuel Giménez Fernández
55	Colegio de Educación Infantil y Primaria	Manuel Siurot
56	Colegio de Educación Infantil y Primaria	María Zambrano
57	Colegio de Educación Infantil y Primaria	Mariana de Pineda
58	Colegio de Educación Infantil y Primaria	Marie Curie
59	Colegio de Educación Infantil y Primaria	Menéndez Pidal
60	Colegio de Educación Infantil y Primaria	Miguel Hernández
61	Colegio de Educación Infantil y Primaria	Ntra.Sra. de la Paz
62	Colegio de Educación Infantil y Primaria	Ntra.Sra. del Águila
63	Colegio de Educación Infantil y Primaria	Ortiz de Zúñiga
64	Colegio de Educación Infantil y Primaria	Pablo Ruiz Picasso
65	Colegio de Educación Infantil y Primaria	Pablo VI
66	Colegio de Educación Infantil y Primaria	Paulo Orosio
67	Colegio de Educación Infantil y Primaria	Paz y Amistad
68	Colegio de Educación Infantil y Primaria	Pedro Garfias
69	Colegio de Educación Infantil y Primaria	Pino Flores
70	Colegio de Educación Infantil y Primaria	Pío XII
71	Colegio de Educación Infantil y Primaria	Prácticas
72	Colegio de Educación Infantil y Primaria	Príncipe de Asturias
73	Colegio de Educación Infantil y Primaria	Rico Cejudo
74	Colegio de Educación Infantil y Primaria	San Ignacio de Loyola
75	Colegio de Educación Infantil y Primaria	San Isidoro
76	Colegio de Educación Infantil y Primaria	San Jacinto
77	Colegio de Educación Infantil y Primaria	San José de Calasanz
78	Colegio de Educación Infantil y Primaria	San José de Palmete
79	Colegio de Educación Infantil y Primaria	San José Obrero
80	Colegio de Educación Infantil y Primaria	San Juan de Ribera
81	Colegio de Educación Infantil y Primaria	San Pablo
82	Colegio de Educación Infantil y Primaria	Santa Clara
83	Colegio de Educación Infantil y Primaria	Sor Ángela de la Cruz

84	Colegio de Educación Infantil y Primaria	Tartessos
85	Colegio de Educación Infantil y Primaria	Teodosio
86	Colegio de Educación Infantil y Primaria	Tierno Galván
87	Colegio de Educación Infantil y Primaria	Valdés Leal
88	Colegio de Educación Infantil y Primaria	Valeriano Bécquer
89	Colegio de Educación Infantil y Primaria	Vara del Rey
90	Colegio de Educación Infantil y Primaria	Vélez de Guevara
91	Colegio de Educación Infantil y Primaria	Victoria Díez
92	Colegio de Educación Infantil y Primaria	Zurbarán
93	Colegio de Educación Primaria	Buenavista
94	Colegio de Educación Primaria	Cruz del Campo
95	Escuela Infantil	Adelfa
96	Escuela Infantil	Andaluna
97	Escuela Infantil	Ángel de la Guarda
98	Escuela Infantil	Arco e Iris
99	Escuela Infantil	Argote de Molina
100	Escuela Infantil	Fernando Villalón
101	Escuela Infantil	Gloria Fuertes
102	Escuela Infantil	Julio César
103	Escuela Infantil	María Inmaculada
104	Escuela Infantil	Martín de Gainza
105	Escuela Infantil	Niño Jesús
106	Escuela Infantil	Nuestra Señora de la Candelaria
107	Escuela Infantil	Pablo de Olavide
108	Escuela Infantil	Sagrada Familia
109	Escuela Infantil	San Jerónimo
110	Escuela Infantil	Santa Catalina
111	Escuela Infantil	Santa Luisa de Marillac
112	Escuela Infantil	Santa María de los Ángeles
113	Escuela Infantil	Santísima Trinidad
114	Escuela Infantil	Toribio de Velasco
115	Escuela Infantil	Torreblanca
116	Escuela Infantil	Virgen de los Reyes
117	Escuela Infantil	De Sevilla
118	Instituto de Educación Secundaria	Albert Einstein
119	Instituto de Educación Secundaria	Antonio Domínguez Ortiz
120	Instituto de Educación Secundaria	Antonio Machado
121	Instituto de Educación Secundaria	Azahar
122	Instituto de Educación Secundaria	Beatriz de Suabia
123	Instituto de Educación Secundaria	Bellavista
124	Instituto de Educación Secundaria	Carlos Haya
125	Instituto de Educación Secundaria	Chaves Nogales
126	Instituto de Educación Secundaria	Ciudad Jardín
127	Instituto de Educación Secundaria	Diamantino García Acosta
128	Instituto de Educación Secundaria	Federico Mayor Zaragoza
129	Instituto de Educación Secundaria	Félix Rodríguez de la Fuente
130	Instituto de Educación Secundaria	Fernando de Herrera
131	Instituto de Educación Secundaria	Gustavo Adolfo Bécquer
132	Instituto de Educación Secundaria	Heliópolis
133	Instituto de Educación Secundaria	Inmaculada Vieira
134	Instituto de Educación Secundaria	Isbilya
135	Instituto de Educación Secundaria	Joaquín Romero Murube
136	Instituto de Educación Secundaria	Joaquín Turina
137	Instituto de Educación Secundaria	Julio Verne
138	Instituto de Educación Secundaria	Leonardo da Vinci
139	Instituto de Educación Secundaria	Llanes
140	Instituto de Educación Secundaria	Los Viveros
141	Instituto de Educación Secundaria	Luca de Tena
142	Instituto de Educación Secundaria	Macarena
143	Instituto de Educación Secundaria	María Moliner
144	Instituto de Educación Secundaria	Martínez Montañés
145	Instituto de Educación Secundaria	Miguel de Cervantes
146	Instituto de Educación Secundaria	Miguel Servet

147	Instituto de Educación Secundaria	Murillo
148	Instituto de Educación Secundaria	Nervión
149	Instituto de Educación Secundaria	Pablo Picasso
150	Instituto de Educación Secundaria	Pino Montano
151	Instituto de Educación Secundaria	Polígono Sur
152	Instituto de Educación Secundaria	Politécnico
153	Instituto de Educación Secundaria	Punta del Verde
154	Instituto de Educación Secundaria	Ramón Carande
155	Instituto de Educación Secundaria	Ramón del Valle Inclán
156	Instituto de Educación Secundaria	Salvador Távora
157	Instituto de Educación Secundaria	San Isidoro
158	Instituto de Educación Secundaria	San Jerónimo
159	Instituto de Educación Secundaria	San Pablo
160	Instituto de Educación Secundaria	Santa Aurelia
161	Instituto de Educación Secundaria	Sevilla-Este
162	Instituto de Educación Secundaria	Siglo XXI
163	Instituto de Educación Secundaria	Torreblanca
164	Instituto de Educación Secundaria	Triana
165	Instituto de Educación Secundaria	V Centenario
166	Instituto de Educación Secundaria	Velázquez
167	Instituto de Educación Secundaria	Vicente Aleixandre

Tabla 2. Centros educativos de Sevilla. (Junta de Andalucía. Consejería de Educación.)

Para entender mejor la ubicación geográfica de los centros educativos se presentan las siguientes imágenes, en la primera se observa un conjunto de puntos ubicados en el mapa, cada punto representa un centro educativo; en la segunda se observa un mapa de calor de los centros educativos.

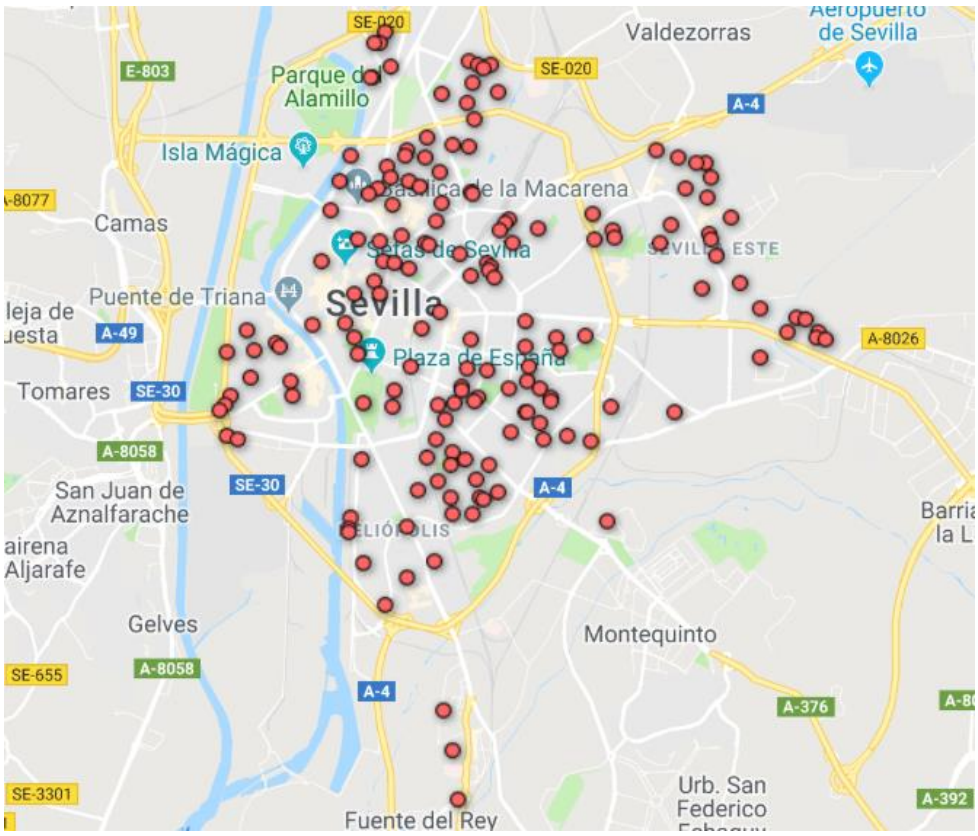

Figura 11. Localización de los centros educativos. (“GoogleMaps,” n.d.).

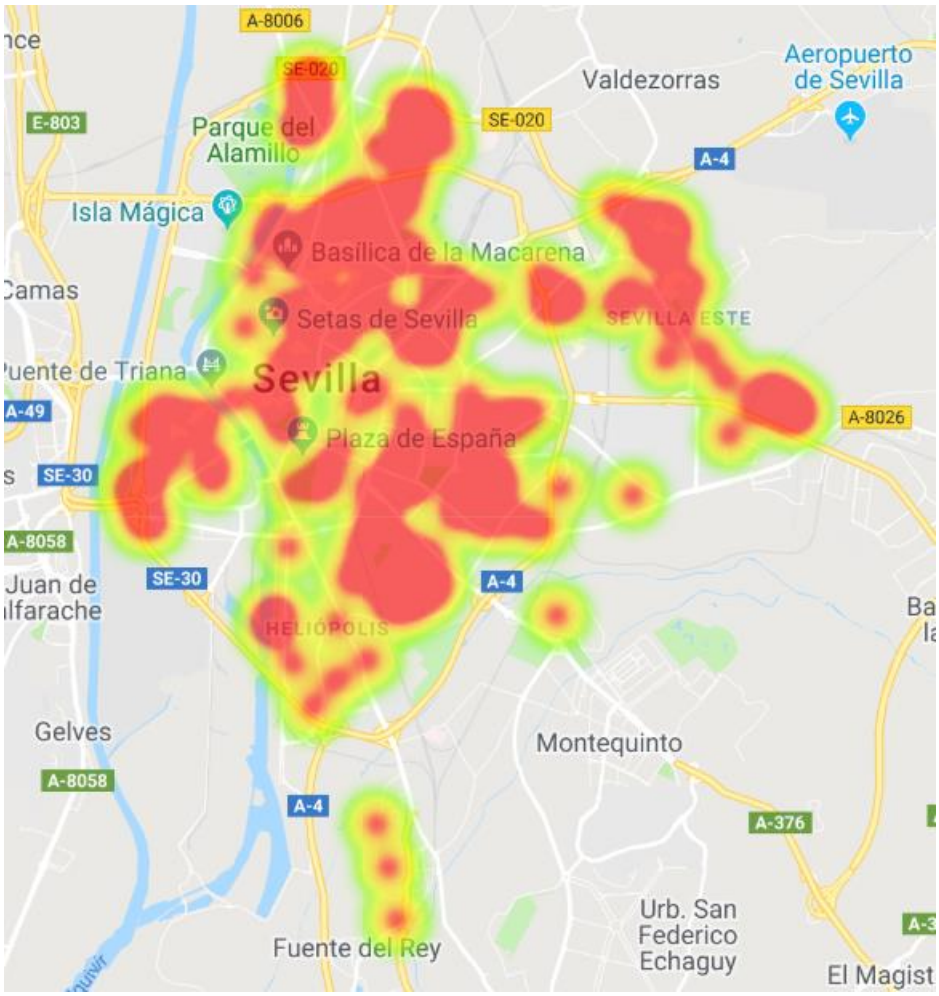

Figura 12. Mapa de calor de los centros educativos. (“GoogleMaps,” n.d.).

4.2 Tratamiento de datos de partida

Para poder saber cuáles son y donde se sitúan, todas las guarderías, colegios e institutos de la ciudad de Sevilla, se realizó una búsqueda, llegando a una página web de la Junta de Andalucía que proporcionaba un archivo .csv con la dirección, nombre, código postal, si era público o privado, teléfono, etc. En dicho archivo se contemplaba toda la provincia de Sevilla, por lo que a través de un filtrado se quedaron solo los pertenecientes a Sevilla capital, y que además fueran públicos. Este archivo .csv separado por comas, se pasó a un fichero Excel en primer lugar. Realizando dichos filtros se quedaron 167 centros educativos entre guarderías, colegios e institutos, todos ellos públicos, los cuales formarán el número de nodos del problema a resolver.

En un principio se utilizó una API de Google para conseguir las latitudes y longitudes de cada centro educativo, para posteriormente, y a través de la misma API, generar los tiempos entre los colegios. Para ello se implementó en Matlab dos códigos. Uno para conseguir la latitud y la longitud, y otro para obtener los tiempos.

Para implementar y para asegurar que los códigos estaban correctos, se utilizó una parte pequeña del listado de centros educativos, cuando se verificó que todo estaba correcto se aplicó a todo el listado.

En un principio los códigos para la API funcionaban correctamente, salvo que la aplicación tenía un límite de peticiones por minuto y por día, este pequeño problema se solventó mediante algunas instrucciones en el código que controlaban los errores y gestionaban una lista de peticiones pendientes.

De esa manera, se obtuvo todas las latitudes y longitudes de la lista pequeña de los centros educativos, y los tiempos entre ellos. Estos tiempos fueron utilizados para realizar los algoritmos de resolución, para así facilitar su implementación y asegurar que estos funcionaban correctamente.

Una vez se comprobó que ambos códigos funcionaban correctamente, se aplicaron al total de los centros educativos, pero al intentar ejecutarlos no se podía, debido a un error relacionado con la API de Google. Buscando información sobre el posible error, se llegó a que para poder utilizar dicha API se necesita una Key, que se puede conseguir facilitando una cuenta para facturación. Google proporciona un número de créditos gratuitos, por lo que se intuye que se ha superado dichos créditos.

Por lo que tras mucha investigación sobre cómo conseguir los tiempos entre los centros educativos, se implementaron variaciones en ambos códigos anteriores, se introdujeron algunas instrucciones para que lanzara peticiones directamente a Google Maps. La respuesta a estas peticiones es mucho más laboriosa de leer, y era más difícil de encontrar la información que se buscaba. Aunque finalmente, y tras un tiempo de implementación del código se consiguió obtener toda la información necesaria de todos los centros educativos.

La matriz final de tiempos entre centros educativos, y las latitudes y longitudes de estos, se exportaron a dos ficheros Excel diferentes para su posterior acceso durante la ejecución de los algoritmos. Además, se comprobaron varios de estos datos obtenidos, introduciéndolos aleatoriamente en Google Maps y obteniendo el resultado esperado. Estos códigos utilizaban como datos de partida:

- Tipo de centro educativo: Colegio, guardería o instituto
- Nombre del centro educativo
- Dirección del centro educativo
- Código Postal del centro educativo

Esta información sirvió posteriormente para calcular las soluciones a cada una de las heurísticas que se implementaron en Matlab. Se debe destacar que la idea de conseguir las latitudes y longitudes, era para que, a partir de estas, se obtuvieran los tiempos. Aunque al final, los tiempos se obtuvieran mediante la propia dirección. Pero dichas latitudes y longitudes sirvieron posteriormente, para realizar un clustering, y clasificar los centros educativos en 4 grupos, ya que, la ruta completa de todos los centros educativos no se podía realizar por motivos que se explicarán posteriormente. El clustering se realizó a través de una aplicación de Matlab que se basa en redes neuronales.

En el anexo del proyecto están reflejados estos códigos, donde se ven las instrucciones, en forma de comentarios, que se utilizaron para la API, y las que posteriormente se utilizaron para Google Maps. Estos códigos se denominaron lat_lot_colegios.m y MATRIZ_OD_COLEGIOS.m.

5 RESULTADOS EXPERIMENTALES

En este capítulo del proyecto, se van a presentar los resultados obtenidos tras aplicar algunos de los algoritmos explicados en el capítulo 1, para cada heurística se explicará brevemente el código implementado, los problemas que surgieron y los resultados obtenidos.

Se va a presentar el resultado de cada heurística, y el resultado de esta tras aplicarle también otra heurística de mejora, K-opt.

Para mostrar la solución de cada heurística se presenta una tabla por solución, en dicha tabla se muestra el orden a seguir en la ruta. Los números que se observan en las tablas corresponden al índice que se le asignó a cada centro educativo en la matriz de origen y destino. Toda ruta debe comenzar con el índice 1 y terminar con el mismo índice, ya que se debe comenzar y terminar en el mismo punto. El orden de la ruta se visualiza en las tablas leyéndolas de izquierda a derecha comenzando por la primera fila.

5.1 Vecino más próximo

Como se ha comentado anteriormente, es la heurística más sencilla y más rápida de implementar en Matlab. Apenas hubo problemas para la realización del código, ya que este era muy intuitivo.

Tiempo de la ruta: 53.617 segundos, 893’62 minutos, 14’9 horas.

1	85	118	27	139	99	79	142	83	87
113	34	18	103	75	61	94	144	122	148
66	37	40	156	160	138	65	127	62	67
6	53	54	26	52	151	119	101	135	154
63	45	126	25	19	5	92	88	44	105
106	91	60	159	116	14	97	81	108	80
136	12	74	143	35	10	95	149	20	51
161	165	155	32	84	11	33	125	47	146

7	28	82	57	22	120	3	9	15	134
141	71	104	112	21	114	8	23	130	153
46	58	128	48	123	39	16	98	132	43
73	140	164	76	38	42	77	89	124	131
4	117	167	111	68	121	110	70	13	133
64	31	56	137	109	93	24	100	158	102
69	55	150	41	145	30	166	49	29	157
129	96	17	162	115	59	86	163	90	72
2	78	107	147	36	50	152	1		

Tabla 3. Ruta del vecino más próximo

En la imagen anterior se aprecia la ruta a seguir según la heurística del vecino más próximo, el tiempo total sería de 14,9 horas. Se debe destacar que este tiempo corresponde al tiempo de ir de un punto a otro, a este tiempo habrá que añadirles otros tiempos necesarios para realizar el reparto.

- Aplicándole la mejora K-opt: 53.252 segundos, 887’53 minutos, 14’80 horas

El orden a seguir es prácticamente el mismo, solo existen pequeñas variaciones de intercambio entre dos puntos. Esta solución ha mejorado la anterior del vecino más próximo en unos 6 minutos.

1	85	118	27	139	99	79	142	83	87
113	34	18	103	75	61	94	144	122	148
66	37	40	156	160	138	65	127	62	67
6	53	54	26	52	151	119	101	135	154
63	45	126	25	19	5	92	88	44	105
106	91	60	159	116	14	97	81	108	80
136	12	74	143	35	10	95	149	20	51
161	165	155	32	84	11	33	125	47	146
7	28	82	57	22	120	3	9	15	134

141	71	104	21	112	114	8	23	130	132
43	153	46	58	128	48	123	39	16	98
73	140	164	76	38	4	131	124	89	77
42	117	167	111	68	121	110	70	13	133
64	31	56	137	109	93	24	100	158	102
69	55	150	41	145	30	166	49	29	157
129	96	17	162	115	59	86	163	90	72
2	78	107	147	36	50	152	1		

Tabla 4. Ruta del vecino más próximo optimizada

5.2 Colonia de hormigas

Algoritmo que depende de algunos parámetros elegidos, como bien se explica en el marco teórico de esta heurística, el cual se puede ver en el capítulo 1 de este proyecto. A continuación, se presentan los valores que se han establecido para estos parámetros, el mejor tiempo obtenido y su ruta.

- Iteraciones: 400
- Número de hormigas: 300
- Coeficiente de evaporación:0.1
- Importancia rastro feromona: 1
- Importancia matriz heurística:1
- Importancia del rastro del nuevo mejor camino: 5

Todos estos parámetros se han ido cambiando, obteniéndose distintas soluciones para esta heurística. La mejor solución obtenida se muestra a continuación, la cual se alcanzó con los parámetros presentados arriba.

Tiempo de la ruta: 50.969 segundos, 849’49 minutos, 14’16 horas.

1	85	118	109	93	24	100	158	129	150
137	31	56	121	139	99	79	142	83	87
113	34	103	75	111	131	4	124	89	77
42	164	76	38	140	73	117	167	157	29
3	120	159	60	116	94	144	122	148	147
14	71	104	44	105	106	91	21	112	114
126	25	5	92	135	154	63	45	160	138
65	127	62	67	6	53	54	26	52	151
88	78	72	17	162	115	59	146	7	28
82	97	81	74	108	80	136	143	35	10
95	149	20	51	165	155	32	84	11	33
125	47	96	161	86	163	90	2	58	153
46	16	98	132	43	128	48	123	39	107
8	141	61	166	49	68	55	69	12	22
57	41	64	102	27	110	70	13	133	9
15	134	30	145	18	23	50	36	19	119
101	66	37	40	156	130	152	1		

Tabla 5. Ruta de la colonia de hormigas

Se ha de destacar que es un algoritmo que mejora la solución del vecino más próximo, el inconveniente es que el tiempo computacional es algo elevado, pero es algo que se puede pasar por alto, debido a la buena solución que ha proporcionado. Este tiempo computacional es alto debido al gran número de nodos que presenta el problema, al tamaño de la población de hormigas que se ha elegido y al número de iteraciones establecido. Si se disminuye esta población de hormigas y/o el número de iteraciones, el tiempo computacional disminuye.

Se hicieron varias pruebas disminuyendo ambos parámetros, y el tiempo computacional disminuyó algo, pero la solución empeoró demasiado, acercándose, e incluso sobrepasando, el tiempo del vecino más próximo.

- Aplicándole la mejora K-opt: 50.549 segundos, 842’48 minutos, 14’04 horas.

1	85	118	158	100	24	93	109	129	150
137	31	56	121	139	99	79	142	83	87
113	34	103	75	111	4	131	124	89	42
77	164	76	38	140	73	117	167	157	29
3	120	159	60	116	94	144	122	148	147
14	71	104	112	21	91	106	105	44	114
126	25	5	92	135	154	63	45	160	138
65	127	62	67	6	53	54	26	52	151
88	78	72	17	162	115	59	146	7	28
82	81	97	74	108	80	136	143	35	10
95	149	20	51	165	155	32	84	11	33
125	47	96	161	86	163	90	2	46	153
58	16	98	132	43	39	123	48	128	107
8	141	61	166	49	68	55	69	12	22
57	41	64	102	27	110	70	13	133	9
15	134	30	145	18	23	50	36	19	119
101	66	37	40	156	130	152	1		

Tabla 6. Ruta de la colonia de hormigas optimizada

Tras la aplicación de la heurística de mejora K-opt, se consiguió disminuir el tiempo en unos 7 minutos, consiguiendo completarse la ruta en 14.04 horas.

5.3 Branch and Bound

En este caso, el código implementado no es exactamente el método de Branch and Bound, aunque sí está basado en él. La diferencia principal es que en el código no se utiliza la técnica de backtracking, mientras que en Branch and Bound es básico su uso. La explicación de porque no se ha utilizado es por el tiempo computacional, es decir, si se usara dicha técnica el tiempo de ejecución sería muy ineficiente.

Como bien se explicó en el apartado teórico de este algoritmo, Branch and Bound es un método de resolución exacto, pero que depende demasiado del número de nodos. Para el ejemplo, que se vio en la imagen de esa sección, se resolvía un problema con 4 ciudades, y para resolverlo con este método se debía resolver 14 subproblemas. En el caso de estudio son 167 puntos a recorrer, por lo que el número de subproblemas se hace inviable en tiempo de ejecución.

Por ello, se ha optado mejorar el tiempo computacional a pesar de que la solución obtenida se verá afectada negativamente. Para eso, se ha obviado el backtracking, por lo que el algoritmo implementado se basa en la aplicación de la matriz reducida. Se aplica la matriz reducida a la matriz de costes, eliminando de la matriz de costes cada posible casilla (origen, destino) en un proceso iterativo, se elige la casilla cuya matriz modificada tenga menor coste. El nodo destino de esa casilla se convierte en origen, y se vuelve a realizar el proceso iterativo. Así, hasta recorrer todos los nodos. El orden de las casillas, será la ruta final.

Al no hacer el backtracking, el árbol que queda es de tan solo una rama, ya que solo se recorre un camino.

Explicado todo esto, se va a presentar la ruta obtenida y el tiempo de dicha ruta, así como la solución con la heurística de mejora.

Tiempo de la ruta: 55.186 segundos, 919’77 minutos, 15’33 horas.

1	85	137	102	27	64	118	150	41	139
99	79	142	68	121	129	109	93	24	158
100	55	69	9	15	87	18	34	113	61
166	157	49	83	29	134	12	136	143	96
11	20	47	28	82	57	22	116	14	147
104	104	141	94	8	66	37	40	156	63
45	160	2	114	126	25	19	5	92	135
154	67	6	53	54	26	132	43	39	123
48	128	58	107	88	78	72	17	162	86
161	51	165	155	32	84	149	95	10	35
146	7	74	108	80	97	81	3	120	13
133	31	56	110	70	117	167	111	152	140
73	124	89	130	23	50	36	148	65	138

127	62	52	151	119	101	105	44	21	112
60	159	91	106	115	59	33	125	90	163
16	98	164	76	38	42	77	4	131	145
30	103	75	153	46	144	122	1		

Tabla 7. Ruta de Branch and Bound

Se ha mejorado mucho el tiempo computacional, pero como bien se pensaba, se ha empeorado demasiado la solución, siendo peor incluso que, la del vecino más próximo. El problema es que para obtener la solución exacta habría que solucionar todos los subproblemas del árbol. Por lo que no es un buen método de resolución para problemas de gran tamaño.

- Aplicándole la mejora K-opt: 54.877 segundos, 914’62 minutos, 15’24 horas

1	85	137	102	27	64	118	150	41	139
99	79	142	68	121	129	100	158	24	93
109	55	69	9	15	87	18	34	113	61
166	157	49	83	29	134	12	136	143	96
11	20	47	28	82	57	22	116	14	147
71	104	141	94	8	66	37	40	156	63
45	160	2	114	126	25	19	5	92	135
154	67	6	53	54	26	132	43	39	123
48	128	58	107	88	78	72	17	162	86
161	51	165	155	32	84	149	95	10	35
146	7	74	108	80	97	81	3	120	70
110	56	31	133	13	117	167	111	152	140
73	124	89	130	23	50	36	148	65	138
127	62	52	151	119	101	105	44	21	112
60	159	91	106	115	59	33	125	90	163
16	98	164	76	38	42	77	4	131	145

30	103	75	153	46	144	122	1

Tabla 8. Ruta de Branch and Bound optimizada

Con la mejora de la ruta obtenida por Branch and Bound, se consigue disminuir el tiempo de la ruta en unos 5 minutos. Aun así, sigue sin ser la mejor ruta encontrada tras la aplicación de todas las heurísticas.

5.4 Algoritmo genético

Algoritmo de gran utilización actualmente en problemas de optimización de rutas, en algunas investigaciones proporcionaba buenos resultados, pero destacaban que era muy sensible a ciertos parámetros, como el número de puntos por los que tiene que pasar la ruta.

Al igual que en el algoritmo de la colonia de hormigas se hicieron distintas pruebas cambiando el valor de los parámetros que influyen en la resolución del algoritmo. Los parámetros utilizados para llegar a la mejor solución obtenida para esta heurística, son los presentados aquí:

- Tamaño de la población: 5000
- Probabilidad de mutación: 0.9
- Probabilidad de recombinación: 0.2

Tiempo de la ruta: 60.804 segundos, 1.013’4 minutos, 16’89 horas.

1	118	102	128	48	123	39	43	132	107
88	86	163	90	96	11	20	51	32	84
106	91	126	94	8	14	12	74	57	22
82	17	162	115	59	27	13	133	109	139
108	80	97	81	3	120	145	30	99	79
142	157	15	42	77	98	16	6	154	21
127	62	161	165	155	2	78	72	146	7
28	40	156	63	148	141	103	75	111	164

76	38	36	19	67	66	25	54	46	153
5	45	160	65	138	129	121	70	110	55
9	18	50	23	130	140	73	124	89	26
112	114	104	105	44	151	52	92	53	135
119	101	58	117	158	100	24	116	71	37
149	95	47	144	122	147	4	131	152	167
68	93	150	143	35	10	33	125	64	56
31	137	69	87	113	34	61	166	49	83
29	134	60	159	136	41	85	1		

Tabla 9. Ruta del algoritmo genético

Es de destacar el resultado tan malo que ha proporcionado este algoritmo, a priori, es una buena heurística de resolución, pero como bien se explicó en el marco teórico de este proyecto, depende demasiado de algunos parámetros. El número de nodos es tan grande que hace poco eficiente en tiempo al algoritmo, debido a la gran posibilidad de rutas aleatorias que existe.

- Aplicándole la mejora K-opt: 60.650 segundos,.1010’83 minutos, 16’85 horas

1	118	102	39	123	48	128	43	132	107
88	86	163	90	96	11	20	51	32	84
106	91	126	94	8	14	12	74	57	22
82	17	162	115	59	27	13	133	109	139
108	80	97	81	3	120	145	30	99	79
142	157	15	42	77	98	16	6	154	21
127	62	161	165	155	2	78	72	146	7
28	40	156	63	148	141	103	75	111	164

76	38	36	19	67	66	25	54	46	153
5	45	160	65	138	129	121	70	110	55
9	18	50	23	130	140	73	124	89	26
112	114	104	105	44	151	52	92	53	135
119	101	58	117	158	100	24	116	47	95
149	37	71	144	122	147	4	131	152	167
68	93	150	143	35	10	33	125	64	56
31	137	69	87	113	34	61	166	49	83
29	134	60	159	136	41	85	1		

Tabla 10. Ruta del algoritmo genético optimizado

Tras la aplicación de la heurística de mejora K-opt a la solución del algoritmo genético, se consigue mejorar la ruta en unos 3 minutos.

Es destacable que la heurística de mejora, que se ha aplicado a todos las soluciones, apenas mejora en 7 minutos en el mejor de los casos. Esto es debido a que dicha heurística se basa en la búsqueda de óptimos locales, por lo que ante problemas de gran tamaño no consigue mejorar demasiado la solución.

Al aplicar todos estos algoritmos, se puede observar que el tiempo de la ruta más corta sobrepasa las 8 horas, que por convenio puede trabajar un empleado. Se sobrepasa más aún, sumándole los tiempos de carga de material y de descarga en los centros educativos, que se detallaron al principio del tercer capítulo. Por lo que no se podría llevar a cabo dicho reparto, tras buscar e investigar varias soluciones, se entendió que lo mejor era dividir los centros educativos en grupos, y que cada día se repartiera a un grupo de centros educativos. Por el número de centros educativos y los tiempos obtenidos, se decidió realizar 4 grupos de centros educativos. A cada uno de estos grupos, se le volvió a aplicar el algoritmo con el que se obtuvo el mejor tiempo de la ruta completa. A parte del algoritmo con el mejor resultado también se le aplicó el vecino más próximo y el algoritmo genético, para comparar las posibles soluciones.

5.5 Clustering

Como bien se ha comentado anteriormente, un solo repartidor no podría realizar la mínima ruta en un solo día. A raíz de este problema, se estuvo investigando y pensando posibles soluciones, y se decidió, que el repartidor hiciera 4 turnos, es decir, se repartiría el material a un grupo de colegios cada día y en 4 días todos los centros educativos tendrían los materiales. Como bien se ha dicho, cada día se repartirá a un grupo de colegios, partiendo y llegando al mismo punto.

En la siguiente imagen conseguida a través de Matlab, se observa la disposición de los puntos según su latitud y longitud. Todos los puntos se observan en azul, excepto uno que se muestra con un asterisco rojo, este pertenece al punto del que saldrán y al que llegarán todas las rutas.

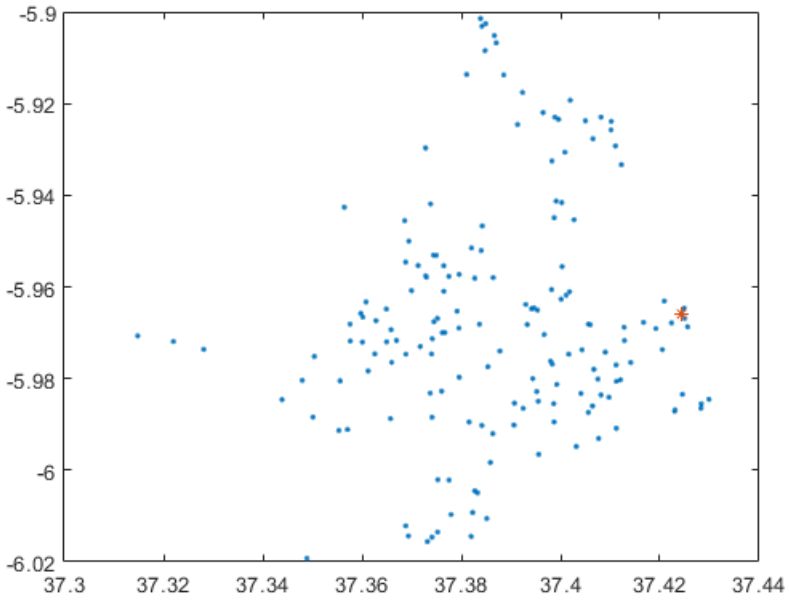

Figura 13. Localización de los centros educativos

Gracias a una herramienta del software Matlab, se pudo realizar dicha clasificación, y se organizaron en 4 grupos. Esta herramienta era nueva, por lo que se tuvo que llevar a cabo una recogida de información sobre ella, para familiarizarse con la aplicación. Esta se basa en la aplicación de redes neuronales a unos datos de entrada, y tras varias iteraciones, proporciona a la salida la agrupación.

A continuación, se expone los pasos seguidos para alcanzar dicha agrupación y qué criterios se han establecido para llegar a ello. El código para esta agrupación de centros educativos se llama cluster.m y se puede observar en el anexo del proyecto. Este código lo proporciona el propio Matlab, tras ejecutar la aplicación anteriormente mencionada. Gracias a que la interfaz de la aplicación es tan intuitiva, no fue difícil llegar a los 4 grupos de centros educativos.

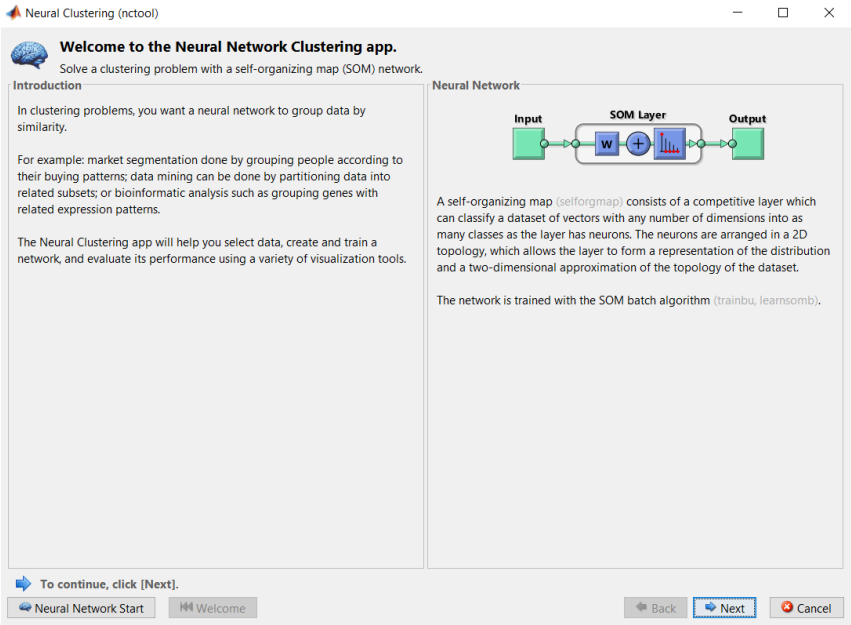

Figura 14. Interfaz de la aplicación de clustering

En primer lugar, se debía decidir el número de grupos, según los tiempos de las rutas obtenidos en las secciones anteriores, a los que hay que sumar los tiempos de carga y descarga, se decidió que fueran 4 los grupos.

Casi al mismo tiempo de esta decisión, se pensó el criterio según el cual agruparlos, se tenía cada una de las longitudes y latitudes de los centros educativos, las cuales se obtuvieron al principio del proyecto, por lo ya comentado en el capítulo 3. Utilizando estas latitudes y longitudes como entrada de datos, y teniendo en cuenta el número de grupos que se quería obtener se alcanzaron los siguientes resultados.

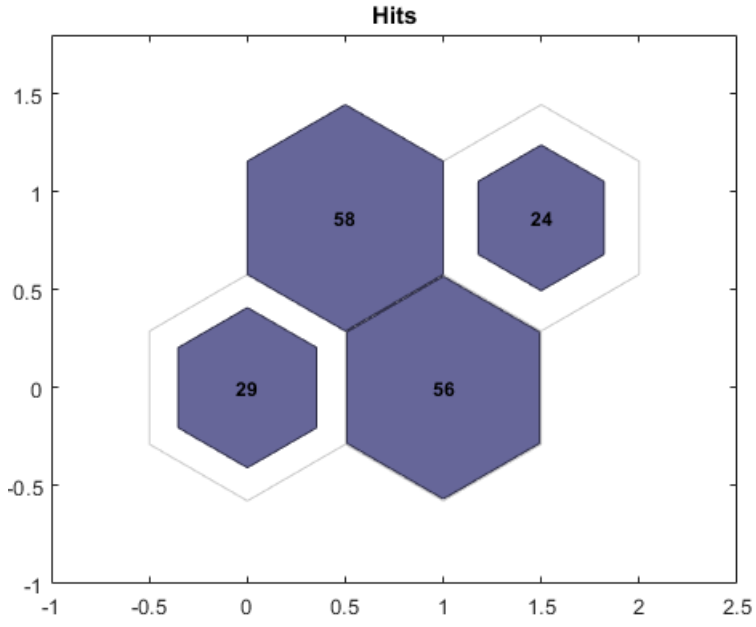

Figura 15. Número de centros educativos por grupo

Como se puede observar en la imagen anterior, los centros educativos se dividieron en 4 grupos, 29 en el grupo 1, 56 en el grupo 2, 58 en el grupo 3 y 24 en el grupo 4. Encontrándose el punto de partida y de llegada en el grupo 3, por lo que al grupo 1,2 y 4 hay que añadir un nodo más. Dicha imagen no representa la ubicación geográfica de los grupos, solo el tamaño de los grupos. Cabe destacar que se observan dos grandes grupos y dos grupos de menor tamaño.

En la siguiente imagen sí que se puede diferenciar donde están los centroides de cada grupo en función de la longitud y la latitud.

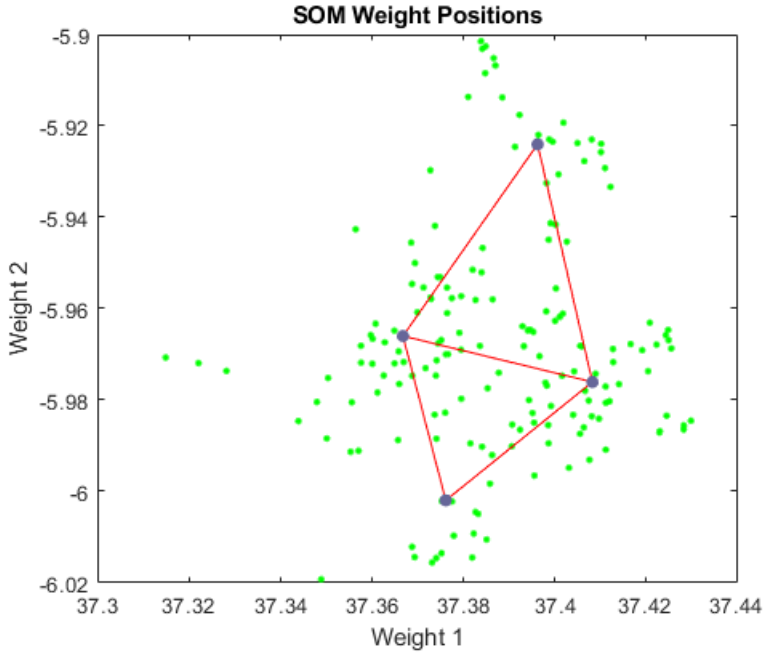

Figura 16. Centroides de los grupos

Una vez se consiguen estos grupos, se crearon 4 submatrices de tiempos origen/destino, cada una para un grupo, para así hacer más fácil las pequeñas modificaciones de los algoritmos utilizados. En la siguiente imagen, se puede diferenciar claramente los 4 grupos representados con un color distinto y la estrella sería el origen y el destino de cada una de las 4 rutas que se llevará a cabo en el reparto de material escolar en los centros educativos de Sevilla. Se puede visualizar como los grupos rojo y amarillo son los de mayor tamaño, y los grupos verde y azul los de menor tamaño.

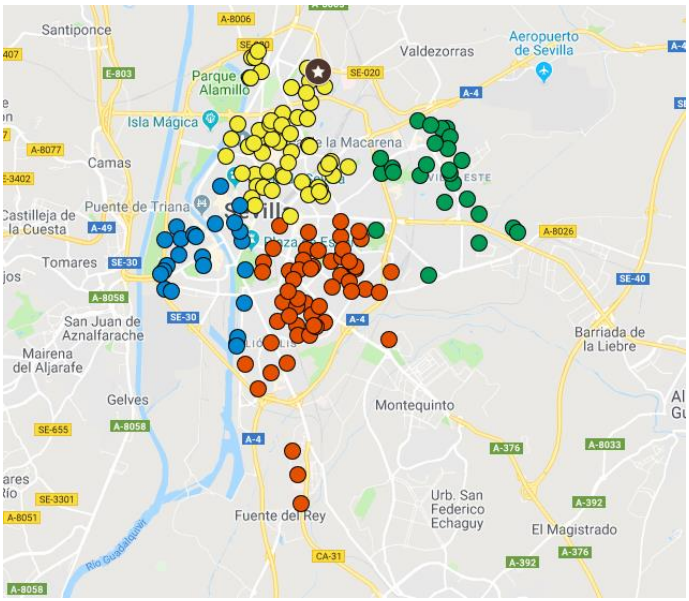

Figura 17. Localización de los centros educativos por grupo. (“GoogleMaps,” n.d.).

Se puede aceptar esta agrupación realizada por la aplicación de Matlab, ya que la disposición de los grupos coincide con el mapa de calor presentado en el capítulo del caso de estudio. En la siguiente imagen se presenta una superposición de ambos mapas. Y en ella se observa como las mayores concentraciones del mapa de calor coincide con los dos grupos mayoritarios de centros educativos.

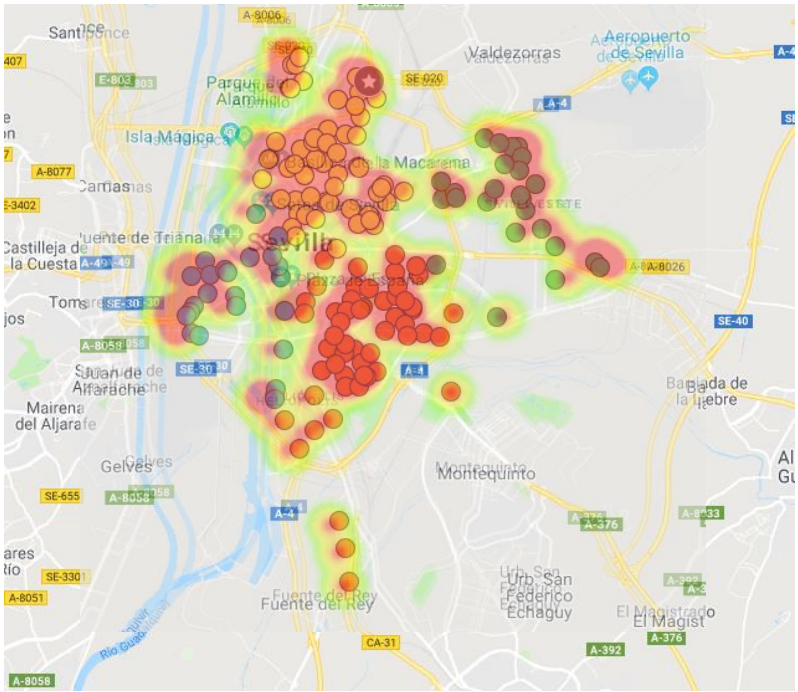

Figura 18. Relación mapa de calor con los grupos. (“GoogleMaps,” n.d.).

Debido a la creación de estas submatrices de tiempos, y de la utilización de estas en los algoritmos para cada grupo, se pierde el índice que se le asignaba a cada centro educativo en la matriz original origen/destino. Por ello se implementó un código, llamado RELACION_INDICE_COLEGIO.m, en este código se relaciona la posición de los nuevos índices con los originales, para así saber la posición de cada colegio en cada ruta.

Una vez ya se tiene los grupos, los tiempos entre los nodos de cada grupo y la relación de los índices de los centros educativos, se hicieron pequeñas modificaciones en los algoritmos antes aplicados, relacionadas con el tamaño de los vectores y matrices utilizadas, el funcionamiento de cada heurística permaneció constante.

Tras todo esto, se presentan los resultados obtenidos para el vecino más próximo, la colonia de hormigas y el algoritmo genético. Al resultado, de aquella heurística que proporcione la mejor solución, se le aplicará el algoritmo K-opt, para intentar optimizar un poco más dicho resultado.

5.6 Vecino más próximo para cada grupo

Como bien se ha comentado, se va a aplicar a cada grupo la heurística del vecino más próximo, obteniéndose 4 rutas, una para cada grupo, y 4 tiempos de ruta.

Tiempo de la ruta 1: 10.339 segundos, 172’32 minutos, 2’88 horas.

1	143	35	10	95	149	20	51	161	165
155	32	11	33	125	47	146	7	28	160
17	162	115	59	86	163	90	96	72	78
1									

Tabla 11. Ruta del grupo 1 con el vecino más próximo

Tiempo de la ruta 2: 19.311 segundos, 321’85 minutos, 5’37 horas.

1	138	65	127	62	67	6	53	54	26
52	151	119	101	135	154	63	45	144	122
148	66	37	40	156	114	126	94	141	61
147	71	104	112	21	44	105	106	91	92
88	58	16	98	132	43	39	123	48	107
2	8	25	19	5	36	50	1		

Tabla 12. Ruta del grupo 2 con el vecino más próximo

Tiempo de la ruta 3: 19.126 segundos, 318’77 minutos, 5’32 horas.

1	85	118	27	139	99	79	142	83	87
113	34	18	3	120	159	60	116	14	97
81	108	80	136	12	74	57	22	55	69
9	15	134	70	110	121	56	31	137	109
93	24	100	158	64	102	133	13	68	150
41	82	129	145	30	29	49	84	1	

Tabla 13. Ruta del grupo 3 con el vecino más próximo

Tiempo de la ruta 4: 10.501 segundos, 175’02 minutos, 2’92 horas.

1	117	73	140	164	76	38	42	77	89
124	130	23	103	75	111	131	4	152	167
157	166	153	46	128	1				

Tabla 14. Ruta del grupo 4 con el vecino más próximo

El tiempo total de todos los grupos sería de: 59.277 segundos, 987’95 minutos, 16’47 horas.

5.7 Colonia de hormigas para cada grupo

Esta heurística proporcionaba la mejor solución, para una ruta por todos los centros educativos, ahora se va aplicar a cada grupo y se comprobará si sigue siendo la mejor heurística para este problema. Los valores de los parámetros utilizados para los grupos fueron los mismos que para el conjunto total de centros educativos, estos parámetros son:

- Iteraciones: 400
- Número de hormigas: 300
- Coeficiente de evaporación:0.1
- Importancia rastro feromona: 1
- Importancia matriz heurística:1
- Importancia del rastro de nuevo mejor camino: 5

Tiempo de la ruta 1: 9.031 segundos, 150’52 minutos, 2’51 horas.

1	143	149	95	10	35	96	51	161	165
155	32	20	11	33	125	47	146	7	28
160	78	72	17	162	115	59	90	163	86
1									

Tabla 15. Ruta del grupo 1 con la colonia de hormigas

Tiempo de la ruta 2: 18.306 segundos, 305’10 minutos, 5’09 horas.

1	44	105	106	91	71	104	21	112	114
126	94	141	61	8	156	40	37	66	144
122	148	147	25	101	119	135	154	63	45
2	138	65	127	62	67	6	53	54	132
43	39	123	48	58	16	98	5	92	52
151	26	36	50	19	107	88	1		

Tabla 16. Ruta del grupo 2 con la colonia de hormiga

Tiempo de la ruta 3: 17.209 segundos, 286’82 minutos, 4’78 horas.

1	85	118	109	64	31	56	121	139	99
79	142	83	29	49	68	27	110	70	13
133	9	113	34	18	3	120	22	57	41
102	137	158	100	24	93	55	69	150	145
30	15	87	116	60	159	136	12	74	108
80	97	81	134	14	82	84	129	1	

Tabla 17. Ruta del grupo 3 con la colonia de hormigas

Tiempo de la ruta 4: 9.447 segundos, 157’45 minutos, 2’63 horas.

1	157	166	103	75	111	4	131	124	89

77	42	73	117	167	164	76	38	152	140
130	23	153	46	128	1				

Tabla 18. Ruta del grupo 4 con la colonia de hormigas

El tiempo total de todos los grupos sería de: 53.993 segundos, 899’89 minutos, 15 horas. El tamaño de los 4 problemas del viajante que se han resuelto es menor, y al conservar los mismos parámetros se llegan a mejores soluciones que aplicando la heurística al conjunto completo.

5.8 Algoritmo genético para cada grupo

Esta heurística fue de las que peores resultados proporcionó al conjunto completo de centros educativos, como ya se explicó, debido al alto número de centros. En esta ocasión se espera que devuelva buenas soluciones, debido a que el número de nodos disminuye, al dividir los centros educativos en 4 grupos. En esta ocasión, sí se modificaron algunos parámetros que influyen en el algoritmo, solo permaneció constante el tamaño de la población de individuos. Con lo cual para cada grupo se utilizaron una probabilidad de mutación y de recombinación diferentes.

Es destacable señalar, que tanto para la colonia de hormigas como para el algoritmo genético se realizaron pruebas de calibración con la misma semilla para establecer los valores de los diferentes parámetros.

Tiempo de la ruta 1: 9.007 segundos, 150’12 minutos, 2’51 horas.

- Tamaño de la población: 5000
- Probabilidad de mutación: 1
- Probabilidad de recombinación: 1

1	143	149	95	10	35	96	11	20	51
161	32	155	165	33	125	47	146	7	28
160	78	72	17	59	115	162	90	163	86
1									

Tabla 19. Ruta del grupo 1 con el algoritmo genético

Tiempo de la ruta 2: 18.291 segundos, 304’85 minutos, 5’09 horas.

- Tamaño de la población: 5000
- Probabilidad de mutación: 0.35
- Probabilidad de recombinación: 1

1	44	105	106	91	71	104	21	112	114
126	94	141	61	8	156	40	37	66	144
122	148	147	25	101	119	135	154	63	45
138	2	65	127	62	67	6	53	54	132
43	39	123	48	58	16	98	5	92	52
151	26	36	50	19	107	88	1		

Tabla 20. Ruta del grupo 2 con el algoritmo genético

Tiempo de la ruta 3: 17.123 segundos, 285’39 minutos, 4’76 horas.

- Tamaño de la población: 5000
- Probabilidad de mutación: 0.9
- Probabilidad de recombinación: 1

1	85	118	109	64	31	56	121	139	99
79	142	83	29	49	68	27	110	70	13
133	9	113	34	18	3	120	22	57	41
102	137	158	100	24	93	55	69	150	145
30	15	87	116	60	159	136	12	74	108

80	97	14	134	81	82	84	129	1

Tabla 21. Ruta del grupo 3 con el algoritmo genético

Tiempo de la ruta 4: 9.325 segundos, 155’42 minutos, 2’60 horas.

- Tamaño de la población: 5000
- Probabilidad de mutación: 0.85
- Probabilidad de recombinación: 1

1	22	24	12	8	13	2	18	20	5
10	14	25	23	9	4	19	7	15	11
17	3	21	6	16	1				

Tabla 22. Ruta del grupo 4 con el algoritmo genético

El tiempo total de todos los grupos sería de: 53.746 segundos, 895’77 minutos, 14’93 horas. Es muy destacable lo que ha mejorado las soluciones obtenidas con esta heurística al disminuir el número de nodos. De ser una de las que peores soluciones proporcionaba para el conjunto completo, a ser la que mejor solución ha proporcionado a cada uno de los grupos.

Por ello, a las soluciones obtenidas por el algoritmo genético se les aplicó la heurística de mejora K-opt, cuyas soluciones se presentan en el siguiente apartado.

5.9 Heurística de mejora para cada ruta

Tras la aplicación de estos algoritmos se observa que las mejores rutas se han obtenido con el algoritmo genético, a la solución de este algoritmo se le va a aplicar una heurística de mejora, k-opt.

A continuación, se presenta los resultados de la heurística de mejora K-opt, aplicada a la solución obtenida con el algoritmo genético:

Tiempo de la ruta 1: 8.915 segundos, 148’59 minutos, 2’48 horas.

1	160	78	72	17	59	115	162	90	163
86	161	20	32	155	165	51	96	11	33
125	47	95	149	10	35	143	146	7	28
1									

Tabla 23. Ruta del grupo 1 con algoritmo genético optimizado.

En la siguiente imagen se observa el orden a seguir en la ruta del grupo 1, los números representan en que orden se va a repartir el material en los centros educativos.

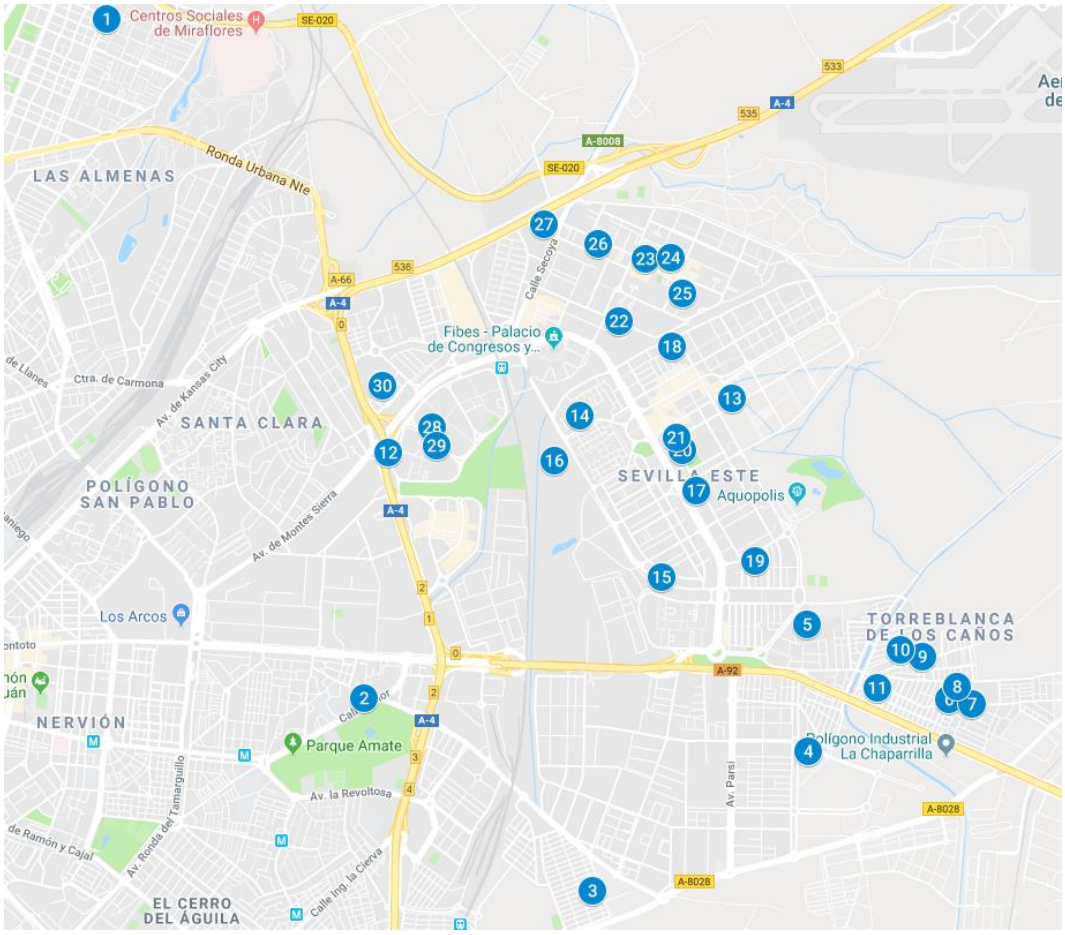

Figura 19. Localización de los centros del grupo 1. (“GoogleMaps,” n.d.)

La siguiente tabla relaciona el orden de la imagen anterior, con el nombre del centro educativo y con el índice que se le asignó en la matriz de origen/ destino.

ORDEN	TIPO DE CENTRO EDUCATIVO	NOMBRE DEL CENTRO EDUCATIVO	ÍNDICE EN LA MATRIZ DE TIEMPOS
1	'Colegio de Educación Infantil y Primaria'	Adriano	1
2	'Instituto de Educación Secundaria'	Santa Aurelia	160
3	'Colegio de Educación Infantil y Primaria'	San José de Palmete	78
4	'Colegio de Educación Infantil y Primaria'	Príncipe de Asturias	72
5	'Colegio de Educación Infantil y Primaria'	Carlos V	17
6	'Colegio de Educación Infantil y Primaria'	Menéndez Pidal	59
7	'Escuela Infantil'	Torreblanca	115
8	'Instituto de Educación Secundaria'	Siglo XXI	162
9	'Colegio de Educación Infantil y Primaria'	Vélez de Guevara	90
10	'Instituto de Educación Secundaria'	Torreblanca	163
11	'Colegio de Educación Infantil y Primaria'	Tierno Galván	86
12	'Instituto de Educación Secundaria'	Sevilla-Este	161
13	'Colegio de Educación Infantil y Primaria'	El Manantial	20
14	'Colegio de Educación Infantil y Primaria'	Isbilya	32
15	'Instituto de Educación Secundaria'	Ramón del Valle Inclán	155
16	'Instituto de Educación Secundaria'	V Centenario	165
17	'Colegio de Educación Infantil y Primaria'	Maestro José Fuentes	51
18	'Escuela Infantil'	Andaluna	96
19	'Colegio de Educación Infantil y Primaria'	Azahares	11
20	'Colegio de Educación Infantil y Primaria'	Jacarandá	33

21	'Instituto de Educación Secundaria'	Chaves Nogales	125
22	'Colegio de Educación Infantil y Primaria'	Lope de Rueda	47
23	'Escuela Infantil'	Adelfa	95
24	'Instituto de Educación Secundaria'	Pablo Picasso	149
25	'Colegio de Educación Infantil y Primaria'	Arrayanes	10
26	'Colegio de Educación Infantil y Primaria'	Concepción Estevarena	35
27	'Instituto de Educación Secundaria'	María Moliner	143
28	'Instituto de Educación Secundaria'	Miguel Servet	146
29	'Colegio de Educación Infantil y Primaria'	Ángel Ganivet	7
30	'Colegio de Educación Infantil y Primaria'	Híspalis	28
1	'Colegio de Educación Infantil y Primaria'	Adriano	1

Tabla 24. Ruta de los centros educativos del grupo 1

Tiempo de la ruta 2: 18.235 segundos, 303’92 minutos, 5’07 horas.

1	44	105	106	91	71	104	21	112	114
126	94	141	61	8	66	37	40	156	144
122	148	147	25	101	119	135	154	63	45
138	2	65	127	62	67	6	53	54	132
43	39	123	48	58	16	98	5	92	52
151	26	36	50	19	107	88	1		

Tabla 25. Ruta del grupo 2 con algoritmo genético optimizado

A continuación, se presenta la ubicación de los centros educativos del grupo 2 y el orden a seguir en el reparto del material escolar.

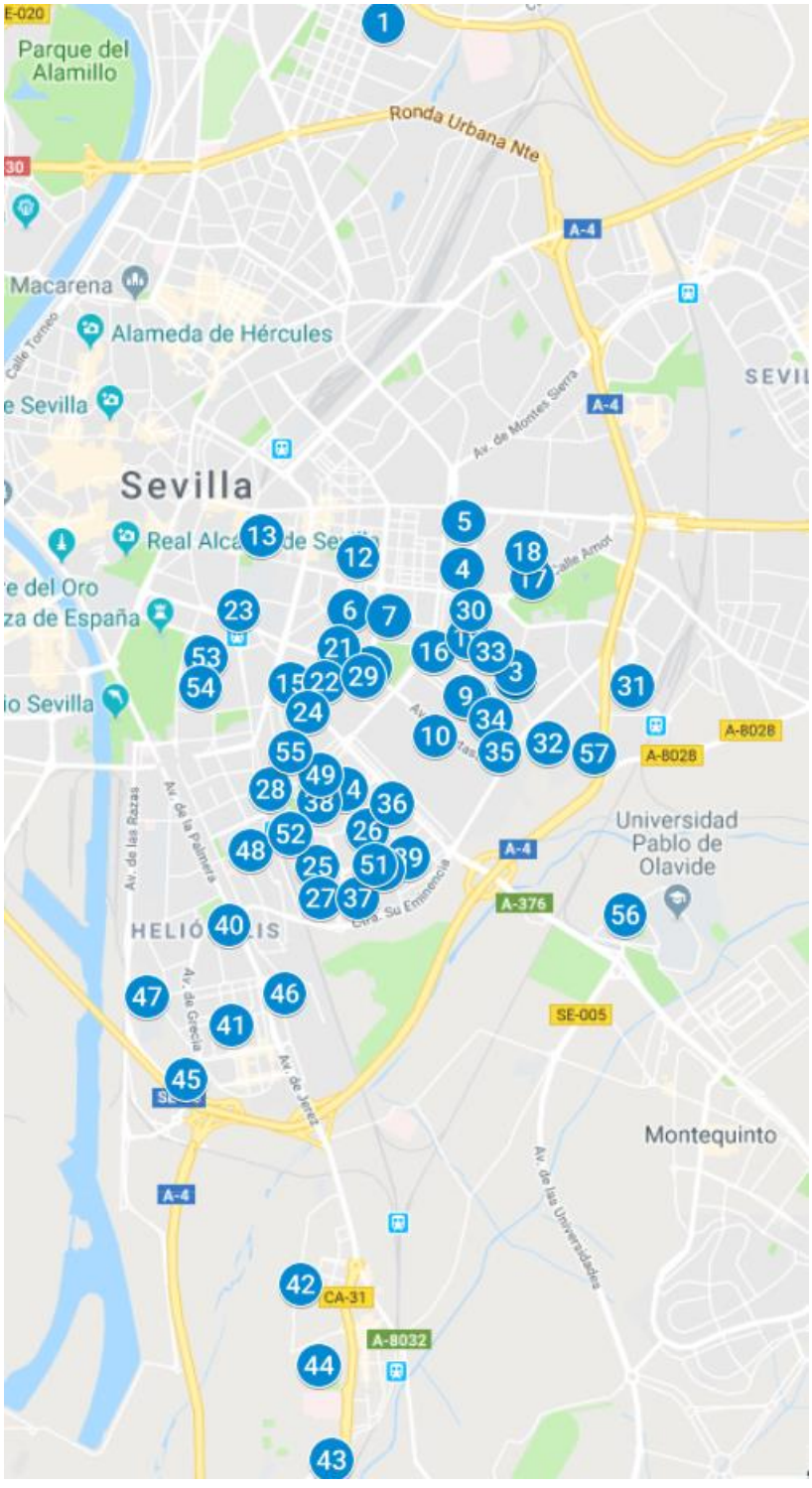

Figura 20. Localización de los centros del grupo 2. (“GoogleMaps,” n.d.)

Como en el primer grupo, ahora se presenta la tabla que relaciona los números de la imagen anterior, con los centros educativos y con el índice de la matriz de tiempos.

ORDEN	TIPO DE CENTRO EDUCATIVO	NOMBRE DEL CENTRO EDUCATIVO	ÍNDICE EN LA MATRIZ DE TIEMPOS
1	'Colegio de Educación Infantil y Primaria'	Adriano	1
2	'Colegio de Educación Infantil y Primaria'	Juan XXIII	44
3	'Escuela Infantil'	Niño Jesús	105
4	'Escuela Infantil'	Nuestra Señora de la Candelaria	106
5	'Colegio de Educación Infantil y Primaria'	Victoria Díez	91
6	'Colegio de Educación Infantil y Primaria'	Prácticas	71
7	'Escuela Infantil'	Martín de Gainza	104
8	'Colegio de Educación Infantil y Primaria'	Emilio Prados	21
9	'Escuela Infantil'	Santa María de los Ángeles	112
10	'Escuela Infantil'	Toribio de Velasco	114
11	'Instituto de Educación Secundaria'	Ciudad Jardín	126
12	'Colegio de Educación Primaria'	Cruz del Campo	94
13	'Instituto de Educación Secundaria'	Luca de Tena	141
14	'Colegio de Educación Infantil y Primaria'	Ntra.Sra. de la Paz	61
15	'Colegio de Educación Infantil y Primaria'	Aníbal González	8
16	'Colegio de Educación Infantil y Primaria'	Paulo Orosio	66
17	'Colegio de Educación Infantil y Primaria'	Jorge Juan y Antonio Ulloa	37
18	'Colegio de Educación Infantil y Primaria'	Juan de la Cueva	40
19	'Instituto de Educación Secundaria'	Salvador Távora	156
20	'Instituto de Educación Secundaria'	Martínez Montañés	144
21	'Instituto de Educación Secundaria'	Beatriz de Suabia	122

22	'Instituto de Educación Secundaria'	Nervión	148
23	'Instituto de Educación Secundaria'	Murillo	147
24	'Colegio de Educación Infantil y Primaria'	Fernán Caballero	25
25	'Escuela Infantil'	Gloria Fuertes	101
26	'Instituto de Educación Secundaria'	Antonio Domínguez Ortiz	119
27	'Instituto de Educación Secundaria'	Joaquín Romero Murube	135
28	'Instituto de Educación Secundaria'	Ramón Carande	154
29	'Colegio de Educación Infantil y Primaria'	Ortiz de Zúñiga	63
30	'Colegio de Educación Infantil y Primaria'	La Candelaria	45
31	'Instituto de Educación Secundaria'	Leonardo da Vinci	138
32	'Colegio de Educación Infantil y Primaria'	Adriano del Valle	2
33	'Colegio de Educación Infantil y Primaria'	Pablo VI	65
34	'Instituto de Educación Secundaria'	Diamantino García Acosta	127
35	'Colegio de Educación Infantil y Primaria'	Ntra.Sra. del Águila	62
36	'Colegio de Educación Infantil y Primaria'	Paz y Amistad	67
37	'Colegio de Educación Infantil y Primaria'	Andalucía	6
38	'Colegio de Educación Infantil y Primaria'	Manuel Canela	53
39	'Colegio de Educación Infantil y Primaria'	Manuel Giménez Fernández	54
40	'Instituto de Educación Secundaria'	Heliópolis	132
41	'Colegio de Educación Infantil y Primaria'	Juan Sebastián Elcano	43
42	'Colegio de Educación Infantil y Primaria'	José Sebastián y Bandarán	39
43	'Instituto de Educación Secundaria'	Bellavista	123
44	'Colegio de Educación Infantil y Primaria'	Lora Tamayo	48
45	'Colegio de Educación Infantil y Primaria'	Marie Curie	58
46	'Colegio de Educación Infantil y Primaria'	Capitán General Julio Coloma Gallegos	16
47	'Escuela Infantil'	Arco e Iris	98

48	'Colegio de Educación Infantil y Primaria'	Almotamid	5
49	'Colegio de Educación Infantil y Primaria'	Zurbarán	92
50	'Colegio de Educación Infantil y Primaria'	Manuel Altolaguirre	52
51	'Instituto de Educación Secundaria'	Polígono Sur	151
52	'Colegio de Educación Infantil y Primaria'	Fray Bartolomé de las Casas	26
53	'Colegio de Educación Infantil y Primaria'	Joaquín Turina	36
54	'Colegio de Educación Infantil y Primaria'	Maestra Isabel Álvarez	50
55	'Colegio de Educación Infantil y Primaria'	Cristóbal Colón	19
56	'Escuela Infantil'	Pablo de Olavide	107
57	'Colegio de Educación Infantil y Primaria'	Valeriano Bécquer	88
1	'Colegio de Educación Infantil y Primaria'	Adriano	1

Tabla 26. Ruta de los centros educativos del grupo 2

Tiempo de la ruta 3: 17.123 segundos, 285’39 minutos, 4’76 horas.

1	85	118	109	64	31	56	121	139	99
79	142	83	29	49	68	27	110	70	13
133	9	113	34	18	3	120	22	57	41
102	137	158	100	24	93	55	69	150	145
30	15	87	116	60	159	136	12	74	108
80	97	14	134	81	82	84	129	1	

Tabla 27. Ruta del grupo 3 con algoritmo genético optimizado

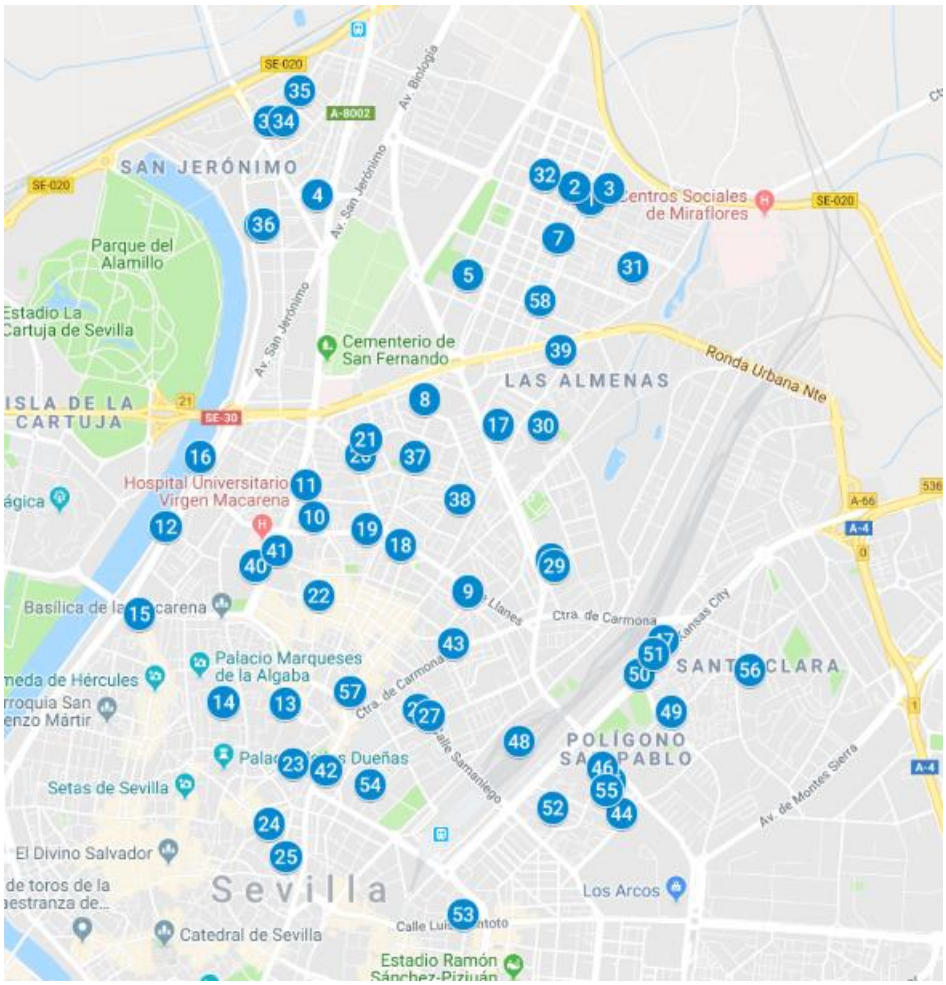

Figura 21. Localización de los centros del grupo 3. (“GoogleMaps,” n.d.)

ORDEN	TIPO DE CENTRO EDUCATIVO	NOMBRE DEL CENTRO EDUCATIVO	ÍNDICE EN LA MATRIZ DE TIEMPOS
1	'Colegio de Educación Infantil y Primaria'	Adriano	1
2	'Colegio de Educación Infantil y Primaria'	Teodosio	85
3	'Instituto de Educación Secundaria'	Albert Einstein	118
4	'Escuela Infantil'	San Jerónimo	109
5	'Colegio de Educación Infantil y Primaria'	Pablo Ruiz Picasso	64

6	'Colegio de Educación Infantil y Primaria'	Ignacio Sánchez Mejías	31
7	'Colegio de Educación Infantil y Primaria'	María Zambrano	56
8	'Instituto de Educación Secundaria'	Azahar	121
9	'Instituto de Educación Secundaria'	Llanes	139
10	'Escuela Infantil'	Argote de Molina	99
11	'Colegio de Educación Infantil y Primaria'	San José Obrero	79
12	'Instituto de Educación Secundaria'	Macarena	142
13	'Colegio de Educación Infantil y Primaria'	Sor Ángela de la Cruz	83
14	'Colegio de Educación Infantil y Primaria'	Huerta de Santa Marina	29
15	'Colegio de Educación Infantil y Primaria'	Macarena	49
16	'Colegio de Educación Infantil y Primaria'	Pedro Garfias	68
17	'Colegio de Educación Infantil y Primaria'	Hermanos Machado	27
18	'Escuela Infantil'	Santa Catalina	110
19	'Colegio de Educación Infantil y Primaria'	Pío XII	70
20	'Colegio de Educación Infantil y Primaria'	Blas Infante	13
21	'Instituto de Educación Secundaria'	Inmaculada Vieira	133
22	'Colegio de Educación Infantil y Primaria'	Arias Montano	9
23	'Escuela Infantil'	Santísima Trinidad	113
24	'Colegio de Educación Infantil y Primaria'	Jardines del Valle	34
25	'Colegio de Educación Infantil y Primaria'	Carmen Benítez	18
26	'Colegio de Educación Infantil y Primaria'	Al-Andalus	3
27	'Instituto de Educación Secundaria'	Antonio Machado	120
28	'Colegio de Educación Infantil y Primaria'	Escritor Alfonso Grosso	22
29	'Colegio de Educación Infantil y Primaria'	Mariana de Pineda	57
30	'Colegio de Educación Infantil y Primaria'	Juan de Mairena	41
31	'Escuela Infantil'	Julio César	102
32	'Instituto de Educación Secundaria'	Julio Verne	137
33	'Instituto de Educación Secundaria'	San Jerónimo	158

34	'Escuela Infantil'	Fernando Villalón	100
35	'Colegio de Educación Infantil y Primaria'	Federico García Lorca	24
36	'Colegio de Educación Primaria'	Buenavista	93
37	'Colegio de Educación Infantil y Primaria'	Manuel Siurot	55
38	'Colegio de Educación Infantil y Primaria'	Pino Flores	69
39	'Instituto de Educación Secundaria'	Pino Montano	150
40	'Instituto de Educación Secundaria'	Miguel de Cervantes	145
41	'Colegio de Educación Infantil y Primaria'	Huerta del Carmen	30
42	'Colegio de Educación Infantil y Primaria'	Calvo Sotelo	15
43	'Colegio de Educación Infantil y Primaria'	Valdés Leal	87
44	'Escuela Infantil'	Virgen de los Reyes	116
45	'Colegio de Educación Infantil y Primaria'	Miguel Hernández	60
46	'Instituto de Educación Secundaria'	San Pablo	159
47	'Instituto de Educación Secundaria'	Joaquín Turina	136
48	'Colegio de Educación Infantil y Primaria'	Baltasar de Alcázar	12
49	'Colegio de Educación Infantil y Primaria'	San Ignacio de Loyola	74
50	'Escuela Infantil'	Sagrada Familia	108
51	'Colegio de Educación Infantil y Primaria'	San Juan de Ribera	80
52	'Escuela Infantil'	Ángel de la Guarda	97
53	'Colegio de Educación Infantil y Primaria'	Borbolla	14
54	'Instituto de Educación Secundaria'	Isbilya	134
55	'Colegio de Educación Infantil y Primaria'	San Pablo	81
56	'Colegio de Educación Infantil y Primaria'	Santa Clara	82
57	'Colegio de Educación Infantil y Primaria'	Tartessos	84
58	'Instituto de Educación Secundaria'	Félix Rodríguez de la Fuente	129
1	'Colegio de Educación Infantil y Primaria'	Adriano	1

Tabla 28. Ruta de los centros educativos del grupo 3

Tiempo de la ruta 4: 9.325 segundos, 155’42 minutos, 2’59 horas.

1	22	24	12	8	13	2	18	20	5
10	14	25	23	9	4	19	7	15	11
17	3	21	6	16	1				

Tabla 29. Ruta del grupo 4 con algoritmo genético optimizado

Figura 22. Localización de los centros del grupo 4. (“GoogleMaps,” n.d.)

ORDEN	TIPO DE CENTRO EDUCATIVO	NOMBRE DEL CENTRO EDUCATIVO	ÍNDICE EN LA MATRIZ DE TIEMPOS
1	'Colegio de Educación Infantil y Primaria'	Adriano	1
2	'Instituto de Educación Secundaria'	San Isidoro	157
3	'Instituto de Educación Secundaria'	Velázquez	166
4	'Escuela Infantil'	María Inmaculada	103
5	'Colegio de Educación Infantil y Primaria'	San Isidoro	75
6	'Escuela Infantil'	Santa Luisa de Marillac	111
7	'Colegio de Educación Infantil y Primaria'	Alfares	4
8	'Instituto de Educación Secundaria'	Gustavo Adolfo Bécquer	131
9	'Instituto de Educación Secundaria'	Politécnico	152
10	'Colegio de Educación Infantil y Primaria'	Juan Ramón Jiménez	42
11	'Colegio de Educación Infantil y Primaria'	San José de Calasanz	77
12	'Escuela Infantil'	De Sevilla	117
13	'Instituto de Educación Secundaria'	Vicente Aleixandre	167
14	'Instituto de Educación Secundaria'	Triana	164
15	'Colegio de Educación Infantil y Primaria'	San Jacinto	76
16	'Colegio de Educación Infantil y Primaria'	José María del Campo	38
17	'Instituto de Educación Secundaria'	Los Viveros	140
18	'Colegio de Educación Infantil y Primaria'	Rico Cejudo	73
19	'Instituto de Educación Secundaria'	Carlos Haya	124
20	'Colegio de Educación Infantil y Primaria'	Vara del Rey	89
21	'Instituto de Educación Secundaria'	Fernando de Herrera	130
22	'Colegio de Educación Infantil y Primaria'	España	23
23	'Instituto de Educación Secundaria'	Punta del Verde	153
24	'Colegio de Educación Infantil y Primaria'	La Raza	46

25	'Instituto de Educación Secundaria'	Federico Mayor Zaragoza	128
1	'Colegio de Educación Infantil y Primaria'	Adriano	1

Tabla 30. Ruta de los centros educativos del grupo 4

El tiempo total de todos los grupos sería de: 53.598 segundos, 893’3 minutos, 14’89 horas.

El funcionamiento del algoritmo genético ha mejorado tanto para pequeños grupos de nodos, que incluso la heurística de mejora local no consigue optimizar las rutas 3 y 4.

También es reseñable el buen comportamiento de la colonia de hormigas, cuya solución para cada grupo es muy cercana en tiempo a la del algoritmo genético.

Esta última solución es la que se adopta como mejor solución del caso estudiado, la cual se caracteriza por tener 4 rutas, una para cada día. Aunque a dicha solución se le aplicará algunas modificaciones. En el siguiente apartado se presentarán las conclusiones y algunos aspectos a tener en cuenta para esta solución.

6 CONCLUSIONES

6.1 Tiempo y ruta óptima

Observando y analizando todos los resultados obtenidos tras la aplicación de todos los algoritmos, se ha llegado a la conclusión de que, primero no se puede realizar todo el reparto del material escolar en una sola ruta, y segundo se ha llegado a una solución que optimiza el tiempo y los recursos necesarios para llevar a cabo el reparto.

No se puede llevar a cabo todo el reparto en un mismo día, ya que el tiempo mínimo que se ha obtenido supera las 8 horas límite. Por ello se realizó la clasificación de los centros educativos en 4 grupos, realizándose así una ruta por día.

La mejor solución obtenida es la conseguida mediante la aplicación del algoritmo genético a cada uno de los 4 grupos de centros educativos, y la posterior aplicación de la heurística de mejora, k-opt. Dichas rutas se pueden observan en las tablas 27, 28, 29 y 30 de este proyecto. Tanto los tiempos de cada ruta como el tiempo total que aparecen junto a dichas tablas, son los tiempos del recorrido, es decir, la suma de los tiempos de ir de un punto a otro.

- Ruta 1: 8.915 segundos, 148’59 minutos, 2’48 horas
- Ruta 2: 18.235 segundos, 303’92 minutos, 5’07 horas
- Ruta 3: 17.123 segundos, 285’39 minutos, 4’76 horas
- Ruta 4: 9.325 segundos, 155’42 minutos, 2’59 horas

A estos tiempos hay que sumarles los tiempos de carga y de descarga del material en cada uno de los centros educativos. Estos tiempos fueron definidos al principio del capítulo 3 y son los siguientes:

- Tiempo de carga: 60 minutos
- Tiempo de descarga: 3 minutos por centro educativo

Al añadir estos tiempos a cada una de las 4 rutas, quedan los siguientes tiempos:

- Ruta 1: 4’93 horas
- Ruta 2: 8’87 horas
- Ruta 3: 8’61 horas
- Ruta 4: 4’79 horas

Como se puede observar en los tiempos anteriores, en las rutas 2 y 3 se sobrepasa las 8 horas de la jornada laboral, por lo que habría que buscar una solución a ese problema. Otra cuestión que sería interesante mejorar, sería equilibrar un poco más los tiempos que se dan en cada reparto diario, debido a que hay dos días con un tiempo demasiado elevado y dos días con apenas 5 horas.

Por todo esto, se estuvo pensando en cómo mejorar ambas cuestiones, había varías posibilidades, como aumentar el número de grupos en los que clasificar los centros educativos o librar del tiempo de carga a aquellos días más cargados y añadírselo a los días menos cargados.

Tras contemplar ambas alternativas, se decidió aplicar la segunda, ya que con la primera disminuiría el tiempo de los días más cargados, pero aumentaría el tiempo ocioso de los días menos cargados. Sin embargo, con la segunda propuesta, el tiempo de los días más cargados sería menor y se equilibraría los tiempos de todos los días.

Al aplicar este criterio, los 4 días de reparto quedaron de la siguiente manera:

- Primer día: Cargar la ruta 1, realizar la ruta 1 y cargar la ruta 2. Tiempo total: 5’93 horas.
- Segundo día: Realizar ruta 2. Tiempo total: 7’87 horas.
- Tercer día: Cargar ruta 4, realizar ruta 4 y cargar ruta 3. Tiempo total: 5’79 horas.
- Cuarto día: Realizar ruta 3. Tiempo total: 7’61 horas.

Como se puede apreciar, los tiempos de cada día están un poco más equilibrado y no se supera las 8 horas diarias. Al final se consigue que la carga del trabajador no tenga picos de muchas horas en un día y pocas horas en otro, que se puede completar el reparto con el mínimo tiempo posible y lo más importante, al disminuir ese tiempo también se están disminuyendo costes para la empresa. El coste disminuye entre otras cosas por el ahorro de combustible. También un aspecto importante que se consigue es aumentar la competitividad de la empresa, debido a que, si se emplea menos tiempo en este reparto, antes se comienza otros repartos, aspecto que los clientes pueden tener muy en cuenta.

6.2 Observaciones finales

Tras haber llegado a la solución, que se acaba de detallar, se puede concluir, que la aplicación del problema del viajante, ha sido una buena manera de llegar a unas rutas eficientes y que ayudan al ahorro de tiempo, costes y kilómetros.

Durante la investigación han surgido algunas observaciones que son interesantes de comentar:

- La dificultad exponencial que se da en la resolución de estos problemas cuando se aumenta el número de puntos a recorrer. Un claro ejemplo, ha sido el algoritmo genético, el cual ha proporcionado una solución pésima para 167 puntos, pero al aplicarlo a los diferentes grupos de menor tamaño sí ha mejorado mucho su solución con respecto a otros algoritmos.

- El tiempo y dinero que se puede llegar a ahorrar una empresa, con muy poca inversión, al aplicar estos algoritmos. Es decir, la diferencia en tiempo entre realizar una ruta cualquiera y una ruta estudiada es muy grande, y por ello quizás un coste demasiado elevado.

- La aplicación de diferentes algoritmos es necesaria, ya que quizás con el empleo de un solo algoritmo no se llega a una ruta lo suficientemente óptima. Por ejemplo, si solo se hubiera empleado el algoritmo genético a todo el conjunto, se estarían desperdiciando casi 3 horas de trabajo, y el coste que supondría estas 3 horas extra, con respecto a la solución de la colonia de hormigas para todo el conjunto.

- La dependencia de algunos algoritmos a cambios en ciertos parámetros. A lo largo del proyecto, se han probado distintos valores para los parámetros tanto de la colonia de hormigas como del algoritmo genético. Las soluciones variaban mucho en función del valor de los parámetros y del tamaño del problema.

- La distinta eficacia de las heurísticas según el problema. Dependiendo del tamaño del problema y de la disposición de los puntos del problema hay veces que funciona mejor un algoritmo y otras veces que funciona mejor otro. Es decir, no siempre es mejor el mismo.

Como bien se ha comentado existe una gran diferencia de eficacia y eficiencia al aplicar una heurística u otra, a continuación, se detalla para cada heurística las principales conclusiones que se han obtenido:

- Vecino más próximo: Como bien se pensaba al principio del proyecto, ha sido un algoritmo realmente fácil de implementar. También se confirmó la idea de que era un algoritmo con un tiempo computacional bajo, pero la optimalidad de la solución no es tan buena. De todos modos, ha proporcionado resultados mejores de los que se esperaba con ella. Combinar este algoritmo con alguna heurística de búsqueda de óptimo local, como se ha realizado en este proyecto, proporciona una solución eficiente y de cierta calidad. Esto último coincide con lo que se dice en Pérez, 2011, donde se analiza el comportamiento del vecino más próximo en el problema del viajante.

- Branch and Bound: Durante la presentación de los resultados obtenidos por este algoritmo se destacó que el código no era exactamente el de este algoritmo. En el código implementado se ejecuta la resolución por cortes basada en la matriz reducida al igual que en el algoritmo Branch and Bound, pero no se realiza el backtracking. No se utiliza esta técnica ya que sería ineficiente en tiempo, si se aplicara el algoritmo exacto de Branch and Bound se obtendría la solución exacta del problema, pero teniendo en cuenta el tamaño de este sería imposible obtenerla en poco tiempo. Por ello se puede concluir que es un algoritmo exacto de resolución y es recomendable para pequeños problemas, para problemas con tamaños similares al de este proyecto no es recomendable.

- Colonia de hormigas: Proporcionó la mejor solución para el conjunto completo de centros educativos y estuvo muy cerca de las mejores soluciones para cada uno de los grupos. En general, se puede decir que es una muy buena heurística para aplicar al problema del viajante. Aunque, para que su funcionamiento sea correcto y se llega a una buena solución, es necesario un buen calibrado del algoritmo para ajustar los parámetros. Además, si se compara en tiempo de ejecución con el algoritmo genético, el tiempo que emplea la colonia de hormigas es algo menor que el del algoritmo genético. Todo lo comentado sobre la importancia que ha tenido en este proyecto el valor de los parámetros y el tamaño del problema, coincide con lo que se presenta en Robles, 2010.

- Algoritmo genético: Cuando se empezó a analizar las diferentes heurísticas que se podían aplicar en este proyecto, se pensó que el algoritmo genético sería de las mejores, y así ha sido. Ha sido el algoritmo que ha proporcionado las mejores soluciones para cada grupo, en cambio ha sido de las peores para el conjunto completo de los centros educativos. Esto confirma las sospechas sobre que depende demasiado del tamaño del problema a tratar. Este aspecto, junto con su elevado tiempo computacional son los dos únicos aspectos negativos que se le pueden encontrar a esta heurística. Ya que es uno de los mejores y más potentes algoritmos que existen. Tal es así, que ha proporcionado las mejores soluciones para cada uno de los grupos. Al igual que para la colonia de hormigas, es muy importante calibrar el problema para los diferentes conjuntos de datos, ya que el valor de los parámetros influirá notablemente en su eficiencia y eficacia. La importancia de estos parámetros se puede observar en Holland, 1992.

Por último, destacar que, con la aplicación de estos resultados en el modelo de distribución del material escolar, se obtendrán los beneficios que se comentaron al principio del proyecto en el ahorro de coste y kilómetros, en las emisiones de CO2 y en la seguridad vial en el entorno de los centros educativos.

6.3 Trabajos futuros

Una posible línea de trabajo futura para mejorar aún más esta propuesta, sería la de incluir uno o varios centros de distribución, de manera que con la ayuda de estas rutas se diseñen localizaciones estratégicas donde emplazar dichos centros. La incorporación de estos centros de distribución sería necesaria, ya que en este proyecto se ha contemplado la hipótesis de que un centro educativo recibe todo el material, y es desde dicho centro, desde donde se inician todas las rutas.

En caso de existir ya un centro de distribución se podría ver si dichas rutas obtenidas pueden ser aplicadas para dicho centro. Si los costes y el tiempo empleado aumentaran demasiado al unir dichas rutas con el centro de distribución, se aplicaría los mismos algoritmos incluyendo el nuevo centro de distribución como punto de partida y de llegada.

7 REFERENCIAS

Alfred, V., John, E., & Jeffrey, D. (1987). Data Structures and Algorithms. Retrieved from http://orion.lcg.ufrj.br/Dr.Dobbs/books/book9/toc.htm

Anaya Fuentes, G. E., Hernández Gress, E. S., & Medina Marín, J. (2012). Traveling Salesman Problem Solved through Clusters and Genetic Algorithms. Universidad Autónoma Del Estado de Hidalgo.

Applegate, D. L., Bixby, R. R., Chvátal, V., & Cook, W. J. (2006). The Traveling Salesman Problem. A Computational Study. Princeton University Press, 1, 51.

Aranda Colubí, B., Olivencia, M. del C. de E., & Núñez Valdés, J. (2007). Un divertido juego inventado por un matemático infeliz. Números: Revista de Didáctica de Las Matemáticas, 66.

Bernal García, J. J., Hontoria Hernández, E., & Aleksovski,D. (2015) El problema del viajante de comercio: Búsqueda de soluciones y herramientas asequibles. Revista Electrónica de Comunicaciones y Trabajos de ASEPUMA, 16, 117-133.

Bonelli, M., & Begliardo, H. F. (2016). Optimización de armaduras planas mediante diseño paramétrico y algoritmos genéticos: Efectos de la no correspondencia objeto real – Modelo idealizado. Asociación Argentina de Mecánica Computacional, 34, 501–515.

Campos Aucejo, V. (2018). Problemas de Rutas. Universidad de Valencia

Christofides, N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem (Technical Report388). Graduate School of Industrial Administration, Carnegie Mellon University

Cook, S. A. (1971). The complexity of theorem-proving procedures. Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, 151–158.

Croes, G. . (1958). A Method for Solving Traveling-Salesman Problems. Operation Research, 6, 791–812.

Dantzig, G. B., Fulkerson, D. R., & Johnson, S. M. (1954). Solution of a Large-Scale Traveling-Salesman Problem. Operations Research Society of America, 393–410.

Der Handlungsreisende-wie er sein soll und was er zu thun hat, um Auftrage zu erhalten und eines glucklichen Erfolgs in seinen Geschaften gewiss zu sein. ein alter Commis-Voyageur. B. Fr. Voigt, Ilmenau. (1832). Reprinted Verlag Bernd Scharmm, Kiel, 1981

Dorigo, M. and Stützle, T. (2004) Ant Colony Optimization, A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England.

Glover, F. and Laguna, M. (1997). Tabu Search. Kluwer Academic Publishers, Norwell, MA

Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. Bulletin of the American Mathematical Society, 64(5), 275–278.

GoogleMaps. (n.d.). Retrieved from https://fusiontables.google.com

Kuhn, H.W. (1955). The Hungarian Method for the Assignment Problem, Naval Research Logistics Quarterly, 83–97.

Holland, J. H. (1992). Adaptation in Natural and Artificial Systems. MIT Press, 2.

Junta de Andalucía. Consejería de Educación. Junta de Andalucía. Consejería de Educación. Retrieved from: http://www.juntadeandalucia.es/educacion/vscripts/centros/_listado1.asp?qhl=SEVILLA

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. Purdue School of Engineering and Technology Indianapolis.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. Science, New Series, Vol. 220, 671–680.

Laporte, G. (2002). Some applications of the clustered travelling salesman problem. Journal of the Operational Research Society, 53, 972–976.

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA 21, 498-516.

López, E., Salas, Ó., & Murillo, Á. (2014). El problema del agente viajero: Un algoritmo determinístico usando búsqueda tabú. Revista matemática: Teoría y aplicaciones, 127-144.

MathWorks. Retrieved from https://es.mathworks.com/

Menger, K. (1931). Bericht über ein mathematisches Kolloquium. Monats-hefte für Mathematik und Physik. 38, 17-38

Muñuzuri Sanz, J. (2003). La logística urbana de mercancías: soluciones, modelado y evaluación. Universidad de Sevilla, Sevilla.

Núñez Valdés, J., Alfonso Pérez, M., Bueno Guillén, S., Diánez del Valle, M. del R., & Elías Olivenza, M. del C. de. (2004). Siete puentes, un camino: Königsberg. Suma: Revista Sobre La Enseñanza y Aprendizaje de Las Matemáticas, 45, 69–78.

Pérez Rave, J. (2011). Heurística inspirada en el análisis sistémico del "Vecino más cercano", para solucionar instancias simétricas TSP, empleando una base comparativa multicriterio. Universidad Nacional de Colombia Facultad de Minas, Escuela de Sistemas Medellín, Colombia.

Robles Algarín, C. A. (2010). Ant colony optimization: applications and trends. Ingeniería Solidaria, 6, 84–89.

Tito Chura, H. E., Silva Delgado, C. A., Alfaro Cotízales, E. E., & Fajardo Espinoza, E. (2015). Application of ant colony algorithm to the travelling salesman problem. Revista Ciencia & Desarrolla, 20, 98-102.

Vandermonde, A.-T. (1771). Remarques sûr des problèmes de situation. Academia Real de Las Ciencias de París.

ANEXO 1. COLONIA DE HORMIGAS

```
%%hormigas

clc
clear
[~, ~, matrizODCOLEGIOS] = xlsread('C:\Users\mncrm\Desktop\MIKEL\US\TFM\tiempo4.xlsx','Hoja1');

matrizODCOLEGIOS = string(matrizODCOLEGIOS);
matrizODCOLEGIOS(ismissing(matrizODCOLEGIOS)) = '';

tiempos=matrizODCOLEGIOS;
tiempos=str2double(tiempos);

%algoritmo

n_nodos=25; %%Aquí se cambia el número de nodos
iteraciones=400;
rastro=1; %importancia del rastro de la feromona
n_hormigas=300;
beta=1;%importancia de la funcion heuristica
dist_max=25000;
coef_evaporacion=0.1;
e=5;%parámetro para ponderar el rastro del mejor camino
m_heuristica=1./tiempos;%matriz heurística
rastro_feromonas=.1*ones(n_nodos,n_nodos);%matriz de feromonas



rutas=zeros(n_hormigas-1,24);% en mi caso sería el n-1

 for i = 1:n_hormigas%filas-->poblacion

     vector(randperm(24))=2:25; %%randperm números aleatorios sin que se repitan en una misma
fila para que la ciudad solo se repita una vez
     rutas(i,:)=vector;
     i=i+1;

 end

 rutas=[ones(n_hormigas,1) rutas ones(n_hormigas,1)];



 for it=1:iteraciones

     for i=1:n_hormigas
```

```
        for j=2:n_nodos-1

            nodo=rutas(i,j-1); %nodo actual
            nodo_sig=rutas(i,j:n_nodos);%nodos posteriores al actual

aux=((rastro_feromonas(nodo,nodo_sig).^rastro).*(m_heuristica(nodo,nodo_sig).^beta))./sum((rastr
o_feromonas(nodo,nodo_sig).^rastro).*(m_heuristica(nodo,nodo_sig).^beta));

            r=rand;

            for z=1:length(aux)
                if r<sum(aux(1:z))
                    nuevo_nodo=j-1+z;% siguiente ciudad en la ruta
                    break
                end
            end

            no=rutas(i,nuevo_nodo);
            rutas(i,nuevo_nodo)=rutas(i,j);
            rutas(i,j)=no;
            %colocar nueva ciudad en su posicion

        end

    end

    feromonas_temp=zeros(n_nodos,n_nodos);

    for i=1:n_hormigas

        dist(i,1)=0;

        for d=1:n_nodos

            dist(i,1)=dist(i)+tiempos(rutas(i,d),rutas(i,d+1));

feromonas_temp(rutas(i,d),rutas(i,d+1))=feromonas_temp(rutas(i,d),rutas(i,d+1))+n_nodos/dist(i,1
);

        end

    end

    [t_min pos]=min(dist);
    if t_min<dist_max
        dist_max=t_min;
        rutas_opt=rutas(pos,:);
    end

    %feromonas del mejor recorrido

    feromonas_ruta=zeros(n_nodos,n_nodos);
    for i=1:n_nodos

        feromonas_ruta(rutas(pos,i),rutas(pos,i+1))=n_nodos/dist_max;

    end
```

```
    %actualiza feromonas

    rastro_feromonas=(1-coef_evaporacion)*rastro_feromonas+feromonas_temp+e*feromonas_ruta;

 end
 dist_max
rutas_opt
```

ANEXO 2. VECINO MÁS PRÓXIMO

```
% ALGORITMO DEL VECINO MÁS PRÓXIMO

clear;
clc;
% Import the data
% Import the data
% [~, ~, matrizODCOLEGIOS] =
xlsread('C:\Users\mncrm\Desktop\MIKEL\US\TFM\matrizOD_COLEGIOS.xlsx','Hoja1');
%
% matrizODCOLEGIOS = string(matrizODCOLEGIOS);
% matrizODCOLEGIOS(ismissing(matrizODCOLEGIOS)) = '';
%
% tiempos=matrizODCOLEGIOS;
% tiempos=str2double(tiempos);


[~, ~, matrizODCOLEGIOS] = xlsread('C:\Users\mncrm\Desktop\MIKEL\US\TFM\tiempo4.xlsx','Hoja1');

matrizODCOLEGIOS = string(matrizODCOLEGIOS);
matrizODCOLEGIOS(ismissing(matrizODCOLEGIOS)) = '';

tiempos=matrizODCOLEGIOS;
tiempos=str2double(tiempos);



for i=1:25
    for j=25
        if i==j
tiempos(i,j)= 0;
        end
    end
end
%inicio algoritmo
t=0;
vecino=1;
ruta=[1];
tiempos_vecino=[];
for i=1:24
    for j=1:25
        saltar=find(ruta==j);
        saltar=isempty(saltar);
       if saltar==1
        tiempos_vecino=[tiempos_vecino tiempos(vecino,j)];
       else
           tiempos_vecino=[tiempos_vecino 0];
        j=j+1;
       end
    end
```

```
    t_0=tiempos_vecino(tiempos_vecino~=0);
    t_vecino =min(t_0);
    vecino=find(tiempos_vecino==t_vecino);
    vecino=vecino(1);
    ruta=[ruta vecino];

    t=t+t_vecino;


    i=i+1;
    tiempos_vecino=[];
end
 ruta
  t
```

ANEXO 3. BRANCH AND BOUND

```
% branch and bound
% clear;
% clc;
% Import the data
% Import the data
[~, ~, matrizODCOLEGIOS] =
xlsread('C:\Users\mncrm\Desktop\MIKEL\US\TFM\matrizOD_COLEGIOS.xlsx','Hoja1');

matrizODCOLEGIOS = string(matrizODCOLEGIOS);
matrizODCOLEGIOS(ismissing(matrizODCOLEGIOS)) = '';

tiempos=matrizODCOLEGIOS;
tiempos=str2double(tiempos);
```

```
for i=1:167
    for j=1:167
        if i==j
tiempos(i,j)= 0;
        end
    end
end
```

```
m_tiempos=tiempos;
m_tiempos(m_tiempos==0)=inf;
red_fil=min(m_tiempos,[],2);
red_fil(isinf(red_fil))=0;
m_tiempos_red_fil=m_tiempos-red_fil;
red_col=min(m_tiempos_red_fil);
m_tiempos_red=m_tiempos_red_fil-red_col;
m_tiempos_red(isnan(m_tiempos_red))=inf;
t1=sum(red_col)+sum(red_fil);

vector=[1:167];
tam=0;
i=1;
ruta=[1];

while tam<167 %n
    vector_t=zeros(1,167);
    i_0=i;
    for j=2:167

        if vector(j)~=inf
        reducida=m_tiempos_red;
        reducida(i,:)=inf;
        reducida(:,j)=inf;
        reducida(j,i)=inf;
        red_fil=min(reducida,[],2);
        red_fil(isinf(red_fil))=0;
        reducida_fil=reducida-red_fil;
        red_col=min(reducida_fil);
```

```
        reducida=reducida_fil-red_col;
        reducida(isnan(reducida))=inf;
        red_fil(red_fil==inf)=0;
        red_col(red_col==inf)=0;
        t=t1+sum(red_col)+sum(red_fil)+m_tiempos_red(i,j);
        tiempos(i,j);
        vector_t(j)=t;
        t=0;
        end
        j=j+1;
    end
    vector_t(vector_t==0)=inf;
    [coste pos]=min(vector_t);
    i=pos;
    if coste~=inf
    t1=coste;
    ruta=[ruta i];

    end
    vector(i)=inf;
    aux=find(vector==inf);
    [tami tamj]=size(aux);
    tam=tamj;
    reducida=m_tiempos_red;
        reducida(i_0,:)=inf;
        reducida(:,i)=inf;
        reducida(i,i_0)=inf;
        red_fil=min(reducida,[],2);
        red_fil(isinf(red_fil))=0;
        reducida_fil=reducida-red_fil;
        red_col=min(reducida_fil);
        reducida=reducida_fil-red_col;
        reducida(isnan(reducida))=inf;
        m_tiempos_red=reducida;
end

t1
ruta=[ruta 1]
```

ANEXO 4. ALGORITMO GENÉTICO

- **Función principal**

```
clear;

clc;

% Import the data
[~, ~, matrizODCOLEGIOS] =
xlsread('C:\Users\mncrm\Desktop\MIKEL\US\TFM\tiempo4.xlsx','Hoja1');

matrizODCOLEGIOS = string(matrizODCOLEGIOS);
matrizODCOLEGIOS(ismissing(matrizODCOLEGIOS)) = '';

tiempos=matrizODCOLEGIOS;
tiempos=str2double(tiempos);
tam_poblacion=100; % Se fue cambiando según el tamaño del problema

menor_valor=30000;
% while menor_valor>9600
[poblacion]=pob_inicial(tam_poblacion);%creación población inicial

for iteracion=1:15




prob_recombinacion=0.2; %Probabilidad para la recombinación
prob_mutacion=0.9; %Probabilidad para la mutación
iter=0;
parar=false;
r=0;



[fun_objetivo]=funObjetivo(poblacion,tiempos,tam_poblacion);%creación de la función
objetivo

%Ahora se abre ciclo hasta cumplir dos condiciones: que el menor valor no
%cambie en 2n generaciones y que llegue a 1000 generaciones

while parar==false

    [num_descendientes]=sel_torneo(fun_objetivo,tam_poblacion);%proceso de selección
por torneo

[poblacion]=recombinacion(num_descendientes,poblacion,prob_recombinacion,tam_poblacion
);%nueva población
    [fun_objetivo]=funObjetivo(poblacion,tiempos,tam_poblacion);%nueva f.obj
```

```
    [poblacion]=mutacion(poblacion,prob_mutacion,tam_poblacion);%mutación
    [fun_objetivo]=funObjetivo(poblacion,tiempos,tam_poblacion);%f.obj de la nueva
población si hay mutación
    [nuevo_valor posicion]=min(fun_objetivo);

    if menor_valor > nuevo_valor
        menor_valor = nuevo_valor;

        x=poblacion(posicion,:);
    end
    % primera condición de parada del bucle while
    if menor_valor==nuevo_valor;
        r=r+1;
        if r==1000 %2n
            parar=true;
        end
    end

    %segunda condición de parada del bucle while
    if iter==1000
        parar=true;
    end
    iter=iter+1;
end
tiempo_ruta=menor_valor
ruta=x;
tiempos_ruta(iteracion,1)=tiempo_ruta;
rutas(iteracion,:)=ruta;

end
[tiempo_ruta pos_optimo]=min(tiempos_ruta);
tiempo_ruta
ruta_optima=[1 rutas(pos_optimo,:) 1]
 %end
```

- Población inicial

```
-   function [poblacion]=pob_inicial(tam_poblacion,ruta)
-
-   %En esta función se crea las poblaciones iniciales a partir de la cual
-   %trabajar, cada población tendrá una ruta aleatoria
-
-   %s=rng(randperm(1000,1))%utiliza la misma semilla para comprobar cálculos del
    algoritmo, es decir, mismos numeros aleatorios,
-   poblacion=zeros(tam_poblacion-1,24);% el n-1
-   %s=rng(938)
-
-
-   for i = 1:tam_poblacion%filas-->población
-
-       vector(randperm(24))=2:25; %%randperm números aleatorios sin que se
    repitan en una misma fila para que la ciudad solo se repita una vez
-       poblacion(i,:)=vector;
-       i=i+1;
```

```
-    end
-
-
-   end
```

- Función objetivo

```
function[fun_objetivo]=funObjetivo(poblacion,tiempos,tam_poblacion)

%En esta función se va a evaluar los tiempos de cada uno de los caminos generados en
pob_inicial

fun_objetivo=zeros(tam_poblacion,1);% vector columna que tendrá los tiempos de cada
ruta creada anteriormente
p=[ones(tam_poblacion,1) poblacion ones(tam_poblacion,1)];%A las poblaciones creadas
en pob_inical se le añade la ciudad de entrada y salida.
t=0;
for i=1:tam_poblacion
    for j=1:25 % n
         t=t+tiempos(p(i,j),p(i,j+1));
         j=j+1;
    end

    fun_objetivo(i,1)=t;
    t=0;
    i=i+1;
end
end
```

- Selección de torneo

```
function [num_descendientes]=sel_torneo(fun_objetivo,tam_poblacion)
% esta selección esta basada en realizar torneos entre k individuos y
% escoger del de mejor f.objetivo
%voy a escoger torneo entre dos individuos
k=2;
comp=zeros(k,2);
num_descendientes=zeros(tam_poblacion,1);% vector columna con el número de
descendientes
while sum(num_descendientes)<tam_poblacion
    i=1;
    while i<=k
        al=floor((tam_poblacion)*rand+1);% se escoge un camino arbitrariamente
        if al ~= comp(:,2) %para que no compare el mismo camino
            comp(i,:)=[fun_objetivo(al,1) al]; %f.obj y camino
            i=i+1;
        end
    end
    [min_comp pob]=min(comp(:,1)); %mínimo del torneo y su camino
```

```
    num_descendientes(comp(pob,2),1)=num_descendientes(comp(pob,2),1)+1;
    %cada camino o padre tendrá unos descendientes, aquí se ve cuantos
    %descendientes tiene cada camino
end
end
```

- Recombinación

```
function
[poblacion]=recombinacion(num_descendientes,poblacion,prob_recombinacion,tam_poblacion
)
%Se va a utilizar el método de recombinación de un punto, para los caminos
%ganadores en la selección de torneo se van a  combinar , es decir, cada
%camino va a mantener sus primeras rutas y a partir de cierto punto tendrá
%las rutas de otro camino.

caminos_opt=zeros(tam_poblacion,24);% matriz que tendrá a los caminos más óptimos
obtenidos en la selección de torneo
i=1;
% ahora se guarda en esa matriz anterior los caminos con sus rutas
while max(num_descendientes)~= 0
    [descendientes camino]=max(num_descendientes);
    for aux=1:descendientes
        caminos_opt(i,:)=poblacion(camino,:);
        i=i+1;
    end
    num_descendientes(camino,1)=0;
end
x=[];
i=0;
while size(caminos_opt,1)~=0 %hasta recorrer todos los caminos más óptimos
    camino1=caminos_opt(1,:); %primer camino
    al=floor((size(caminos_opt,1)-1)*rand+2); %para elegir otro camino
    camino2=caminos_opt(al,:);%segundo camino

    caminos_opt([1 al],:)=[];%se borran ambos caminos


    if prob_recombinacion>=rand % si se recombinan crear otros caminos combinación de
ambos
        punto_recomb=floor((3)*rand+1);
        c1=camino1;
        c2=camino2;

        for j=1:punto_recomb

            parte1=find(c2(1,:)==camino1(1,j));
            if parte1~=0
                c2(parte1)=[];
            end

            parte2=find(c1(1,:)==camino2(1,j));
            if parte2~=0
```

```
                c1(parte2)=[];
            end
        end

    i=i+1;
    x(i,:)=[camino1(1,1:punto_recomb) c2];
    i=i+1;
    x(i,:)=[camino2(1,1:punto_recomb) c1];

    else %si no se recombinan dejar los mismos caminos
        i=i+1;
        x(i,:)=camino1;
        i=i+1;
        x(i,:)=camino2;
    end
end
poblacion=x;
end
```

- Mutación

```
- function [poblacion]=mutacion(poblacion,prob_mutacion,tam_poblacion);
-
- %en este apartado se va a cambiar algún camino al azar para podrá alcanzar
- %una mejor f.obj que no se haya detectado en la recombinación. Se le asigna
- %una probabilidad de mutación baja por lo que pocas veces se va a dar una
- %transformación de un camino.
-
- %para un camino elegido al azar se intercambian el orden de dos de sus
- %ciudades
- if prob_mutacion>= rand
-     camino_muta=floor((tam_poblacion)*rand+1);
-     ciudad1=floor(24*rand+1);
-     ciudad2=ciudad1;
-     while ciudad1==ciudad2
-         ciudad2=floor(24*rand+1);
-     end
-     aux=poblacion(camino_muta,ciudad1);
-     poblacion(camino_muta,ciudad1)=poblacion(camino_muta,ciudad2);
-     poblacion(camino_muta,ciudad2)=aux;
- end
-
- end
```

```
% k-opt


clc
```

```
clear

% Import the data
[~, ~, matrizODCOLEGIOS] =
xlsread('C:\Users\mncrm\Desktop\MIKEL\US\TFM\tiempo4.xlsx','Hoja1');

matrizODCOLEGIOS = string(matrizODCOLEGIOS);
matrizODCOLEGIOS(ismissing(matrizODCOLEGIOS)) = '';

tiempos=matrizODCOLEGIOS;
tiempos=str2double(tiempos);



n=25;
iteraciones=5000;
t_act=0;
%el siguiente vector es el óptimo del vecino más próximo
%si pongo el vector óptimo una vez aplicada el 2-opt el tiempo disminuye un
%poco 13141 el que menos tiempo dio
 ruta_opt=[1,22,24,12,8,13,2,18,15,11,10,5,7,14,25,23,9,4,20,19,17,3,21,6,16,1];
for i=1:n

            t_act=t_act+tiempos(ruta_opt(i),ruta_opt(i+1));

end


for iter=1:iteraciones
       %nodos que se van a intercambiar, invirtiendo así parte del vector
       %ruta
    nodo_cambio1=2+(10-2)*rand;
    nodo_cambio1=floor(nodo_cambio1)
    nodo_cambio2=7+(25-7)*rand;
    nodo_cambio2=floor(nodo_cambio2)
    if nodo_cambio1>nodo_cambio2
        a=nodo_cambio1;
        nodo_cambio1=nodo_cambio2;
        nodo_cambio2=a;
    end
    if nodo_cambio1~=nodo_cambio2
        nueva=ruta_opt(nodo_cambio1:nodo_cambio2);%parte del vector ruta que se va
invertir
        aux=nodo_cambio2-nodo_cambio1+1;
        nueva=nueva(aux:-1:1);%vector invertido
        nueva_ruta=[ruta_opt(1:nodo_cambio1-1) nueva ruta_opt(nodo_cambio2+1:n+1)];
       t1=0;

        for i=1:n
            t1=t1+tiempos(nueva_ruta(i),nueva_ruta(i+1));

        end
        t1
        if t1<=t_act
            t_act=t1;
            ruta_opt=nueva_ruta;
```

```
        end

    end
    iter=iter+1;
end
ruta_opt
t_act
```

ANEXO 5. K-OPT

```
% k-opt


clc
clear

% Import the data
[~, ~, matrizODCOLEGIOS] =
xlsread('C:\Users\mncrm\Desktop\MIKEL\US\TFM\tiempo4.xlsx','Hoja1');

matrizODCOLEGIOS = string(matrizODCOLEGIOS);
matrizODCOLEGIOS(ismissing(matrizODCOLEGIOS)) = '';

tiempos=matrizODCOLEGIOS;
tiempos=str2double(tiempos);



n=25;
iteraciones=5000;
t_act=0;
%el siguiente vector es el óptimo del vecino más próximo
%si pongo el vector óptimo una vez aplicada el 2-opt el tiempo disminuye un
%poco 13141 el que menos tiempo dio
 ruta_opt=[1,22,24,12,8,13,2,18,15,11,10,5,7,14,25,23,9,4,20,19,17,3,21,6,16,1];
for i=1:n

            t_act=t_act+tiempos(ruta_opt(i),ruta_opt(i+1));

end


for iter=1:iteraciones
       %nodos que se van a intercambiar, invirtiendo así parte del vector
       %ruta
    nodo_cambio1=2+(10-2)*rand;
    nodo_cambio1=floor(nodo_cambio1)
    nodo_cambio2=7+(25-7)*rand;
    nodo_cambio2=floor(nodo_cambio2)
    if nodo_cambio1>nodo_cambio2
        a=nodo_cambio1;
        nodo_cambio1=nodo_cambio2;
        nodo_cambio2=a;
    end
    if nodo_cambio1~=nodo_cambio2
        nueva=ruta_opt(nodo_cambio1:nodo_cambio2);%parte del vector ruta que se va
invertir
        aux=nodo_cambio2-nodo_cambio1+1;
        nueva=nueva(aux:-1:1);%vector invertido
```

```
        nueva_ruta=[ruta_opt(1:nodo_cambio1-1) nueva ruta_opt(nodo_cambio2+1:n+1)];
       t1=0;

        for i=1:n
            t1=t1+tiempos(nueva_ruta(i),nueva_ruta(i+1));

        end
        t1
        if t1<=t_act
            t_act=t1;
            ruta_opt=nueva_ruta;
        end

    end
    iter=iter+1;
end
ruta_opt
t_act
```

ANEXO 6. CLUSTERING

```
clear

clc

opts = spreadsheetImportOptions("NumVariables", 2);

% Specify sheet and range
opts.Sheet = "Sheet1";
opts.DataRange = "N2:O168";

% Specify column names and types
opts.VariableNames = ["LATITUD", "LONGITUD"];
opts.VariableTypes = ["double", "double"];

% Import the data
listadoCOLEGIOS =
readtable("C:\Users\mncrm\Desktop\MIKEL\US\TFM\listadoCOLEGIOS.xlsx", opts,
"UseExcel", false);
listadoCOLEGIOS = table2array(listadoCOLEGIOS);

clear  opts




%%%%%%%%%%%%%%%%%%

plot(listadoCOLEGIOS(:,1),listadoCOLEGIOS(:,2),'.')
hold on
plot(listadoCOLEGIOS(1,1),listadoCOLEGIOS(1,2),'*')



% Solve a Clustering Problem with a Self-Organizing Map
% Script generated by Neural Clustering app
% Created 29-Oct-2018 20:30:53
%
% This script assumes these variables are defined:
%
%   listadoCOLEGIOS - input data.

x = listadoCOLEGIOS';

% Create a Self-Organizing Map
dimension1 = 2;
dimension2 = 2;
net = selforgmap([dimension1 dimension2]);

% Train the Network
```

```
[net,tr] = train(net,x);

% Test the Network
y = net(x);

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
figure, plotsomtop(net)
figure, plotsomnc(net)
figure, plotsomnd(net)
figure, plotsomplanes(net)
figure, plotsomhits(net,x)
figure, plotsompos(net,x)



[~, ~, matrizODCOLEGIOS] =
xlsread('C:\Users\mncrm\Desktop\MIKEL\US\TFM\matrizOD_COLEGIOS.xlsx','Hoja1');

matrizODCOLEGIOS = string(matrizODCOLEGIOS);
matrizODCOLEGIOS(ismissing(matrizODCOLEGIOS)) = '';

tiempos=matrizODCOLEGIOS;
tiempos=str2double(tiempos);

%4 grupos

grupo1=[1];
grupo2=[1];
grupo3=[];
grupo4=[1];
for j=1:167
    if y(1,j)==1
        grupo1=[grupo1 j];
    end
     if y(2,j)==1
         grupo2=[grupo2 j];
     end
     if y(3,j)==1
         grupo3=[grupo3 j];
     end
     if y(4,j)==1
         grupo4=[grupo4 j];
    end
end
%
%matriz de tiempos
%grupo1
t1=size(grupo1);
t1=t1(2);
for i=1:t1
```

```
    for j=1:t1
    tiempo1(i,j)=tiempos(grupo1(i),grupo1(j));
    end
end

%grupo2
t2=size(grupo2);
t2=t2(2);
for i=1:t2
    for j=1:t2
    tiempo2(i,j)=tiempos(grupo2(i),grupo2(j));
    end
end

%grupo3
t3=size(grupo3);
t3=t3(2);
for i=1:t3
    for j=1:t3
    tiempo3(i,j)=tiempos(grupo3(i),grupo3(j));
    end
end

%grupo4
t4=size(grupo4);
t4=t4(2);
for i=1:t4
    for j=1:t4
    tiempo4(i,j)=tiempos(grupo4(i),grupo4(j));
    end
end

xlswrite('tiempo1.xlsx',tiempo1);
xlswrite('tiempo2.xlsx',tiempo2);
xlswrite('tiempo3.xlsx',tiempo3);
xlswrite('tiempo4.xlsx',tiempo4);
```

ANEXO 7. RELACIÓN ÍNDICE-COLEGIO

```
clc

clear

%algoritmo para relacionar cada índice del clúster con el colegio
%correspondiente
grupo1=[1,7,10,11,17,20,28,32,33,35,47,51,59,72,78,86,90,95,96,115,125,143,146,149,155
,160,161,162,163,165]
grupo2=[1,2,5,6,8,16,19,21,25,26,36,37,39,40,43,44,45,48,50,52,53,54,58,61,62,63,65,66
,67,71,88,91,92,94,98,101,104,105,106,107,112,114,119,122,123,126,127,132,135,138,141,
144,147,148,151,154,156]
grupo3=[1,3,9,12,13,14,15,18,22,24,27,29,30,31,34,41,49,55,56,57,60,64,68,69,70,74,79,
80,81,82,83,84,85,87,93,97,99,100,102,108,109,110,113,116,118,120,121,129,133,134,136,
137,139,142,145,150,158,159]
grupo4=[1,4,23,38,42,46,73,75,76,77,89,103,111,117,124,128,130,131,140,152,153,157,164
,166,167]
[~, ~, matrizODCOLEGIOS] =
xlsread('C:\Users\mncrm\Desktop\MIKEL\US\TFM\matrizOD_COLEGIOS.xlsx','Hoja1');

 matrizODCOLEGIOS = string(matrizODCOLEGIOS);
 matrizODCOLEGIOS(ismissing(matrizODCOLEGIOS)) = '';

 tiempos=matrizODCOLEGIOS;
 tiempos=str2double(tiempos);
%tamaño grupo 1 es 30
%tamaño grupo 2 es 57
%tamaño grupo 3 es 58
%tamaño grupo 4 es 25
t_ruta_1=0;
t_ruta_2=0;
t_ruta_3=0;
t_ruta_4=0;
%introduzco ruta obtenida por cada algoritmo de resolución
ruta1=[1,22,10,3,18,24,6,8,25,30,27,12,19,4,9,21,11,23,2,7,26,15,14,5,13,20,28,17,29,1
6];
ruta2=[1,50,27,47,25,29,4,21,22,48,15,13,45,18,23,6,35,40,31,16,38,39,32,26,17,46,34,5
1,24,53,30,28,52,44,54,19,11,9,7,36,43,49,56,20,55,10,3,33,5,37,8,41,42,57,14,12,2];
ruta3=[1,33,45,11,53,37,27,54,31,34,43,15,8,2,46,58,21,44,6,36,29,26,40,28,51,4,9,20,1
6,39,52,14,19,47,42,25,5,49,3,7,50,55,13,12,17,23,18,24,56,30,32,22,41,35,10,38,57,48]
;
ruta4=[1,22,24,12,8,13,2,18,20,19,7,15,11,5,10,14,25,23,9,4,3,17,21,6,16];

%grupo1
for i=1:30
    ruta_optima_cluster1(i)=grupo1(ruta1(i));

end
ruta_optima_cluster1=[ruta_optima_cluster1 1];
for i=1:30
    t_ruta_1=t_ruta_1+tiempos(ruta_optima_cluster1(i),ruta_optima_cluster1(i+1));
```

```
end
%grupo 2
for i=1:57
    ruta_optima_cluster2(i)=grupo2(ruta2(i));
end

ruta_optima_cluster2=[ruta_optima_cluster2 1];
for i=1:57
    t_ruta_2=t_ruta_2+tiempos(ruta_optima_cluster2(i),ruta_optima_cluster2(i+1));
end

%grupo3
for i=1:58
    ruta_optima_cluster3(i)=grupo3(ruta3(i));
end

ruta_optima_cluster3=[ruta_optima_cluster3 1];
for i=1:58
    t_ruta_3=t_ruta_3+tiempos(ruta_optima_cluster3(i),ruta_optima_cluster3(i+1));
end

%grupo 4
for i=1:25
    ruta_optima_cluster4(i)=grupo4(ruta4(i));
end

ruta_optima_cluster4=[ruta_optima_cluster4 1];
for i=1:25
    t_ruta_4=t_ruta_4+tiempos(ruta_optima_cluster4(i),ruta_optima_cluster4(i+1));
end
t_ruta_1
ruta_optima_cluster1
t_ruta_2
ruta_optima_cluster2
t_ruta_3
ruta_optima_cluster3
t_ruta_4
ruta_optima_cluster4
```

ANEXO 8. MATRIZ DE TIEMPOS

```
% IMPORTAR EXCEL colegios, a través de la opción IMPORT DATA/IMPORT
SELECTION/GENERATE SCRIPT
clear
clc
 format long

% Setup the Import Options
opts = spreadsheetImportOptions("NumVariables", 13);

% Specify sheet and range
opts.Sheet = "colegios_sevilla";
opts.DataRange = "A2:M168";

% Specify column names and types
opts.VariableNames = ["Cdigo", "Denominacin", "Nombre", "Dependencia", "Domicilio",
"Localidad", "Municipio", "Provincia", "CdPostal", "Telfono", "Enseanzas",
"Servicios", "Programas"];
opts.VariableTypes = ["double", "categorical", "string", "categorical", "string",
"categorical", "categorical", "categorical", "double", "double", "categorical",
"categorical", "categorical"];
opts = setvaropts(opts, [3, 5], "WhitespaceRule", "preserve");
opts = setvaropts(opts, [2, 3, 4, 5, 6, 7, 8, 11, 12, 13], "EmptyFieldRule", "auto");

% Import the data
listadoCOLEGIOS =
readtable("C:\Users\mncrm\Desktop\MIKEL\US\TFM\listado_COLEGIOS.xlsx", opts,
"UseExcel", false);



% SOLO COLUMNA DE CALLES
COLEGIOS= listadoCOLEGIOS(:,2);
COLEGIOS = table2array(COLEGIOS);

NOMBRES= listadoCOLEGIOS(:,3);
NOMBRES = table2array(NOMBRES);

CALLE= listadoCOLEGIOS(:,9);
CALLE = table2array(CALLE);
  %%PARA CADA CALLE HAGO LA DIRECCIÓN DE LA PETICIÓN A GOOGLE
 for i=1:167

     COLEGIOS(i)= strrep(COLEGIOS(i),' ','+');

%      CALLE_COLEGIOS(i,)= strrep(CALLE_COLEGIOS(i),'C/','Calle');
     COLEGIOS(i)= strrep(COLEGIOS(i),'ñ','n');
     COLEGIOS(i)= strrep(COLEGIOS(i),'','o');
     NOMBRES(i)= strrep(NOMBRES(i),' ','+');
      NOMBRES(i)= strrep(NOMBRES(i),'ñ','n');
      NOMBRES(i)= strrep(NOMBRES(i),'í','i');
```

```
      NOMBRES(i)= strrep(NOMBRES(i),'á','a');
      NOMBRES(i)= strrep(NOMBRES(i),'ó','o');
      NOMBRES(i)= strrep(NOMBRES(i),'é','e');
      NOMBRES(i)= strrep(NOMBRES(i),'ú','u');
       CALLE(i)= strrep(CALLE(i),'ñ','n');
      CALLE(i)= strrep(CALLE(i),'í','i');
      CALLE(i)= strrep(CALLE(i),'á','a');
      CALLE(i)= strrep(CALLE(i),'ó','o');
      CALLE(i)= strrep(CALLE(i),'é','e');
      CALLE(i)= strrep(CALLE(i),'ú','u');
    i=i+1;
 end
%cambio a colegio publico
for i=1:92
    COLEGIOS(i)='Colegio Publico';
    i=i+1;
end
%
 for i=1:167
      for j=1:167
          if i~=j
     %%QUITO LAS COMILLAS
     A1=categorical(COLEGIOS(i));
     B1=categorical(NOMBRES(i));
     C1=categorical(CALLE(i));
     A2=categorical(COLEGIOS(j));
     B2=categorical(NOMBRES(j));
     C2=categorical(CALLE(j));

     A1=cellstr(A1);
     B1=cellstr(B1);
     C1=cellstr(C1);
     A2=cellstr(A2);
     B2=cellstr(B2);
     C2=cellstr(C2);
%     % PET={'https://maps.googleapis.com/maps/api/geocode/json?address='};
%     API DE GOOGLE
    pet_tiempo={'https://www.google.es/maps/dir/'};
     %%introduzco sevilla
     sev={'Sevilla#'};

pet_tiempo=strcat(pet_tiempo,A1,'+',B1,'+',C1,'+','SEVILLA','/',A2,'+',B2,'+',C2,'+','
SEVILLA','/');


%       pet_tiempo={'https://www.google.es/maps/dir/'};
%
pet_tiempo=strcat(pet_tiempo,DIRECCION_LAT(i),',',DIRECCION_LON(i),'/',DIRECC
ION_LAT(j),',',DIRECCION_LON(j),'/');
     pet_tiempo=char(pet_tiempo);
     tiempo=urlread(pet_tiempo);

     %API DE GOOGLE
     %PARA EVITAR QUE SE QUEDE EN EL LIMITE DE PETICIONES
%      l = strfind(tiempo,'OVER_QUERY_LIMIT');
%      while l>0
```

```
%          tiempo=urlread(pet_tiempo)
 %       l = strfind(tiempo,'OVER_QUERY_LIMIT');

      [tiempo,matches] = strsplit(tiempo,{'m\'},'CollapseDelimiters',true);
          tiempo=tiempo(6);
      tiempo=char(tiempo);
% % % % %
          [tiempo,matches] = strsplit(tiempo,{'\'},'CollapseDelimiters',true);
               tiempo=tiempo(2);
        tiempo=char(tiempo);
          [tiempo,matches] = strsplit(tiempo,{','},'CollapseDelimiters',true);
           tiempo=tiempo(2);
%      tiempo=char(tiempo);
          matrizOD(i,j)=tiempo;
      end
      j=j+1
    end
      i=i+1
  end
 xlswrite('matrizOD_COLEGIOS.xlsx',matrizOD);
```

- Limpieza de de los tiempos

```
opts = spreadsheetImportOptions("NumVariables", 167);

% Specify sheet and range
opts.Sheet = "Hoja1";
opts.DataRange = "A1:FK167";

% Specify column names and types
opts.VariableNames = ["VarName1", "VarName2", "VarName3", "VarName4",
"VarName5", "VarName6", "VarName7", "VarName8", "VarName9", "VarName10",
"VarName11", "VarName12", "VarName13", "VarName14", "VarName15", "VarName16",
"VarName17", "VarName18", "VarName19", "VarName20", "VarName21", "VarName22",
"VarName23", "VarName24", "VarName25", "VarName26", "VarName27", "VarName28",
"VarName29", "VarName30", "VarName31", "VarName32", "VarName33", "VarName34",
"VarName35", "VarName36", "VarName37", "VarName38", "VarName39", "VarName40",
"VarName41", "VarName42", "VarName43", "VarName44", "VarName45", "VarName46",
"VarName47", "VarName48", "VarName49", "VarName50", "VarName51", "VarName52",
"VarName53", "VarName54", "VarName55", "VarName56", "VarName57", "VarName58",
"VarName59", "VarName60", "VarName61", "VarName62", "VarName63", "VarName64",
"VarName65", "VarName66", "VarName67", "VarName68", "VarName69", "VarName70",
"VarName71", "VarName72", "VarName73", "VarName74", "VarName75", "VarName76",
"VarName77", "VarName78", "VarName79", "VarName80", "VarName81", "VarName82",
"VarName83", "VarName84", "VarName85", "VarName86", "VarName87", "VarName88",
"VarName89", "VarName90", "VarName91", "VarName92", "VarName93", "VarName94",
"VarName95", "VarName96", "VarName97", "VarName98", "VarName99", "VarName100",
"VarName101", "VarName102", "VarName103", "VarName104", "VarName105",
"VarName106", "VarName107", "VarName108", "VarName109", "VarName110",
"VarName111", "VarName112", "VarName113", "VarName114", "VarName115",
"VarName116", "VarName117", "VarName118", "VarName119", "VarName120",
"VarName121", "VarName122", "VarName123", "VarName124", "VarName125",
```

```
"VarName126", "VarName127", "VarName128", "VarName129", "VarName130",
"VarName131", "VarName132", "VarName133", "VarName134", "VarName135",
"VarName136", "VarName137", "VarName138", "VarName139", "VarName140",
"VarName141", "VarName142", "VarName143", "VarName144", "VarName145",
"VarName146", "VarName147", "VarName148", "VarName149", "VarName150",
"VarName151", "VarName152", "VarName153", "VarName154", "VarName155",
"VarName156", "VarName157", "VarName158", "VarName159", "VarName160",
"VarName161", "VarName162", "VarName163", "VarName164", "VarName165",
"VarName166", "VarName167"];
opts.VariableTypes = ["string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string", "string", "string", "string", "string", "string", "string", "string",
"string"];
opts = setvaropts(opts, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101,
102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118,
119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135,
136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152,
153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167],
"WhitespaceRule", "preserve");
opts = setvaropts(opts, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101,
102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118,
119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135,
136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152,
153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167],
"EmptyFieldRule", "auto");

% Import the data
matrizODCOLEGIOS1 =
readtable("C:\Users\mncrm\Desktop\MIKEL\US\TFM\matrizOD_COLEGIOS.xlsx", opts,
"UseExcel", false);
```

```
b=[];
clear opts
for i=1:167
    a=matrizODCOLEGIOS1(i,:);
    a=table2array(a)
    for j=1:167
        if i~=j
    a(j)= strrep(a(j),'[','');
        end
        j=j+1
    end

    b=[b;a];
    i=i+1
end
for i=1:167
    for j=1:167
        b(i,j)= strrep(b(i,j),"'","");
    end
end
```

```
opts = spreadsheetImportOptions("NumVariables", 167);

% Specify sheet and range
opts.Sheet = "Hoja1";
opts.DataRange = "A1:FK167";

% Specify column names and types
opts.VariableNames = ["NaN", "VarName2", "VarName3", "VarName4", "VarName5",
"VarName6", "VarName7", "VarName8", "VarName9", "VarName10", "VarName11",
"VarName12", "VarName13", "VarName14", "VarName15", "VarName16", "VarName17",
"VarName18", "VarName19", "VarName20", "VarName21", "VarName22", "VarName23",
"VarName24", "VarName25", "VarName26", "VarName27", "VarName28", "VarName29",
"VarName30", "VarName31", "VarName32", "VarName33", "VarName34", "VarName35",
"VarName36", "VarName37", "VarName38", "VarName39", "VarName40", "VarName41",
"VarName42", "VarName43", "VarName44", "VarName45", "VarName46", "VarName47",
"VarName48", "VarName49", "VarName50", "VarName51", "VarName52", "VarName53",
"VarName54", "VarName55", "VarName56", "VarName57", "VarName58", "VarName59",
"VarName60", "VarName61", "VarName62", "VarName63", "VarName64", "VarName65",
"VarName66", "VarName67", "VarName68", "VarName69", "VarName70", "VarName71",
"VarName72", "VarName73", "VarName74", "VarName75", "VarName76", "VarName77",
"VarName78", "VarName79", "VarName80", "VarName81", "VarName82", "VarName83",
"VarName84", "VarName85", "VarName86", "VarName87", "VarName88", "VarName89",
"VarName90", "VarName91", "VarName92", "VarName93", "VarName94", "VarName95",
"VarName96", "VarName97", "VarName98", "VarName99", "VarName100",
"VarName101", "VarName102", "VarName103", "VarName104", "VarName105",
"VarName106", "VarName107", "VarName108", "VarName109", "VarName110",
"VarName111", "VarName112", "VarName113", "VarName114", "VarName115",
"VarName116", "VarName117", "VarName118", "VarName119", "VarName120",
"VarName121", "VarName122", "VarName123", "VarName124", "VarName125",
"VarName126", "VarName127", "VarName128", "VarName129", "VarName130",
```

```
"VarName131", "VarName132", "VarName133", "VarName134", "VarName135",
"VarName136", "VarName137", "VarName138", "VarName139", "VarName140",
"VarName141", "VarName142", "VarName143", "VarName144", "VarName145",
"VarName146", "VarName147", "VarName148", "VarName149", "VarName150",
"VarName151", "VarName152", "VarName153", "VarName154", "VarName155",
"VarName156", "VarName157", "VarName158", "VarName159", "VarName160",
"VarName161", "VarName162", "VarName163", "VarName164", "VarName165",
"VarName166", "VarName167"];
opts.VariableTypes = ["double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double"];

% Import the data
matrizODCOLEGIOS =
readtable("C:\Users\mncrm\Desktop\MIKEL\US\TFM\matrizOD_COLEGIOS.xlsx", opts,
"UseExcel", false);
matrizODCOLEGIOS = table2array(matrizODCOLEGIOS);

ab=matrizODCOLEGIOS;

clear  opts
```

ANEXO 9. LATITUD-LONGITUD

```
%%A partir de la información del excel con nombre, calle y código postal,
%%se obtuvo latitud y longitud de cada centro educativo
clear

clc

 format long
opts = spreadsheetImportOptions("NumVariables", 13);

% Specify sheet and range
opts.Sheet = "colegios_sevilla";
opts.DataRange = "A2:M168";

% Specify column names and types
opts.VariableNames = ["Cdigo", "Denominacin", "Nombre", "Dependencia",
"Domicilio", "Localidad", "Municipio", "Provincia", "CdPostal", "Telfono",
"Enseanzas", "Servicios", "Programas"];
opts.VariableTypes = ["double", "categorical", "string", "categorical",
"string", "categorical", "categorical", "categorical", "double", "double",
"categorical", "categorical", "categorical"];
opts = setvaropts(opts, [3, 5], "WhitespaceRule", "preserve");
opts = setvaropts(opts, [2, 3, 4, 5, 6, 7, 8, 11, 12, 13], "EmptyFieldRule",
"auto");

% Import the data
listadoCOLEGIOS =
readtable("C:\Users\mncrm\Desktop\MIKEL\US\TFM\listado_COLEGIOS.xlsx", opts,
"UseExcel", false);
COLEGIOS= listadoCOLEGIOS(:,2);

COLEGIOS = table2array(COLEGIOS);



NOMBRES= listadoCOLEGIOS(:,3);

NOMBRES = table2array(NOMBRES);

  %%PARA CADA CALLE HAGO LA DIRECCIÓN DE LA PETICIÓN A GOOGLE
 for i=1:167
%
     COLEGIOS(i)= strrep(COLEGIOS(i),' ','+');
%      CALLE_COLEGIOS(i,)= strrep(CALLE_COLEGIOS(i),'C/','Calle');
     COLEGIOS(i)= strrep(COLEGIOS(i),'ñ','n');
     NOMBRES(i)= strrep(NOMBRES(i),' ','+');
      NOMBRES(i)= strrep(NOMBRES(i),'ñ','n');
    i=i+1;
 end
```

```
 DIRECCION_LAT=strings([167,1]);
 DIRECCION_LON=strings([167,1]);
%
 for i=1:167
     %%QUITO LAS COMILLAS
     A=categorical(COLEGIOS(i));
     B=categorical(NOMBRES(i));


     A=cellstr(A);
     B=cellstr(B);
%     % PET={'https://maps.googleapis.com/maps/api/geocode/json?address='}
%     esta era una instrucción para la API
%
    PET={'https://maps.google.com/?q='};
     %%introduzco sevilla
     sev={'Sevilla#'};
     PET_GOOGLE= strcat(PET,A,B,sev);
     PET_GOOGLE= char(PET_GOOGLE);
     DIRECCION=urlread(PET_GOOGLE);


%  INSTRUCCIONES PARA LA API
%     %PARA EVITAR QUE SE QUEDE EN EL LIMITE DE PETICIONES
% %
% %     k = strfind(DIRECCION,'OVER_QUERY_LIMIT');
% %     while k>0
% %         DIRECCION=urlread(PET_GOOGLE)
% %         k = strfind(DIRECCION,'OVER_QUERY_LIMIT')
% %     end
%      [C,matches] = strsplit(DIRECCION,{'"location"
','"location_type"'},'CollapseDelimiters',true);%%CON ESTE COMANDO SEPARO LA
CADENA EN TRES ME INTERESA LA DE EN MEDIO LAT LON


[C,matches] =
strsplit(DIRECCION,{'window.APP_INITIALIZATION_STATE='},'CollapseDelimiters',tru
e);
    C=C(2);
    C=char(C);
    [C,matches] = strsplit(C,{']'},'CollapseDelimiters',true);
      C=C(1);
       C=char(C);
      [C,matches] = strsplit(C,{','},'CollapseDelimiters',true);


        DIRECCION_LAT(i,1)=C(3);%%latitud


        DIRECCION_LON(i,1)=C(2);%%longitud


 end
%CAMBIOS MANUALES FACILES DE ENCONTRAR PQ SALE MI LAT Y LONG
```

```
 DIRECCION_LON(12)="-5.9704283"
 DIRECCION_LAT(12)="37.3966988"

 DIRECCION_LON(31)="-5.9870951"
 DIRECCION_LAT(31)="37.4230164"

 DIRECCION_LON(38)="-6.0045139"
 DIRECCION_LAT(38)="37.3827182"

 DIRECCION_LON(42)="-6.0145721"
 DIRECCION_LAT(42)="37.3741403"

 DIRECCION_LON(54)="-5.9632543"
 DIRECCION_LAT(54)="37.3608215"

 DIRECCION_LON(62)="-5.9545899"
 DIRECCION_LAT(62)="37.3687977"

 DIRECCION_LON(63)="-5.9675832"
 DIRECCION_LAT(63)="37.3745059"

 DIRECCION_LON(129)="-5.96907"
 DIRECCION_LAT(129)="37.4192452"

 DIRECCION_LON(132)="-5.9804657"
 DIRECCION_LAT(132)="37.3556112"

 DIRECCION_LON(144)="-5.9699571"
 DIRECCION_LAT(144)="37.3762112"

 DIRECCION_LON(152)="-6.0021141"
 DIRECCION_LAT(152)="37.3753019"

 listadoCOLEGIOS.Var14(:)=DIRECCION_LAT; %% UNO LAS LATITUDES A TODA LA
INFORMACIÓN DE LAS FARMACIAS
listadoCOLEGIOS.Var15(:)=DIRECCION_LON; %% UNO LAS LONGITUDES A TODA LA
INFORMACIÓN DE LAS FARMACIAS
listadoCOLEGIOS.Properties.VariableNames{14} = 'LATITUD';
listadoCOLEGIOS.Properties.VariableNames{15} = 'LONGITUD';
writetable(listadoCOLEGIOS,'listadoCOLEGIOS.xlsx');
```