UNIVERSITAS
Miguel Hernández

El problema del viajante de comercio con rutas suficientemente próximas.

UNIVERSIDAD MIGUEL HERNÁNDEZ

FACULTAD DE CIENCIAS SOCIALES Y JURÍDICAS DE ELCHE

GRADO EN ESTADÍSTICA EMPRESARIAL

TRABAJO DE FIN DE GRADO

Autor: Juan Antonio Ramos Mora

Tutor: Mercedes Landete Ruiz

CURSO 2019-2020

Índice

1 Introducción

La optimización combinatoria es uno de los campos de la optimización en el mundo de las matemáticas. Esta técnica, se basa en la evaluación de un conjunto de soluciones finito para la toma de decisiones. En este ámbito, uno de los problemas más estudiados a lo largo de la historia, es el problema del viajante de comercio, el cual trata de realizar una ruta que pase por toda una serie de puntos minimizando la distancia.

En el presente proyecto, abordaremos este problema, así como algunas de sus adaptaciones y aplicaciones más relevantes. Por ejemplo, una de las aplicaciones de este problema, sería google maps. Esta herramienta nos permite hallar los caminos más eficientes entre dos puntos. Además de las restricciones físicas de carreteras y edificios, podríamos configurar la ruta evitando peajes o incluso eligiendo entre si vamos andando, en coche o en bicicleta. Todo esto forma parte de un conjunto de restricciones que hay que reflejar en el problema. Más adelante, exponemos algunos ejemplos.

Una de las adaptaciones más novedosas, es el problema del viajante de comercio con rutas suficientemente próximas, donde ya no se ha de visitar el punto en sí, sino que se puede pasar lo suficientemente cerca de él. Es una de las adaptaciones más recientes y que tiene múltiples aplicaciones en la vida real. Por ejemplo, existen empresas de logística punteras en el sector que están empezando a implantar flotas de drones para sus servicios de entrega, en las que basan este problema, adaptándolo a su casuística concreta. También tiene uso en aplicaciones de ámbito militar.

Por otro lado, también veremos otro método alternativo mediante el cual calcular la distancia de la ruta, pero sin calcular la misma. Este método recibe el nombre de métodos de aproximación continua. A lo largo de la historia, múltiples autores han realizado aportaciones a este campo, con numerosos estudios donde ofrecían diferentes adaptaciones de la fórmula. Este campo, tiene múltiples aplicaciones en la vida real, como por ejemplo, la creación de distritos o la evaluación previa de un estudio mediante este método, como por ejemplo, la creación de una nueva linea de autobús, dados x número de paradas. Con esta fórmula, hallamos una aproximación del posible coste de la ruta y a partir de aquí, podríamos implantar métodos más precisos. Uno de nuestros objetivos va a ser calcular una fórmula que estime la distancia de estas rutas, en especial, para el caso de rutas suficientemente próximas.

A lo largo del grado en Estadística Empresarial, se han impartido diferentes asignaturas de optimización en las que se han asentado las bases de esta disciplina. En concreto, la asignatura de tercero, "Modelos de Optimización", trató el problema del viajante de comercio junto con otros renombrados problemas. Esta asignatura sirvió de base para el planteamiento de este proyecto. En cursos posteriores, se impartió optimización con programación cuadrática, básica para algunas adaptaciones de este problema. En concreto, en la adaptación a rutas suficientemente cercanas, se usa la programación cuadrática en la formulación del problema en múltiples ocasiones.

Otra de las disciplinas que se han impartido durante el grado y han servido de mucha ayuda en el trabajo, ha sido la regresión estadística. Son múltiples las asignaturas que han impartido este

método, ya que es una herramienta básica del estadístico. A lo largo del grado, esta materia, se ha estudiado en diversos lenguajes de programación o interfaces como R, SPSS, o Excel. Asignaturas como "Minería de datos", "Análisis de mercados", "Series temporales" o "Mejora de procesos", trataron este tema ampliamente. En el caso que nos ocupa, nos ha sido de gran utilidad para dar una estimación de un modelo de aproximación continua para el problema de viajante con rutas suficientemente próximas, evaluando qué aspectos o parámetros tienen influencia en esta estimación.

En este caso, la totalidad del proyecto se ha realizado bajo las mismas condiciones informáticamente hablando. Se ha empleado un terminal con Windows 10 de 64 bits, un procesador Intel Core i5.8250U de 1.80GHz y 4 Núcleos sumado a una RAM de 8 GB y un almacenamiento SSD de 256 GB. Concretamente, usaremos la interfaz de R, RStudio en su versión 1.2.5033.

2 Problemas actuales relacionados con rutas

2.1 Problema del viajante de comercio (en inglés, Travelling Salesman Problem o TSP)

El TSP (Travelling Salesman Problem) es uno de los conocidos problemas de optimización más estudiados. El problema trata de minimizar una distancia, pasando por una serie de puntos. A estos puntos, los llamamos nodos y a la línea que los une, arcos. Cada arco tendrá una distancia diferente. La distancia total es la resultante del recorrido de la totalidad de nodos. Como todo problema de optimización, vamos a tener una serie de restricciones.

A lo largo de la historia, se han definido varios modelos para este problema. Principalmente, se dividen en formulación compacta y formulación no compacta. Formulación compacta implica que se pueden escribir todo el conjunto de restricciones. En cambio, la no compacta el número de restricciones pasa de ser polinomial a ser exponencial. A la hora de escribir el modelo en un solver, en este último caso, se precisa cierta destreza computacional. Para incorporar las restricciones a medida que van siendo necesarias, el solver, va a ir creando una serie de conjuntos y subconjuntos los cuales, imposibles de definir en una sola restricción de forma explícita.

Modelo con formulación compacta:

Definimos variables y parámetros necesarios:

Parámetros:

C_{ij} : Distancia que une el nodo i al nodo j. $\forall$ i,j=1,...,n.

Variables:

U_i : Posición del nodo en la ruta $\forall$ i=1,...,n.

$$X_{ij} \begin{cases} 1 & \text{Si el nodo j precede al nodo i.} \quad \forall i,j = 1,...,n. \\ 0 & \text{En caso contrario.} \end{cases}$$

Formulación matemática:

$$Minimizar \qquad z = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}X_{ij} \tag{1}$$

$s.a:$

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad i = 1, ..., N \tag{2}$$

$$\sum_{i=1}^{n} X_{ij} = 1 \qquad j = 1, ..., N \tag{3}$$

$$U_i - U_j + nX_{ij} \leq n - 1 \qquad i, j = 2, ..., n \quad i \neq j \tag{4}$$

$$X_{ij} \in 0, 1 \qquad \forall i, j \tag{5}$$

$$C_{ii} = \infty \qquad i = 1, 2, ..., n. \tag{6}$$

La función objetivo (1), trata de minimizar la distancia total. Por ejemplo, si trazamos dos rutas diferentes, tendremos un conjunto de arcos con distancias C_{ij}, y la variable X_{ij} reflejará la ruta a seguir. El modelo nos dirá qué ruta es la óptima, es decir, la de menor distancia. Este proceso se formulará para el conjunto de rutas posibles.

Veamos a continuación las restricciones. Tenemos que la restricción (2) obliga al modelo a que en cada nodo entre un arco exactamente. Por otro lado, la restricción (3) hace que de cada nodo salga un arco exactamente.
De esta forma, lo que estamos haciendo es darles entrada y salida a todos los nodos y que no se quede ningún nodo sin visitar. En caso contrario, podría darse el siguiente caso (Figure 1):

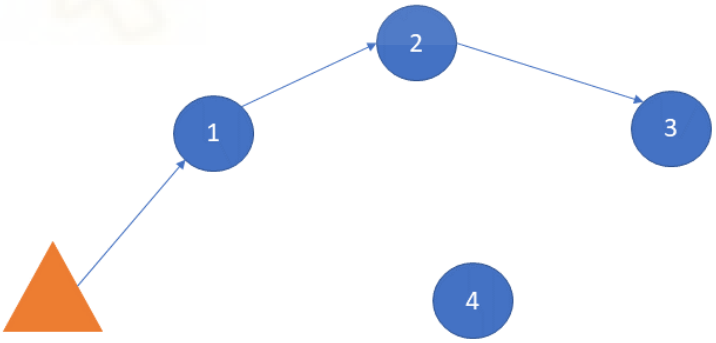

Figure 1: Restricciones 1 ó 2 inactivas. Fuente: elaboración propia.

Como podemos ver, el nodo 4, no tendría ningún nodo precedente. Las restricciones 2 y 3 impedirían este hecho, para todo i,j=1,...,N. De esta forma, todos los nodos serían visitados.

La restricción (4) implica que la ruta no puede tener ciclos. Y ya que todos los nodos han de ser visitados, se crea un flujo continuo en el que cada arco necesariamente tiene que estar unido a un nodo.

Como vemos claramente en la Figura 2, el flujo se corta entre el nodo 5 y 6. La restricción (4) fuerza a que ese corte no se produzca, siendo todo una única ruta, como vemos en la Figura 3.

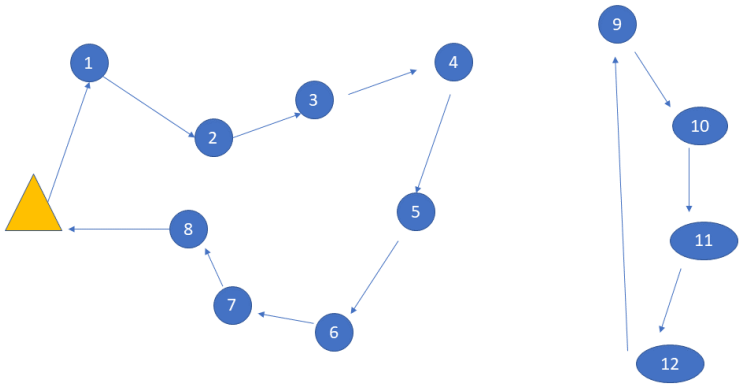

Figure 2: Restricción (4) inactiva. Fuente: elaboración propia.

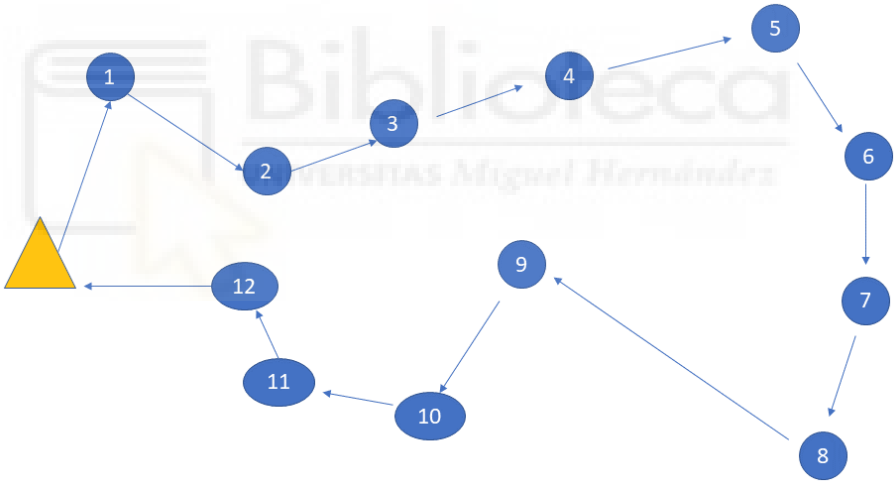

Figure 3: Restricción (4) activa. Fuente: elaboración propia.

Por último, en la restricción (5), vemos que se declara la variable X_{ij} como binaria. Anteriormente, establecimos los posibles valores junto con su significado.

Además, tenemos que C_{ii} es infinito para evitar que lo más barato sea no salir del nodo.

Veamos un ejemplo con números para dejar claros los conceptos arriba definidos:

Recordemos que la variable X_{ij} nos decía si viajábamos del nodo i al nodo j.

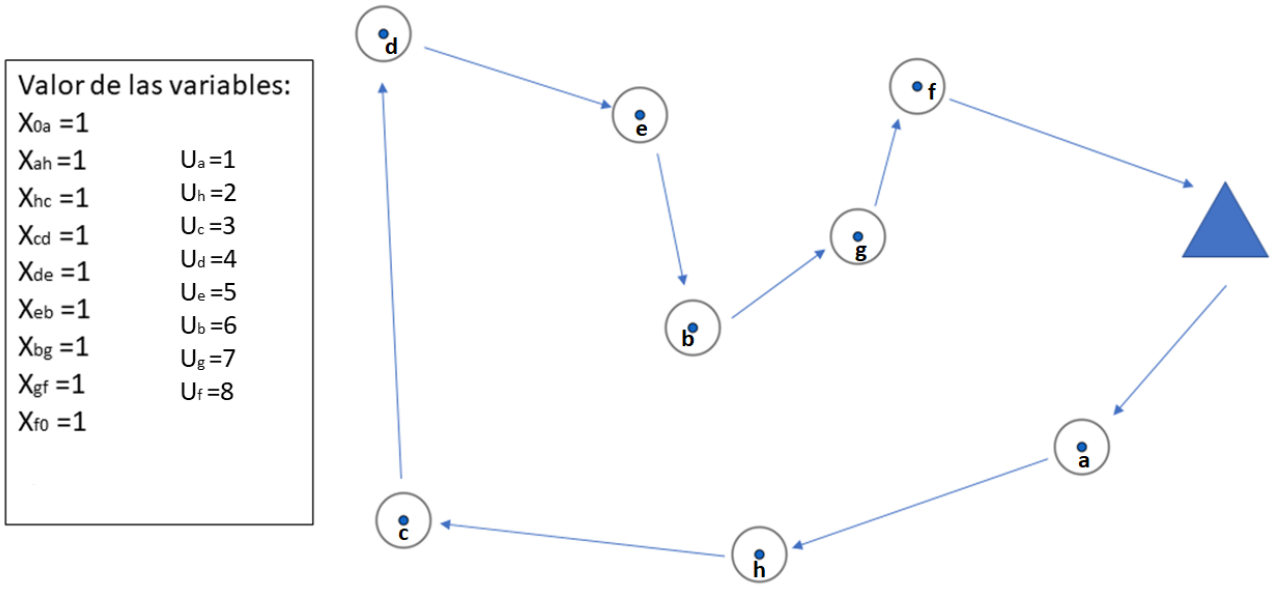

Figure 4: Ejemplo de un TSP. Fuente: elaboración propia.

Suponemos que la ruta es la óptima minimizando la mayor distancia. Por tanto, si la variable X_{ah} vale 1, significa que viajaremos del nodo a al h. En dicho caso, la distancia vendrá definida por la suma de los C_{ij} multiplicado por las X_{ij}. Es decir:

$$C_{0a} + C_{ah} + C_{hc} + C_{cd} + C_{de} + C_{eb} + C_{bg} + C_{gf} + C_{f0}$$

Porque el resto de $C_{ij} * X_{ij}$ es 0.

Modelo con formulación no compacta

Definimos variables y parámetros necesarios:

Parámetros

C_{ij} : Distancia que une el nodo i al nodo j. $\forall$ i,j=1,...,n.

Variables

$$X_{ij} \begin{cases} 1 & \text{Si el nodo j precede al nodo i.} \quad \forall i,j = 1,...,n. \\ 0 & \text{En caso contrario.} \end{cases}$$

El objetivo en este caso es el mismo que el anterior, minimizar la distancia total de la ruta C_{ij}.

Por otro lado, las restricciones son similares, solo cambia el modo en el que formulamos la restricción (10), pues es la que determina que la formulación es no compacta. Veamos el modelo formulado:

$$Minimizar \qquad z = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}X_{ij} \tag{7}$$

$s.a:$

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad i = 1, ..., N \tag{8}$$

$$\sum_{i=1}^{n} X_{ij} = 1 \qquad j = 1, ..., N \tag{9}$$

$$\sum_{(i,j)\in A:\, i\in U,\, j\in(V\setminus U)} X_{ij} \geq 1 \qquad 2 \leq |U| \leq |V| - 2 \tag{10}$$

En un primer vistazo, vemos que el modelo es muy similar al método compacto. Bien, en principio es así, pero con ciertas salvedades. La más importante, es el modo en el que nos aseguramos de que creamos una ruta única, sin ciclos. Esto es, la restricción (10). En este caso, este modelo formula que dados dos nodos i y j, pertenecientes a dos conjuntos diferentes, U y V, existe al menos una ruta que une ambos puntos. Estos dos conjuntos pueden tener puntos que coincidan en la intersección, pero en este caso, estos dos puntos no se encuentran en dicha intersección.

Como antes hemos mencionado, este método es exponencial, pues el número de conjuntos y subconjuntos que podemos formar con los diferentes nodos puede aumentar de forma exponencial, variando el número de nodos y sus diferentes combinaciones.

Por ejemplo, tomamos 100 nodos, (n=100), en este caso, el número de combinaciones diferentes es enorme. Por no hablar del número de combinaciones de las posibles rutas y sus distancias pertinentes. Este modelo, es más complejo debido al número de restricciones que lo compondrían por lo anteriormente mencionado.

Hemos visto dos métodos de formular el mismo problema (TSP). Ambos le dan solución a este básico problema, aunque con distintas formas internamente. Adicionalmente, a lo largo de la historia, el problema se ha modificado añadiendo o eliminando restricciones en función de las necesidades dadas. Así como distintas variaciones y adaptaciones. Expongamos algunos ejemplos de las distintas modulaciones:

El TSP máximo.
En este caso, como claramente se deduce por el título, el objetivo es maximizar la distancia de la ruta. Vulgarmente, esta adaptación se conoce como *"El problema de la estafa del taxi"*. Centrándonos en la formulación matemática, cambiaríamos minimizar la distancia por maximizar. En el caso de usar un solver para resolver este caso, usaríamos el mismo que para un TSP normal, pero cambiando los coeficientes de la matriz de distancias.
Veamos un ejemplo:

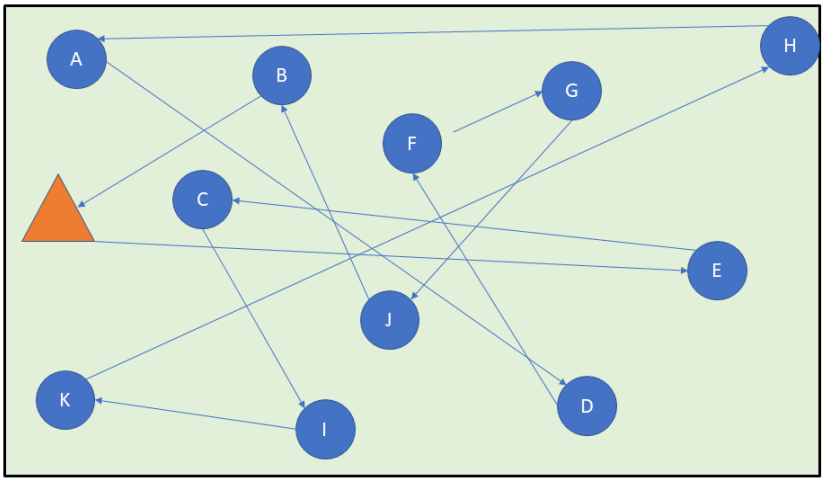

Figure 5: TSP máximo. Fuente: elaboración propia.

En el ejemplo, suponemos que queremos realizar una carrera de fondo y el terreno por el cual podemos realizar el recorrido, es la zona sombreada en la Figura 5. Bien, pues dentro de esa zona, intentaremos maximizar la distancia, pues tenemos ciertas restricciones de territorio por el que no nos está permitido pasar.

TSP con múltiples visitas (TSPM).
En este caso, se pretende encontrar una ruta la cual parte desde el nodo inicial hacia cualquier nodo, visitándolo al menos una vez. Una vez visitados todos vuelve al origen. El objetivo es minimizar la distancia. Entenderemos mejor este caso gráficamente:

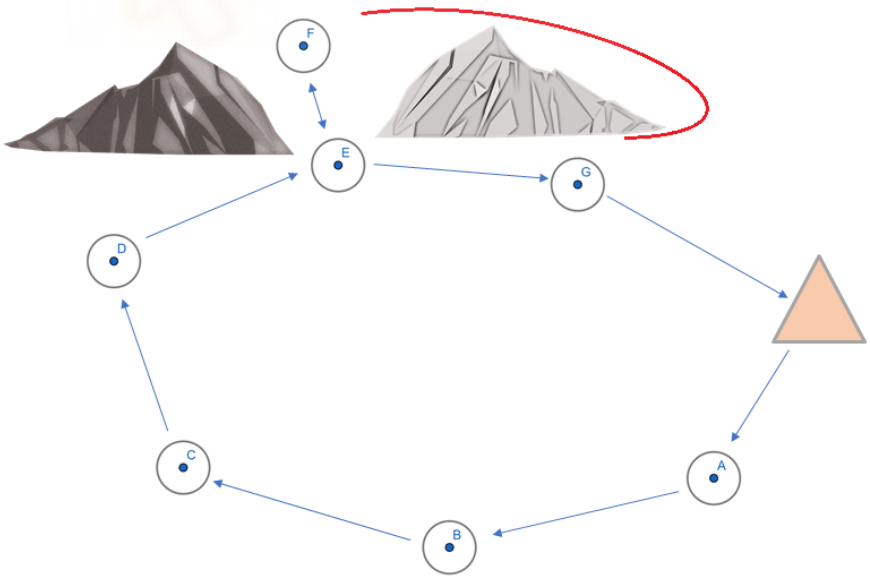

Figure 6: TSP con múltiples visitas. Fuente: elaboración propia.

Como podemos ver, la única forma de ir del nodo F al nodo G, es volviendo al nodo E, pues la

distancia rodeando el obstáculo que puede ser una cadena montañosa sin camino que la cruce, va a ser mucho mayor que volver a E e ir a G. El caso sería el mismo si nos planteamos ir del nodo D al F. A la hora de querer solventar este modelo mediante un solver, no nos va a ser posible, sino que tendremos que modelar a mano el modelo en un software. Por ejemplo, las restricciones (2) y (3) serían de $\geq$ en lugar de $=$.

The pick-up-delivery problem (El problema de recogida y entrega).
Esta adaptación, se basa en que hay una serie de nodos en los que se entrega mercancía y otros en los que se recoge. Como vemos en la Figura (7), los nodos 1 y 3 son de abastecimiento, mientras que los puntos 2,4,5 son de entrega. En el caso de querer usar un software, decir que no tenemos un solver que nos lo haga, sino que tendríamos que modelarlo.

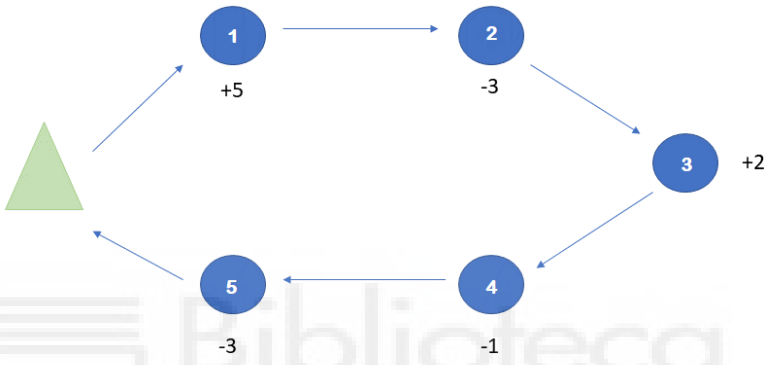

Figure 7: Pick-up-delivery problem. Fuente: elaboración propia.

TSP con clústers.
En este particular caso, el conjunto de nodos (G) se divide en clústeres $V_1, V_2, ..., V_k$. El objetivo es encontrar el recorrido de menor coste o distancia, sujeto a que las ciudades que estén dentro del mismo clúster se han de visitar consecutivamente. Como vemos en la Figura 8 Se ha de minimizar la distancia dentro del clúster y luego, fuera de él, uniendo el siguiente clúster. Esta última adaptación tampoco se encuentra recogida en un solver, por lo que deberíamos modelar también.

TSP con múltiples vehículos (VRP).
Todas estas adaptaciones, tienen un elemento en común, una ruta que recorrer. Dado el caso, se puede contemplar la existencia de varios vehículos. Esto es conocido como el *Vehicle Routing Problem (VRP)*. Todas las adaptaciones tendrían cabida bajo este enfoque, pero ya no recibirían el nombre de TSP sino VRP. Un ejemplo claro, sería la Figura 9

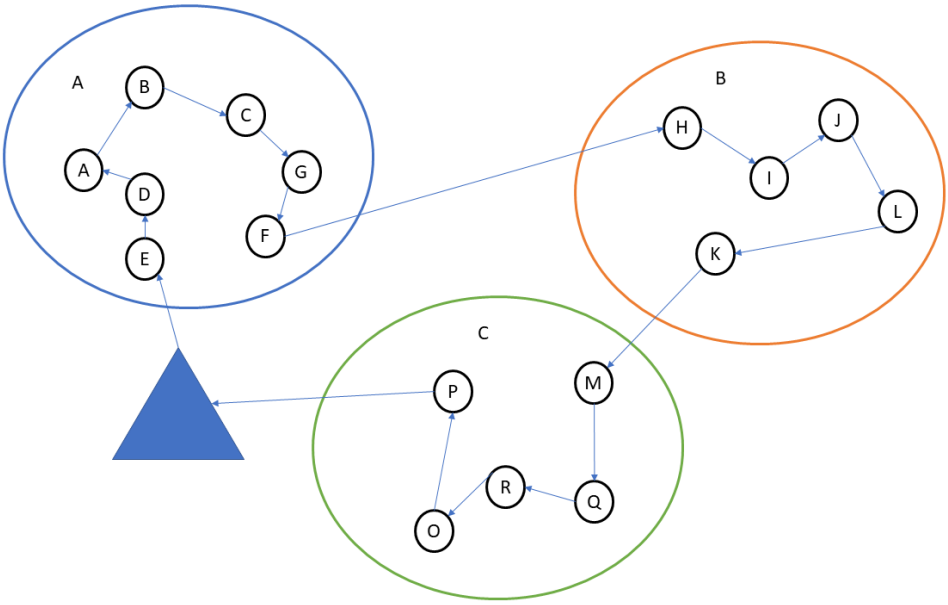

Figure 8: TSP con clústers. Fuente: elaboración propia.

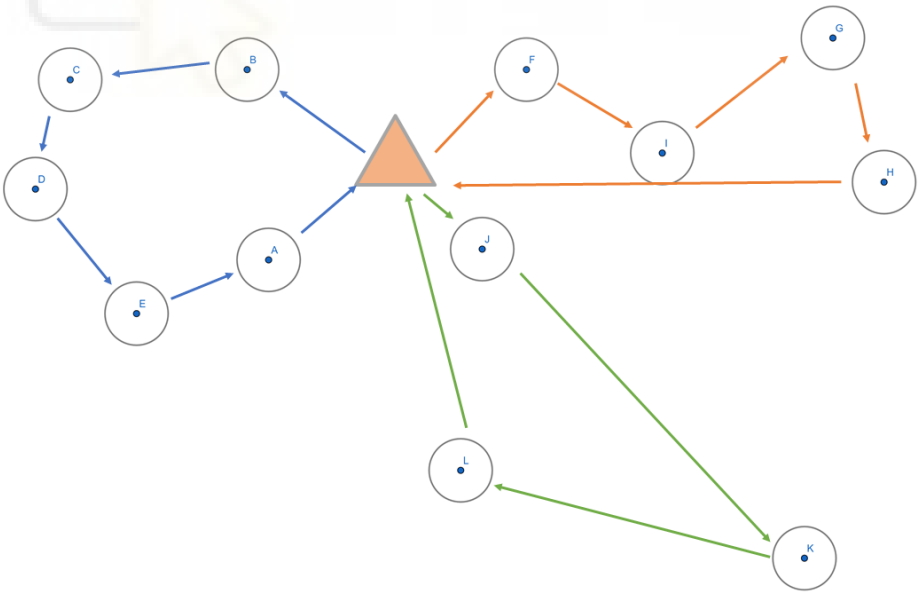

Figure 9: Vehicle Routing Problem. Fuente: elaboración propia.

2.2 El problema del viajante de comercio con rutas suficientemente próximas. (En inglés, Close Enough TSP).

Close Enough TSP es una variante del TSP. La característica de este problema es que ya no se precisa pasar por el nodo en sí, sino que se establece un radio alrededor de cada nodo de tal forma que la ruta ha de pasar lo suficientemente cerca del nodo a visitar.
Como consecuencia, la distancia puede ser menor, pues si coinciden los radios de varios nodos, en lugar de pasar por cada nodo, solo se ha de pasar por la intersección de esos nodos. Lo veremos más claro en la Figura 10:

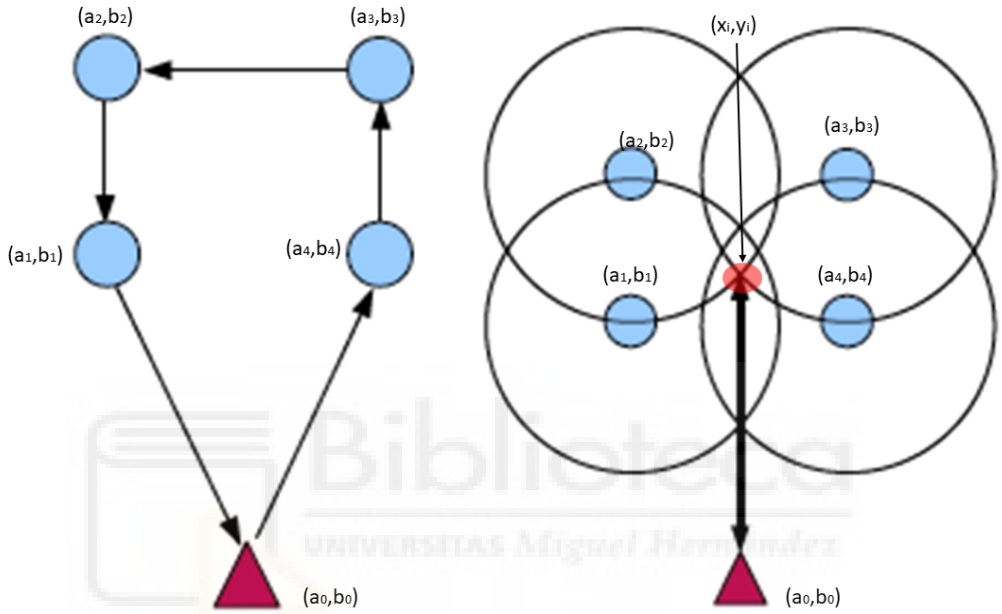

Figure 10: Diferencia TSP - Close Snough TSP. Fuente: elaboración propia.

Como vemos claramente en la la Figura 10, en la parte de la izquierda tenemos un TSP clásico. A la derecha, tenemos lo que sería graficamente un close enough TSP. En este caso, cualquier ruta que toque o pase por el radio del nodo, visita satisfactoriamente el nodo i, (con la excepción del nodo inicial, del cual partimos y no tendría radio, a priori).

Las variables y parámetros de este caso, van a ser las siguientes:

Parámetros:

(a_i, b_i) : Coordenadas de cada nodo $\forall$ i,j=1,...,n.

r :Distancia que cada nodo (cliente) está dispuesto a desplazarse.

Variables:

$$e_{ij}\begin{cases}1 & \text{Si el nodo j precede inmediatamente al nodo i.} \qquad \forall i,j=1,...,n.\\ 0 & \text{En caso contrario.}\end{cases}$$

(x_i, y_i) : Coordenadas del punto que minimiza la distancia $\quad \forall$ i,j=1,...,n.

$$Minimizar \qquad \sum_{i=0}^{n}\sum_{j=0}^{n} e_{ij}\sqrt{(x_i-x_j)^2+(y_i-y_j)^2} \tag{11}$$

$s.a$:

$$(x_i-a_i)^2+(y_i-b_i)^2 \leq r^2 \qquad i=1,...,n \tag{12}$$

$$\sum_{i=0}^{n} e_{ij}=1 \qquad j=0,...,n \tag{13}$$

$$\sum_{j=0}^{n} e_{ij}=1 \qquad i=0,...,n \tag{14}$$

$$\sum_{(i,j)\in A:\, i\in U,\, j\in(V-U)} e_{ij} \geq 1 \qquad 2\leq|U|\leq|V|-2 \tag{15}$$

$$\sqrt{(x_i-x_j)^2+(y_i-y_j)^2} \geq \epsilon \qquad i,j=0,...,n \tag{16}$$

$$e_{ij}\in[0,1], \quad (x_i,y_i)\in RxR \qquad i,j=0,...,n \tag{17}$$

La función objetivo (11) trata de minimizar la distancia total tras recorrer todos puntos (x_i, y_i). Recordemos que este punto es una variable y (a_i, b_i) es la coordenada real del punto. Por lo que para cada (a_i, b_i), vamos a tener un (x_i, y_i).

En la restricción (12) vemos como a cada variable le resta su correspondiente coordenada y eleva al cuadrado, ya que para cada (a_i, b_i) tenemos un punto (x_i, y_i). Esto ha de ser al menos menor que el radio al cuadrado. Haciendo esto, nos aseguramos que cada nodo tenga un disco de radio r por el que poder pasar para ser visitado.

Tanto la restricción (13) como la (14), al igual que en el TSP clásico, fuerzan a que se les de salida y entrada a los nodos.

La restricción (15) evita que se cree más de una ruta. Es decir, dados dos nodos i y j, pertenecientes a dos conjuntos diferentes, U y V, existe al menos una ruta que une ambos puntos (estos dos puntos no se encuentran en la intersección de los conjuntos). La restricción (15) es la análoga a (10). Se podría escribir una formulación compacta imitando (4).

Aunque la restricción (16) no sea estrictamente necesaria, hemos de ponerla para forzar al software a que la solución sea única. Pongamos un ejemplo: en la Figura 10, de no poner esta restricción, la distancia sería 0. De esta forma, podríamos tener múltiples opciones en las que

la distancia sea 0 y el solver tardaría demasiado en resolverlo. En realidad, lo que estamos haciendo es ayudar al software a tener menos soluciones alternativas.

Por último, la restricción (17) sería la declaración de que la variable y que la variable e_{ij} es binaria y que x_i, y_i están en el plano R x R.

De igual forma que hemos visto anteriormente con el TSP, en el presente problema, podemos incorporar restricciones y modificarlo en función de las necesidades dadas. Por ejemplo, se puede dar el caso de que cada cliente tenga unas preferencias de desplazamiento. En este caso, el disco que rodea cada nodo será diferente.
Es decir, la restricción ahora sería $(x_i - a_i)^2 + (y_i - b_i)^2 \leq r_i^2$ para todo i = 1,...,n.

A raíz de este aspecto, existe la posibilidad de que tengamos la superposición de discos de radio r. A este hecho, se le llama zona de Steiner. En dicho caso, como bien ilustramos en la Figura 11, se visitarían varios nodos de una sola vez.

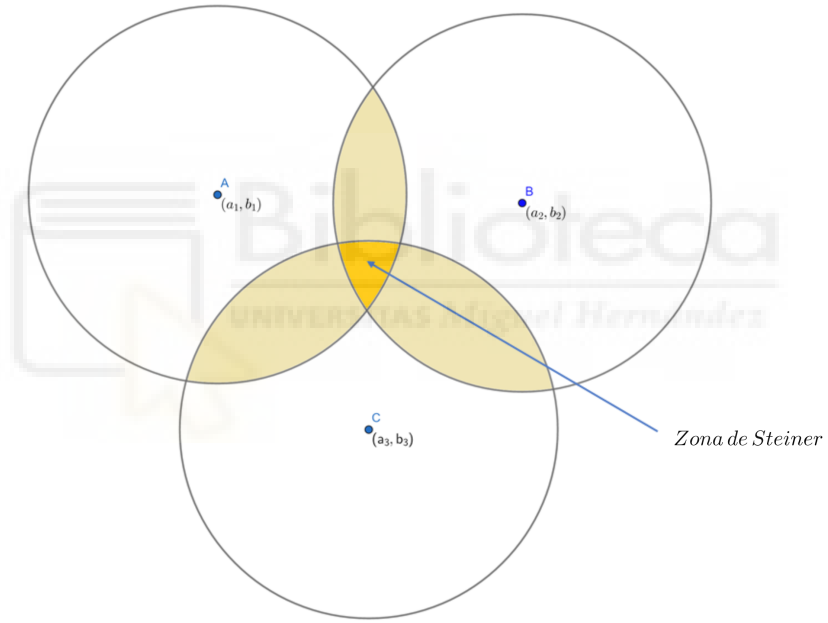

Figure 11: Zona de Steiner. Fuente: elaboración propia.

Como vemos, la zona de Steiner puede tener diferentes grados en función del número que intersecciones. Como podemos ver en la Figura 11, tenemos zonas de grado 1, donde solo interviene un nodo. Zonas de grado 2, donde hay una intersección entre dos nodos. Y por último, zonas de grado 3, donde interseccionan 3. Pueden haber tantos grados como nodos intersecciones entre sí.

El Close Enough TSP tiene diferentes aplicaciones en la vida real. Una de ellas, se puede ver en el ámbito de operaciones de vehículos aéreos no tripulados (UAV) como reconocimiento aéreo. Por ejemplo, consideremos que necesitamos hacer un reconocimiento geográfico. Para

ello, este dron ha de pasar lo suficientemente cerca del centro de cada región para una correcta recopilación de información.

Otra aplicación, podría ser la medición en contadores que emiten una señal wifi. El encargado de realizar la medición, debe establecer una ruta por la cual pase lo suficientemente cerca del contador para realizar la medición.

Un ejemplo diferente sería aquel en que una empresa logística tiene que realizar una serie de entregas y los clientes están dispuestos a desplazarse cierta distancia para que se produzca la entrega. En los últimos años, estamos viendo como este método se esta empleando más frecuentemente. Por ejemplo, hay ciertas empresas que han implantado un sistema de puntos de recogida de sus envíos. Cuando la mercancía se halla en el punto, el cliente se desplaza hacia él para recogerla.

Al igual que en el TSP, en este caso también tenemos adaptaciones del problema. Veamos algunos ejemplos:

TSP Close Enough con múltiples visitas (TSPM).
Al igual que en un TSP clásico, en un close enough también podemos implantar un modelo con múltiples visitas. La diferencia radica en que podremos pasar lo suficientemente cerca del nodo. Por lo cual, la distancia se verá minimizada en gran medida. Veamos la Figura 12 para una mejor comprensión:

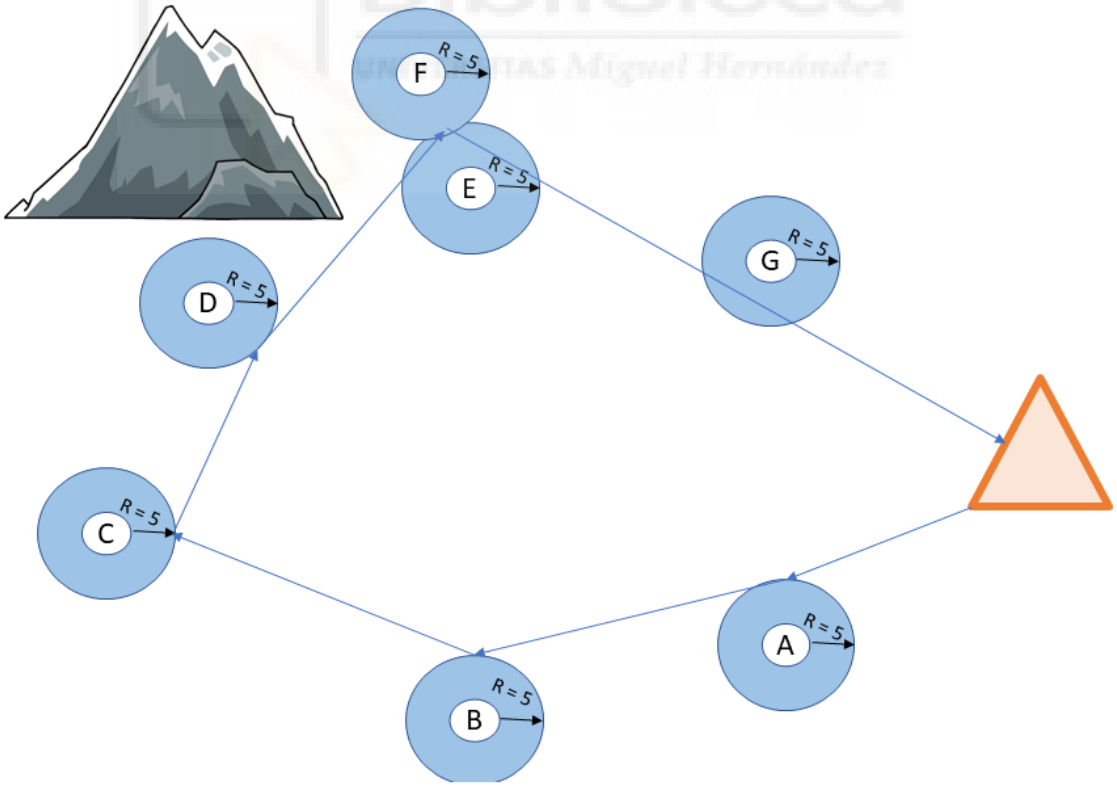

Figure 12: TSP con múltiples visitas. Fuente: elaboración propia.

A la hora de resolver estas variaciones, así como en cualquier TSP Close Enough, la forma va a ser modelando a mano, ya que no tenemos ningún solver el cual facilitándole los datos nos retorne una solución.

The close-enough pick-up-delivery problem (El problema de recogida y entrega).
Esta adaptación, se basa en que hay una serie de nodos en los que se entrega mercancía y otros en los que se recoge. Como vemos en la Figura (13), los nodos 1 y 3 son de abastecimiento, mientras que los puntos 2,4,5 son de entrega. Pasaremos lo suficientemente cerca de cada nodo para realizar la operación correspondiente.

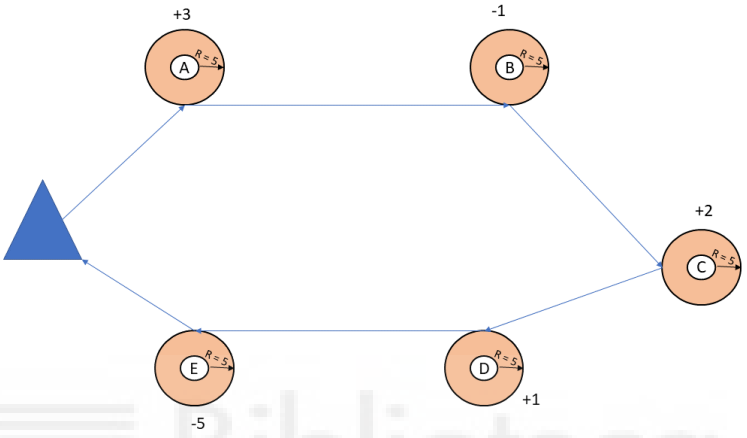

Figure 13: Pick-up-delivery problem. Fuente: elaboración propia.

TSP Close Enough con clústers.
En este particular caso, el conjunto de nodos (G) se divide en clústeres $V_1, V_2, ..., V_k$. El objetivo es encontrar el recorrido de menor coste ó distancia, sujeto a que las ciudades que estén dentro del mismo clúster se han de visitar consecutivamente lo suficientemente cerca de cada ciudad. Como vemos en la 8 Se ha de minimizar la distancia dentro del clúster y luego, fuera de él, uniendo el siguiente clúster. Quizás en este ejemplo, tenga todavía más sentido un close enough, ya que dentro del clúster, cubriríamos las visitas con una distancia mínima

Close-Enough Vehicle Routing Problem.
Esta adaptación se caracteriza por la existencia de múltiples vehículos. Cada nodo demanda una cantidad y debe ser servida solo por un vehículo. Cada vehículo está sujeto a una capacidad. Debe pasar a una distancia específica del nodo y todos los nodos han de ser visitados. Este caso tiene múltiples aplicaciones en las operaciones de vehículos aereos no tripulados, mencionado anteriormente.
Como vemos en la Figura 15, la ruta pasa lo suficientemente cerca de cada nodo (dentro de su radio). En este caso, tendríamos 3 vehículos con sus rutas correspondientes.

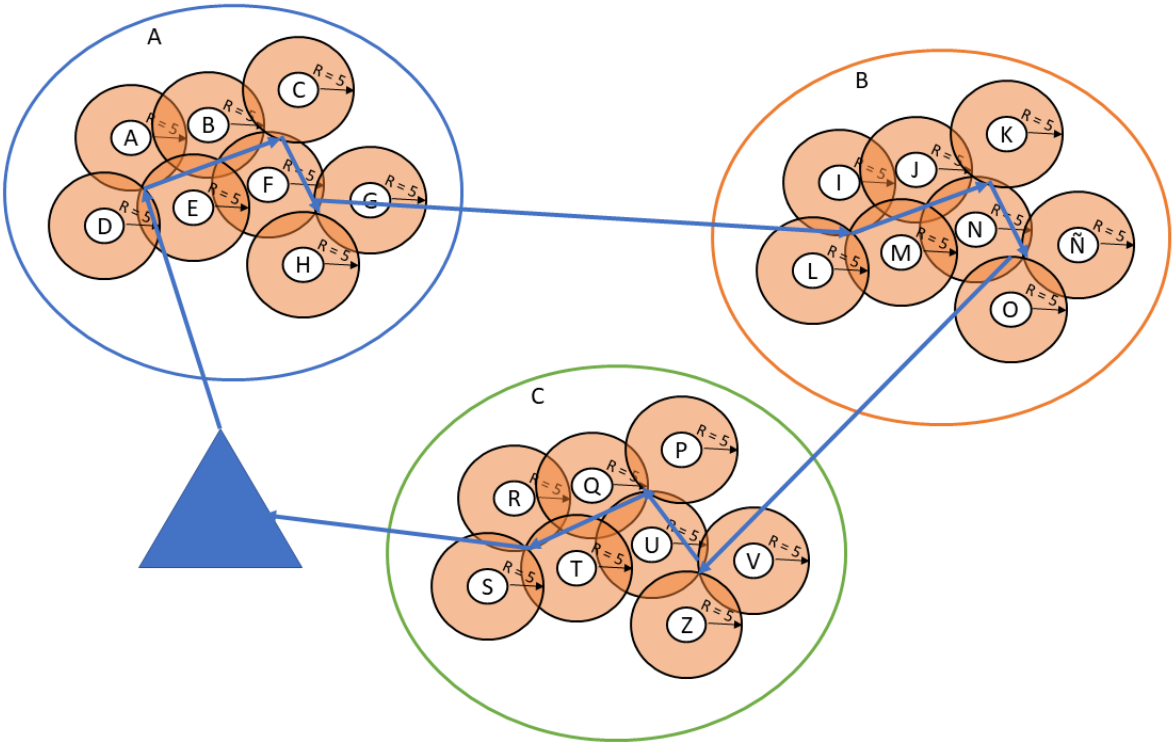

Figure 14: TSP Close Enough con clústers. Fuente: elaboración propia.

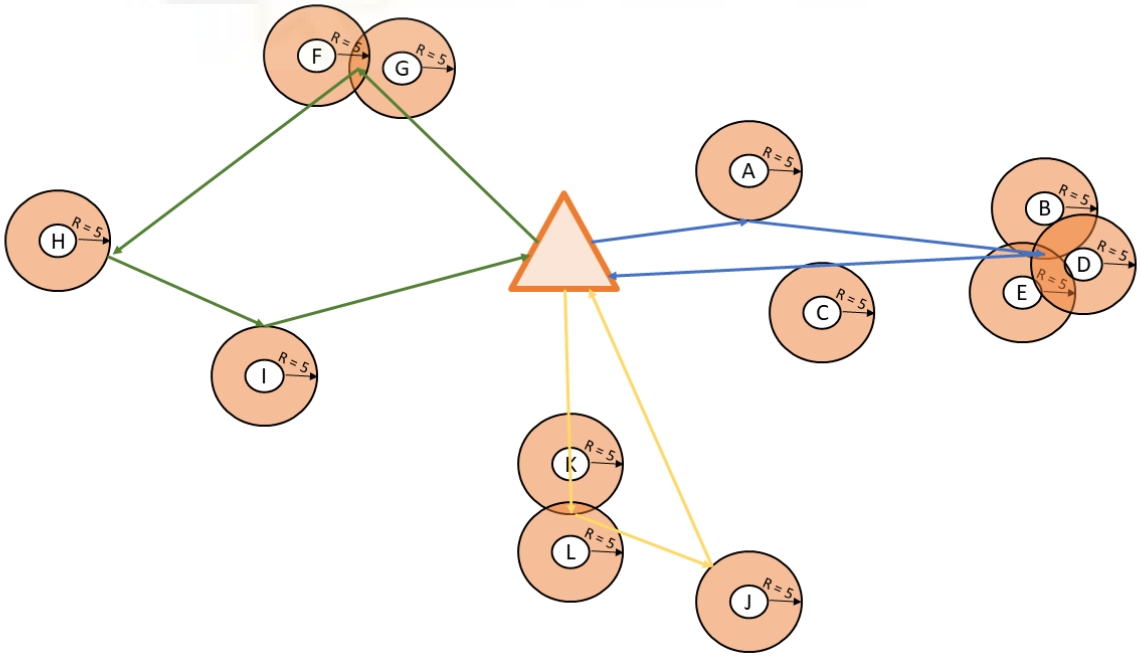

Figure 15: Vehicle Routing Problem en un Close Enough. Fuente: elaboración propia.

2.3 Revisión de las diferentes constantes para el TSP

Alternativamente a los anteriores apartados, existe una fórmula mediante la cual podríamos estimar la longitud de la ruta sin calcular la propia ruta. A priori, la fórmula es la siguiente:

$$L^* = \beta * \sqrt{An}$$

Donde β es una constante, A el área del polígono donde se encuentran los nodos y n, el número de puntos o nodos. El requisito básico para que se cumpla esta condición, es que los nodos son puntos al azar en la región del área A.

A lo largo de los años, diferentes autores le han dado diferentes valores a la constante. Así como diferentes acotaciones. Por ejemplo, Marks (1948) demostró que $L^* \geq \sqrt{nA} - 1/(2\sqrt{n})$, es decir, $\beta \geq (n-1)/(2n)$. Más tarde, Verblunsky (1951), demostró que $L^* \leq 2 + \sqrt{2.8n}$ para una unidad cuadrada. Es decir, $\beta \leq 2/\sqrt{n} + \sqrt{2.8}$. Posteriormente, Few (1995) definió que $L^* \leq \sqrt{2n} + 1.75$ para cualquier conjunto de n puntos en la unidad cuadrada. Alternativamente, $\beta \leq \sqrt{2n} + 1.75/\sqrt{n}$. Stein (1978) propuso una estimación ligeramente mayor, $\beta = 0.765$.

En la práctica, más allá de esta simple fórmula, parece que funciona mejor si estimamos esta constante en función de n. Para ello, veamos cómo los diferentes autores han adaptado la fórmula.

Antes de ver los modelos vamos a ver las variables independientes más destacadas. (No todos los autores las utilizarán todas).

Lista de variables independientes	
Variable	Definición
A	Área de la región.
D	Distancia media que hay entre los puntos de la subregión al origen en línea recta.
N	Número de puntos (nodos más el origen) en un TSP.
R	Área del rectángulo más pequeño que envuelve a todos los nodos menos el origen.
R'	Área del rectángulo más pequeño que envuelve a todos los nodos y el origen.
S	Relación entre longitud y altura de la región. S $\geq$ 1.

Daganzo, en 1984 desarrolló una serie de fórmulas de la distancia para el problema del coche en ruta (VRP). Éstas determinan la longitud de una ruta por los diferentes puntos N, localizados en una subregión de A la cual no incluye el origen. C es el número máximo de paradas que puede efectuar cada vehículo. La densidad de los puntos vendría dada por la fórmula $\delta = N/A$. La fórmula para calcular esta distancia sería la siguiente:

$$L^* \approx 2D(N/C) + 0.57(N\delta^{1/2})$$

Adaptado a un TSP, la fórmula quedaría de la siguiente forma:

$$L^* \approx 2D + 0.57(NA)^{1/2}$$

Más recientemente, Chien puso a prueba la exactitud de 7 diferentes modelos para estimar la distancia. Para ello, utilizó 16 formas diferentes para la región, rectangulares, triangulares... Ubicó el almacén en el inicio. Generó 4160 problemas aleatoriamente con un número de nodos entre 5 y 30. Tras un exhaustivo análisis, determinó que la constante que menor error MAPE tenía era de 0.88. Es decir, $L^* \approx 0.88(NA)^{1/2}$. El modelo resultante es el siguiente:

$$L^* = 2.1D + 0.67[R(N-1)]^{1/2}$$

Según Chien, este modelo debe ser capaz de proveer una certera aproximación a la distancia de la ruta, con o sin el conocimiento de las formas de la región. Desarrollando los modelos, Chien consideró 8 variables independientes, pero limitó los modelos finales a solamente 1 ó 2 variables. Al parecer, este exhaustivo proceso de búsqueda solía encontrar los mejores valores para los coeficientes, pero computacionalmente, era difícil construir modelos más grandes de manera eficiente. Por ello en sus problemas, usó una cantidad de nodos pequeña.

Kwon et al. en 1995, usa la siguiente aproximación:

$$L^* = 0.41R + [0.77 - 0.0088(n+1) + 0.9S/(n+1)]\sqrt{nA}$$

Cuando el origen esta localizado en el centro del triángulo, la aproximación se convierte en:

$$L^* = 1.15R + [0.79 - 0.0012(n+1) + 0.97S/(n+1)]\sqrt{nR'}$$

Çavdar y Sokol en 2015 propusieron una fórmula aproximada la cual, en contraste con Beardwood et al. (1959), no precisa saber la dispersión de los nodos. La fórmula es la siguiente:

$$L^* = 2.791\sqrt{n(cstdev_x * cstdev_y)} + 0.2669\sqrt{n(\sigma_x * \sigma_y)A/(\bar{c}_x\bar{c}_y)}$$

Los parámetros $\bar{c}_x$ y $\bar{c}_y$, representan las distancias medias de los puntos a los ejes vertical y central.
σ_x, σ_y, son las desviaciones estándar de los puntos, desde las coordenadas vertical y horizontal.
$cstdev_x$, $cstdev_y$, son las desviaciones estándar de las distancias absolutas, desde el centro de los ejes vertical y horizontal.

En contraposición, estas aproximaciones no siempre funcionan. Hindle y Worthington, en 2004 expusieron un estudio de la comunidad de servicios de enfermería enfocado en el medio rural del norte de Irlanda. En este caso, los datos mostraban grupos de puntos en los que las fórmulas ya usadas no se ajustaban, por lo que tuvieron que modificarla complejamente.

Aplicaciones

Distritos
Este problema se basa en la división de un territorio en distritos para diversos fines, como por ejemplo, un servicio postal. Aplicaríamos la fórmula para estimar el coste esperado del reparto postal en ese distrito. Así podríamos establecer también las cargas de cada mensajero y distrito.

Otro de los diferentes aspectos de hacer distritos, es el caso de la circunscripción política. Con esta fórmula, podríamos evaluar en cuantos distritos debemos dividir el territorio dado el número de habitantes y su dispersión.Veamos un ejemplo gráfico de los distritos de Nueva York (Figura 16):

Figure 16: Distritos de la ciudad de NY. Fuente: Google

Localización
En los problemas de localización, los modelos de aproximación continua sirven para estimar el coste de servicio dentro de esa área. Especialmente cuando no conocemos a ciencia cierta la localización de los nodos. Aplicado a un caso real, podríamos hablar de una compañía de internet, la cual quiere hacer una estimación de lo que le va a costar realizar la instalación de fibra óptica en un área determinada.

Composición de flotas
El problema trataría de atender a un gran conjunto de clientes con una flota de vehículos propia o subcontratada. En el primer caso, los coches realizarían rutas TSP. Se incurrirían costes fijos de los vehículos y los costes de realizar la ruta. La fórmula entra a formar parte de esto, al estimar cuáles serían los costes de dicha ruta.

Por ejemplo, se puede poner el caso de una agencia de autobuses, la cual atiende a una gran demanda. Dados n clientes, estima con esta fórmula cuáles serían los gastos de la ruta, dadas x paradas.

Enrutamiento del vehículo
Este caso se utiliza en aplicaciones VRP. Por ejemplo, Figliozzi (2010) aplicó un modelo de aproximación continua derivado de Daganzo (1984) donde estudiaba el efecto de la congestión de la ruta del vehículo en el los costes de la ruta. Demostró que con tasas de congestión altas, no solo se incrementaba el tiempo de la ruta, sino también la distancia. Es decir si tuviésemos más congestión de nodos que visitar, no solo incrementaría el tiempo, sino además, la distancia. Todo esto se traduce en un mayor coste.

2.4 Implementación en R

La implementación en R por parte del TSP, se resume a la utilización de un paquete. En este paquete, tenemos varias casuísticas acerca del problema. Por ejemplo, podemos tratar un TSP asimétrico o simétrico.

Para realizar la orden del asimétrico, utilizaríamos:

```
ATSP(x, labels = NULL, method = NULL)
```

En cambio, para un TSP simétrico:

```
TSP(x, labels = NULL, method = NULL)
```

Siendo los diferentes parámetros de entrada:

x, object: un objeto, matriz.
labels: etiqueta opcional del nombre de las ciudades o nodos.
method: nombre opcional de la métrica utilizada.
col: esquema de color.
order: orden de las ciudades en un vector de números enteros.

Tenemos otra opción para crear un TSP euclídeo.

```
ETSP(x, labels = NULL)
```

Estos objetos están internamente representados en una matriz. Este solver también nos permite introducir ciudades "dummy" en el objeto TSP o ATSP. Este caso puede ser usado para transformar el problema a un tamaño menor. El comando sería el siguiente:

```
insert_dummy(x, n = 1, const = 0, inf = Inf, label = "dummy")
```

x: un objeto de la clase TSP o ATSP.
n: número de ciudades dummy.
const: distancia de las ciudades dummy a otras ciudades.
inf: distancia entre ciudades dummy.
label: etiquetas de las ciudades dummy.

Para ejecutar el solver y hallar la solución óptima, deberemos ejecutar el siguiente comando:

```
solve_TSP(x, method = NULL, control = NULL, ...)
```

Por otra parte, tenemos el siguiente comando con el que haremos una permutación del vector que contiene el orden en el que se visitan las ciudades:

```
TOUR(x, method=NA, tsp=NULL)
```

Por último, para saber la longitud de la ruta, tenemos esta otra orden:

```
tour_length(x, tsp = NULL, ...)
```

A continuación, vamos a resolver 3 problemas TSP. En dos de ellos, probaremos este paquete de R. Para el otro, modelizaremos el modelo TSP compacto en R. Para comprobar que tanto si usamos el solver como si no hayamos la misma solución, generaremos 10 puntos aleatorios cuyas coordenadas estén limitadas entre 0 y 100.

Ejemplo con 10 puntos aleatorios y paquete TSP

(Todos los comandos utilizados en este apartado están reflejados en el anexo).

Calculamos los 10 puntos aleatorios en el plano. Las coordenadas son las siguientes:

	longitud	latitud
1)	79.778432	79.79891
2)	75.356509	59.37940
3)	39.125568	90.53100
4)	34.155670	88.08486
5)	36.129411	99.38366
6)	19.834473	89.59563
7)	53.485796	87.86434
8)	9.652624	19.76057
9)	98.784694	33.49936
10)	16.756948	77.72063

A partir de estos puntos, calculamos las distancias entre cada uno:

$$\begin{pmatrix} 0 & & & & & & & & & \\ 20.893 & 0 & & & & & & & & \\ 42.046 & 47.782 & 0 & & & & & & & \\ 46.369 & 50.215 & 5.539 & 0 & & & & & & \\ 47.841 & 56.028 & 9.346 & 11.470 & 0 & & & & & \\ 60.739 & 63.212 & 19.314 & 14.401 & 19.009 & 0 & & & & \\ 27.502 & 35.913 & 14.606 & 19.331 & 20.831 & 33.696 & 0 & & & \\ 92.316 & 76.725 & 76.662 & 72.585 & 83.910 & 70.573 & 80.991 & 0 & & \\ 50.049 & 34.909 & 82.534 & 84.596 & 90.920 & 96.850 & 70.764 & 90.185 & 0 & \\ 63.056 & 61.403 & 25.777 & 20.252 & 29.062 & 12.267 & 38.104 & 58.394 & 93.188 & 0 \end{pmatrix}$$

Con estos números, el solver encuentra la solución representada en la Figura 17. La distancia total o coste incurrido es de 293.972 .

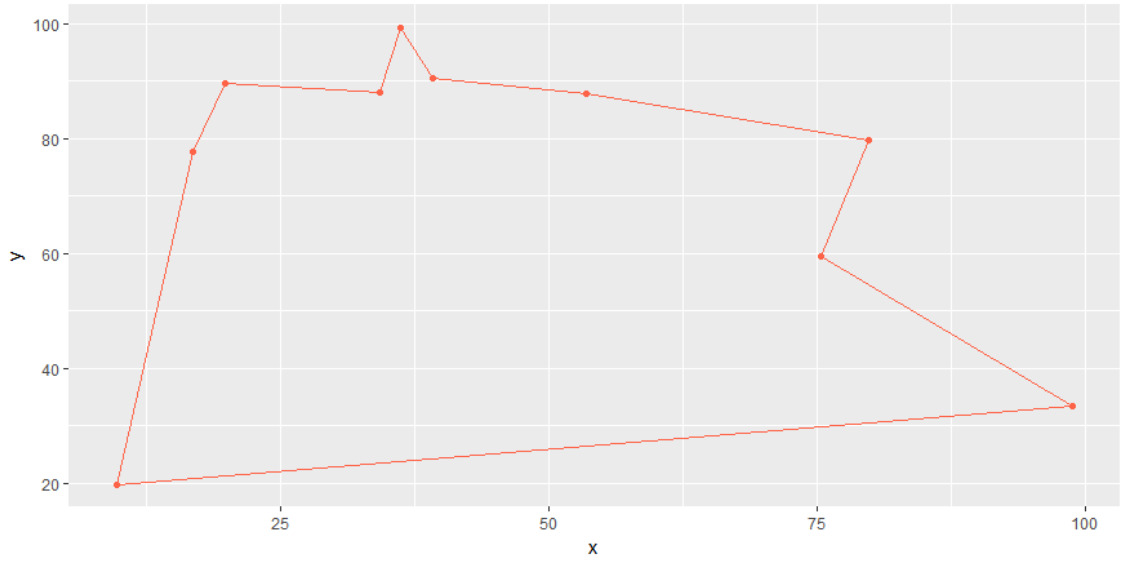

Figure 17: Solución óptima al ejemplo con el paquete TSP. Fuente: elaboración propia.

Ejemplo con 10 puntos aleatorios sin solver

En este caso, como hemos dicho anteriormente, vamos a utilizar los mismos puntos que en el caso anterior para verificar que tanto el paquete como la modelización a mano, ofrecen la misma solución.

Tras programar el modelo en R, alcanzamos un óptimo cuya distancia total, es de 293.972. Coincide exactamente con el ejemplo realizado con el solver, como podemos ver en la Figura 18

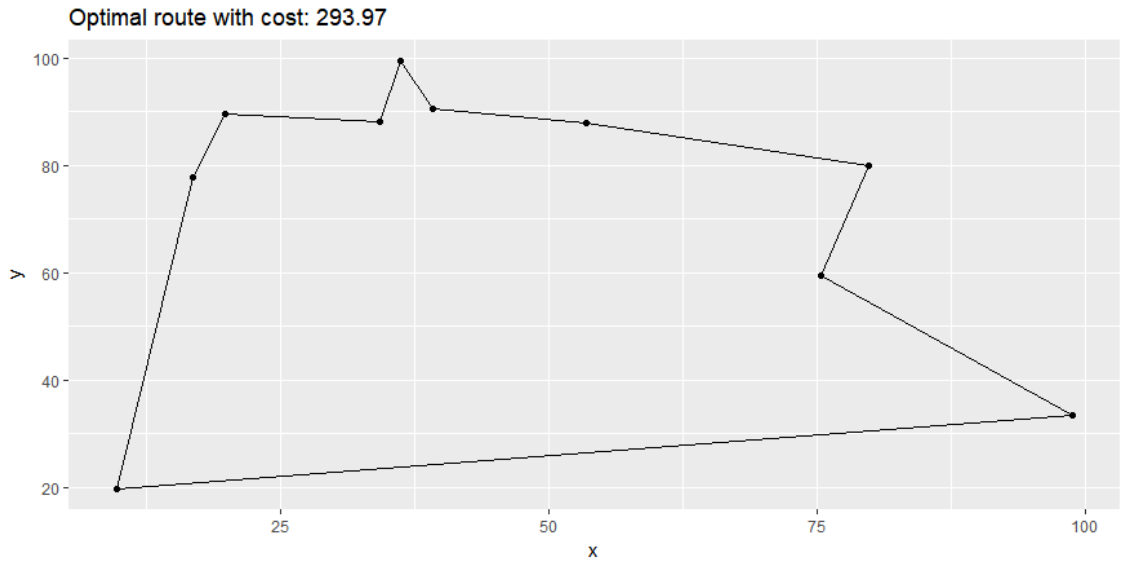

Figure 18: Solución óptima al ejemplo sin solver. Fuente: elaboración propia.

Ejemplo del paquete de R, TSP

Hemos visto dos problemas TSP simulados. Ahora, vamos a ver un ejemplo más gráfico. El paquete TSP nos proporciona un conjunto de datos a modo ejemplo. En concreto, tenemos 312 ciudades. Además, podemos dibujar los datos en un mapa. La distancia total de la ruta es 40856. Veamos la ruta en la Figura 19.

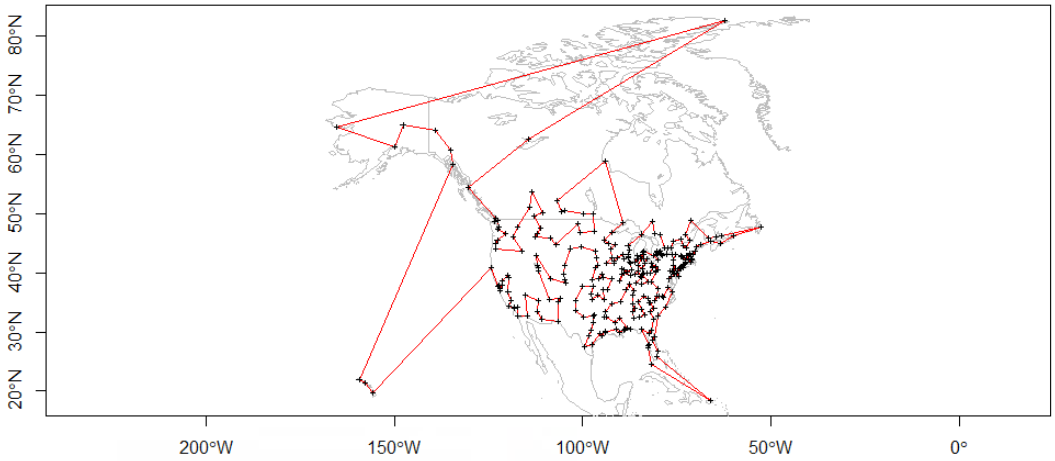

Figure 19: Representación gráfica del ejemplo facilitado por el paquete de R, TSP. Fuente: elaboración propia.

Ejemplos para los modelos de aproximación continua

Como hemos visto en el apartado anterior, podríamos hacer una estimación de la solución del TSP con modelos de aproximación continua. Veamos como funcionan todas las variaciones y estimaciones anteriormente mencionadas.

Lista de fórmulas	
Autor	Fórmula L^*
Fórmula general	$L^* = \beta * \sqrt{An}$
Marks (1948)	$L^* \geq \sqrt{nA} - 1/(2\sqrt{n})$
Verblunsky (1951)	$L^* \leq 2 + \sqrt{2.8n}$
Few (1995)	$L^* \leq \sqrt{2n} + 1.75$
Stein (1978)	$L^* = 0.765 * \sqrt{An}$
Daganzo (1984)	$L^* \approx 2D + 0.57(NA)^{1/2}$
Chien (1992)	$L^* = 2.1D + 0.67[R(N-1)]^{1/2}$
Kwon et al. (1995)	$L^* = 0.41Dx + [0.77 - 0.0088(n+1) + 0.9S/(n+1)]\sqrt{nA}$
Çavdar y Sokol (2015)	$L^* = 2.791\sqrt{n(cstdev_x * cstdev_y)} + 0.2669\sqrt{n(\sigma_x * \sigma_y)A/(\bar{c}_x\bar{c}_y)}$

Al igual que en ejemplos anteriores, usaremos 10 nodos ubicados en un plano 100x100. Veamos cómo se comportan las diferentes fórmulas definidas.

Marks (1948)

$L* \geq \sqrt{10 * 100^2} - 1/2\sqrt{10}$ $;L^* \geq 142.3025$

Verblunsky (1951)

$L^* \leq 2 + \sqrt{2.8 * 10}$ $;L^* \leq 729.1503$

Few(1995)

$L^* \leq \sqrt{2 * 10} + 1.75$ $;L^* \leq 1589.214$

Stein (1978)

$L^* = 0.765 * \sqrt{100^2 * 10}$ $;L^* = 241.9142$

Daganzo (1984)

$L^* \approx 2D + 0.57(NA)^{1/2}$ $;L^* = 270.4119$

Chien (1992)

$L^* = 2.1D + 0.67[R(N-1)]^{1/2}$ $;L^* = 263.9997$

Kwon et al. (1995)

$L^* = 0.41D + [0.77 - 0.0088(n+1) + 0.9S/(n+1)]\sqrt{nA}$ $;L^* = 246.7188$

Çavdar y Sokol (2015)

$L^* = 2.791\sqrt{n(cstdev_x * cstdev_y)} + 0.2669\sqrt{n(\sigma_x * \sigma_y)A/(\bar{c}_x\bar{c}_y)}$ $;L^* = 1.312832e-53$

Como podemos observar, los métodos más aproximados al ejemplo realizado con el solver, son los de Kwon et al, Chien, Daganzo y Stein. El resto, aunque no erróneos, no dan una estimación muy precisa. Como por ejemplo, el caso de Few, que la longitud de la ruta va a ser inferior a 1589.214. Dista mucho del resultado real (293.972).

Más tarde, nos basaremos en algunos de estos autores para intentar dar una estimación al close enough TSP, con un modelo de aproximación continua.

3 El problema de la longitud para el close enough

3.1 Descripción

El problema del viajante de comercio con rutas suficientemente próximas, es uno de los problemas de la investigación operativa que está siendo aplicado en la actualidad cada vez más. Casos como los de las empresas de logística u operaciones con aeronaves no tripuladas, usan este problema para el día a día.

Como anteriormente hemos mencionado, no existe un solver como tal que le de una rápida solución al problema. Por otra parte, tampoco se han implementado los modelos de aproximación contínua a esta variación del problema TSP.

En esta sección, vamos a realizar una modelización para dar solución a este problema, así como intentar ajustar la constante de los modelos de aproximación contínua para que pueda ser de utilidad al estimar la distancia como alternativa o paso previo a dar solución al problema close enough TSP.

Al igual que antes, vamos a usar el software R Studio para ello.

3.2 Implementación en R

La implementación en R para el Close Enough TSP, ya no se resume a la utilización de un paquete, sino que tenemos que modelar a mano un modelo para hallar la solución del problema. Para ello, hemos intentado dar solución al problema con las siguientes librerías:

- library(rmpk)
- library(ROI.plugin.alabama)
- library(ROI.plugin.glpk)
- library(ROI.plugin.quadprog)

Partimos de la base del TSP, pero adaptando las diferentes ecuaciones al problema que nos ocupa. Después de intentarlo varias y de diferentes formas, concluimos afirmando que por ahora, no es posible la implementación con estos paquetes, ya que tenemos constantes problemas con las partes cuadráticas de las diferentes ecuaciones. Cabe decir, que este paquete está en desarrollo. Más adelante, puede que sí sea posible mediante este método.

A modo alternativo, proponemos un método con el cual aproximar la posible solución. Generaremos 10 puntos aleatorios en el plano y de aquellos nodos los cuales conformen una zona de Steiner, sea del grado que sea, escogeremos 1 de ellos tan solo. Posteriormente realizaremos un TSP con la selección de puntos resultante.

Veamos un ejemplo (Figura 20):

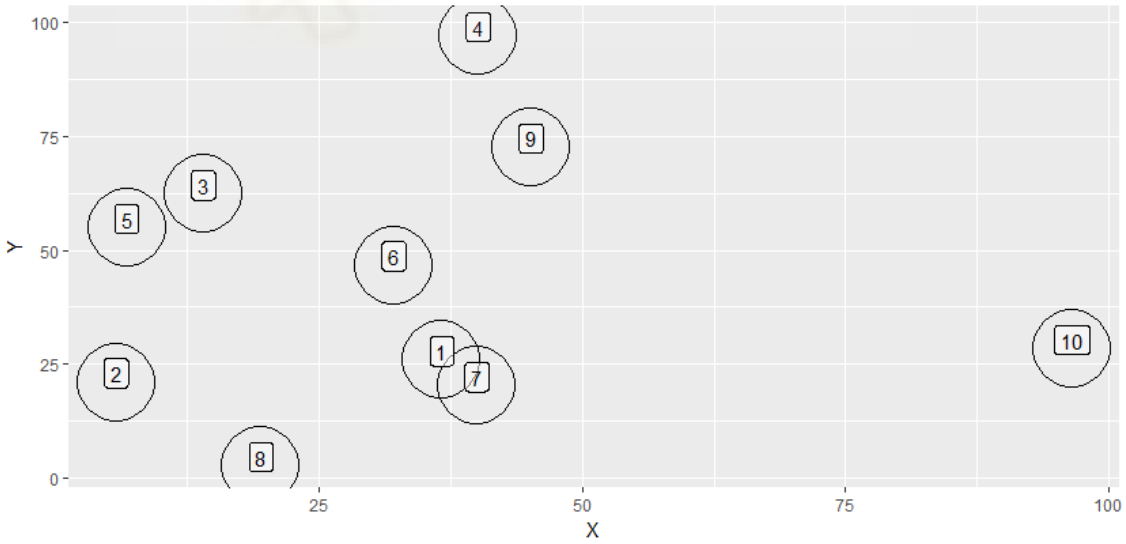

Figure 20: Ubicación de los nodos generados aleatoriamente. Fuente: elaboración propia.

Como podemos ver, tenemos dos nodos que conforman una zona de Steiner de orden 2. En este

caso nos quedaremos con uno de ellos y calcularemos cuál es la ruta más corta. Como el nodo 1 lo tomamos como inicio, el nodo 7 lo damos por visitado al estar en una zona de Steiner. Cuantas más zonas de Steiner tengamos o mayores grados tenga la zona, menor distancia deberemos de recorrer.

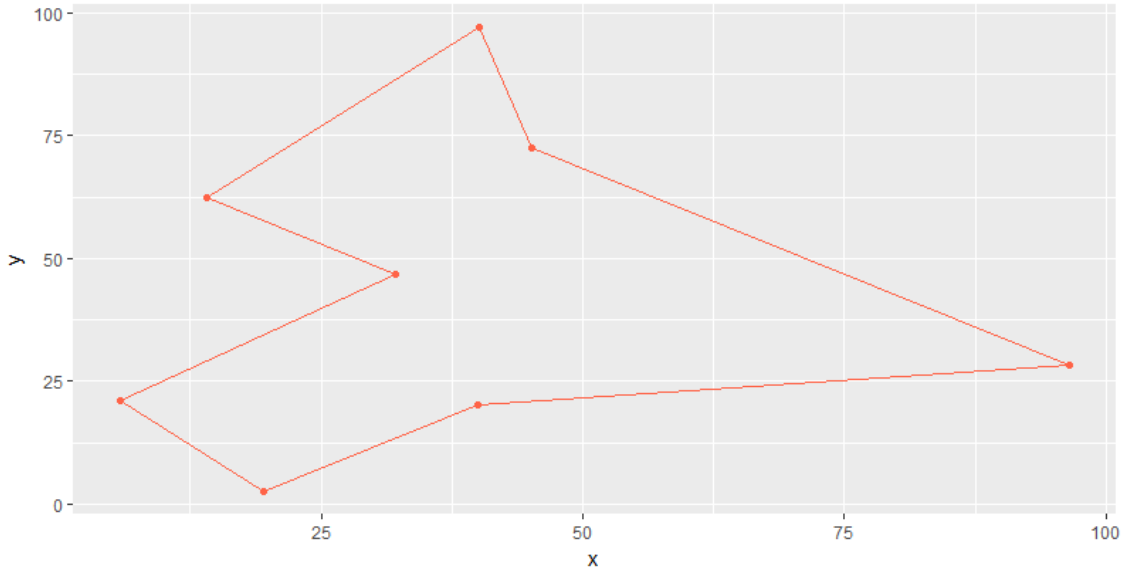

Figure 21: Solución al ejemplo. Fuente: elaboración propia.

En este caso, la longitud de la ruta, se reduce en 29,56 con un coste total de 424,5317 (Figura 21).

Dicho esto, vamos a poner a prueba esta aproximación para ver cómo funciona con diferentes ejemplos, pero utilizando los mismos parámetros:

Resultados de los ejemplos			
Nº ejemplo	Nº de nodos de la zona de Steiner	Coste TSP	Coste CETSP
1	3	454,0864	424,53
2	3	575,2300	414,23
3	4	588,3977	343,4749
4	2	469,5688	458,5539
5	2	540,4710	459,1944
6	3	589,4705	536,0866
7	3	433,3491	414,3971
8	2	613,0516	570,3158
9	4	380,2971	337,3304
10	4	462,8587	375,537

Como vemos, con un close enough TSP, las distancias son mucho menores respecto un TSP. Cabe decir, que todo depende de la dispersión de los puntos. En un plano donde los nodos estén muy dispares, este método puede disminuir la distancia a recorrer, pero no tanto comparado con un caso en el que tenemos ciertas nubes de puntos a los que con una visita, cubrimos el servicio. El apartado de número de nodos de la zona de Steiner hace referencia a la suma de

nodos que están en una zona o varias zonas de Steiner. Es decir, si tenemos 4 nodos, pueden ser 4 en un mismo sitio o ser 2 parejas de puntos diferentes.

Para ver como funciona con otros parámetros, lo que vamos a hacer, va a ser variar el valor del área, del número de nodos y del radio. A continuación, calcularemos la distancia resultante mediante el mismo método anteriormente descrito. Al igual que en todos los ejemplos anteriores, las coordenadas de los puntos serán generadas aleatoriamente, ubicadas dentro del área definida y con el radio que se haya establecido.

Veamos los resultados derivados al cambio en los valores de estos tres parámetros:

Resultados de los ejemplos							
Distancia	Área	Nº nodos	Radio	R	Devx	Devy	D
519.0470	10000	10	10	6240.594	209.84448	226.00127	55.72179
514.4407	10000	10	10	7459.846	250.80101	259.22341	67.57191
405.4596	10000	10	10	6499.445	215.85562	167.25892	46.42675
277.2084	10000	10	20	5747.373	164.94269	140.27864	52.18232
185.1737	10000	10	20	3898.410	102.83914	81.26968	43.36642
568.8845	22500	15	20	9440.517	281.59417	179.11123	55.10129
1140.6616	40000	20	18	28264.386	430.69062	533.00480	138.86187
1493.6295	40000	20	18	26091.399	603.20186	512.67558	119.39105
1017.7333	22500	20	18	19007.154	519.89119	594.84650	69.78170
1094.6907	22500	20	16	16564.237	434.43723	457.99491	62.43015
887.2981	5625	25	7	3710.520	416.66468	388.20576	36.00105
172.1757	5625	5	7	2301.702	65.96693	86.48027	35.18331
224.1764	5625	5	20	2700.906	72.84999	81.66141	33.13147
140.3212	5625	5	30	1765.281	63.74794	48.02526	20.57271
3829.1691	160000	40	30	145171.668	2024.97075	2140.30917	163.08989
3692.4215	160000	40	30	125226.285	2112.35377	1530.98993	228.98512
3730.8854	160000	25	20	126504.534	1411.21296	1593.46847	160.24896
691.3723	5625	20	16	4599.015	352.63842	362.49861	42.25061
806.4558	5625	25	16	4698.649	389.44636	366.24839	31.36596
1439.5262	40000	15	7	30093.329	686.09394	563.67831	121.77828
1541.0243	40000	15	7	30123.690	583.60780	721.74795	82.04347

Como podemos observar, a mayor número de nodos, mayor distancia, igual que si tenemos un radio menor. A mayor radio, menor distancia. Las medidas de dispersión también afectan. A mayor dispersión, mayor longitud de la ruta.

Propuesta de modelo de aproximación continua para el CETSP

En este apartado, vamos a partir de los modelos de aproximación continua para con la intención de dar una estimación del close enough TSP lo más precisa posible. Usaremos los ejemplos anteriores como base de datos con la que empezar a trabajar sobre la fórmula.

Como bien vimos en el apartado de la implementación del TSP, el modelo que más se aproximaba a la solución del problema, era la del autor Chien. Recordemos:

$$L^* = 2.1D + 0.67[R(N-1)]^{1/2}$$

Dado que tenemos n número nodos, los cuales están dispuestos a desplazarse una distancia de r (radio), podríamos decir que la longitud se reducirá r*n. Es decir, si la ruta de un TSP a través de 8 nodos es de 250, si están dispuestos a desplazarse un radio de 5, la ruta minimizará su distancia en 40. Veamos un ejemplo más claro (Figura 22):

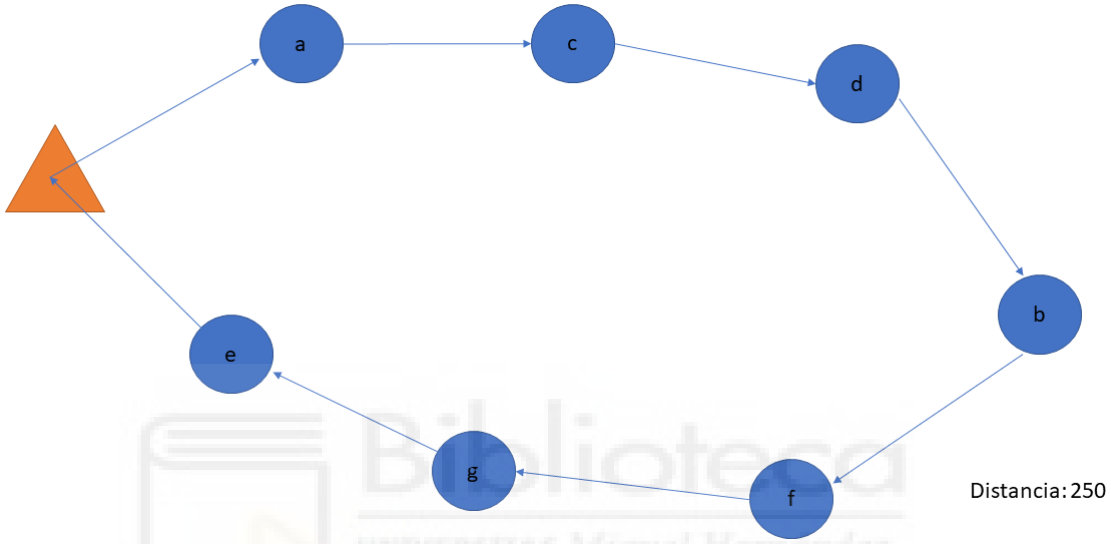

Figure 22: TSP. Fuente: elaboración propia.

Por tanto, si adaptamos la fórmula de Chien para un close enough TSP, la ecuación sería la siguiente:

$$L^* = 2.1D + 0.67[R(N-1)]^{1/2} - r * n$$

Veamos un ejemplo para clarificar este aspecto (Figura 23):
Como anteriormente hemos mencionado, al pasar lo suficientemente cerca del nodo, reducimos la distancia en 5 (radio), 8 veces (8 nodos). Por tanto, la longitud de la ruta se reduce en 40. Este aspecto es válido para rutas con ciertas características. Como hemos visto en los ejemplos, las rutas son circulares. Si cambiara la forma de la ruta, no siempre se ajustaría a la realidad.

Por ello, lo que haremos es crear una base de datos donde se recojan los ejemplos que hemos visto anteriormente, donde las rutas no son todas con forma circular ,con la intención de dar solución al problema mediante una regresión lineal. Trataremos de explicar la distancia a través del área (A), el número de nodos (n), el radio (r), desviaciones medias de las coordenadas (x, y) $(D\bar{x}, D\bar{y})$, área del rectángulo más pequeño que envuelve a los nodos (R) y la distancia media de los puntos al origen (D).

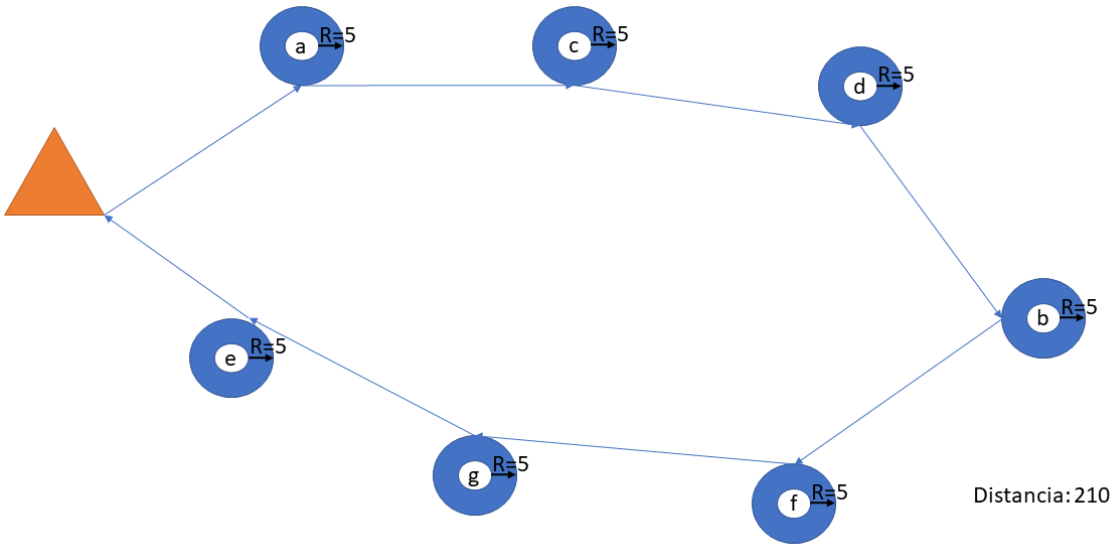

Figure 23: Close Enough TSP. Fuente: elaboración propia.

Tras introducir todas las variables al modelo, usamos uno de los métodos más utilizados en la minería de datos para reducir el número de variables que inciden en el problema, eligiendo las que más influencia tienen en el modelo. La fórmula inicial de la regresión múltiple, con todas las variables, es la siguiente:

$$L^* \approx \sqrt{R} + \sqrt{A} + n + r + D\bar{x} + D\bar{y} + D$$

Tras usar el criterio de selección paso a paso, el software nos indica que las variables más significantes en el modelo son las siguientes:

$$L^* \approx \sqrt{A} + r + D\bar{x} + D\bar{y}$$

La fórmula resultante con la que estimaremos las distancias de las rutas, es la siguiente:

$$-51.0315696 + 3.9706978 * \sqrt{A} - 9.1966490 * r + 0.5911935 * D\bar{x} + 0.7635737 * D\bar{y}$$

Con estos valores, podremos estimar la distancia de la ruta sin saber a priori, cuál es. A continuación, le daremos diferentes valores a los parámetros para ver qué resultados ofrece. A su vez, los compararemos con los de la aproximación al close que anteriormente hemos resuelto. Los resultados deberían asemejarse bastante.

Comparación de resultados					
Distancia Close Enough	Área	Radio	Devx	Devy	Distancia fórmula
1541,024	40000	7	583,6078	721,748	1574,864
1439,526	40000	7	686,0939	563,678	1514,756
519,0470	10000	10	209,8448	226,0013	550,6990
337,3304	10000	10	17,49889	24,98713	283,4965
343,4749	10000	10	37,56259	22,74216	293,6438
568,8845	22500	20	281,5942	179,1112	663,8814
224,1764	5625	20	72,84999	81,66141	168,2607
3692,422	160000	30	2112,354	1530,9899	3679,182
140,3212	5625	30	63,74795	48,02527	45,22949

Como podemos observar, los resultados de la fórmula están en torno a los hallados mediante el método de la aproximación al close enough. Aunque no siempre es así, a simple vista, vemos que a medida que aumenta el radio, menor es la distancia. Esto tiene sentido, ya que si el cliente está dispuesto a desplazarse más, es una distancia que no recorremos. Esto ocurre sobretodo en los casos donde la dispersión es baja.

En casos con dispersiones altas, áreas grandes y un radio pequeño, las longitudes de las posibles rutas, serán más parecidas a las del TSP, pues si los nodos están muy dispersos, implicaría que posiblemente, tengamos pocas zonas de Steiner. Este caso, reflejaría el hecho que se reflejó antes, en rutas circulares donde la ruta se reducía tan solo r*n.

3.3 Aplicaciones reales con rutas

En este apartado, se va a exponer un caso real con la intención de darle solución con los 3 métodos anteriormente descritos. Dadas unas coordenadas, vamos a obtener las distancias del problema, mediante un TSP, un Close Enough TSP y las dos fórmulas de aproximación continua.

Aplicación real TSP

Una empresa de logística, quiere saber si es viable implantar un servicio de mensajería mediante drones en la ciudad de Alicante. Para ello, se ha determinado la ubicación de 19 puntos de entrega, lo que simularía la jornada media de un mensajero en la ciudad. El objetivo, es hallar la distancia mínima que recorra la totalidad de los nodos, ya que los drones tienen una capacidad de batería limitada y no podría recorrer ciertas distancias.

En la siguiente imagen, tenemos la ubicación de los puntos (Figura 24):

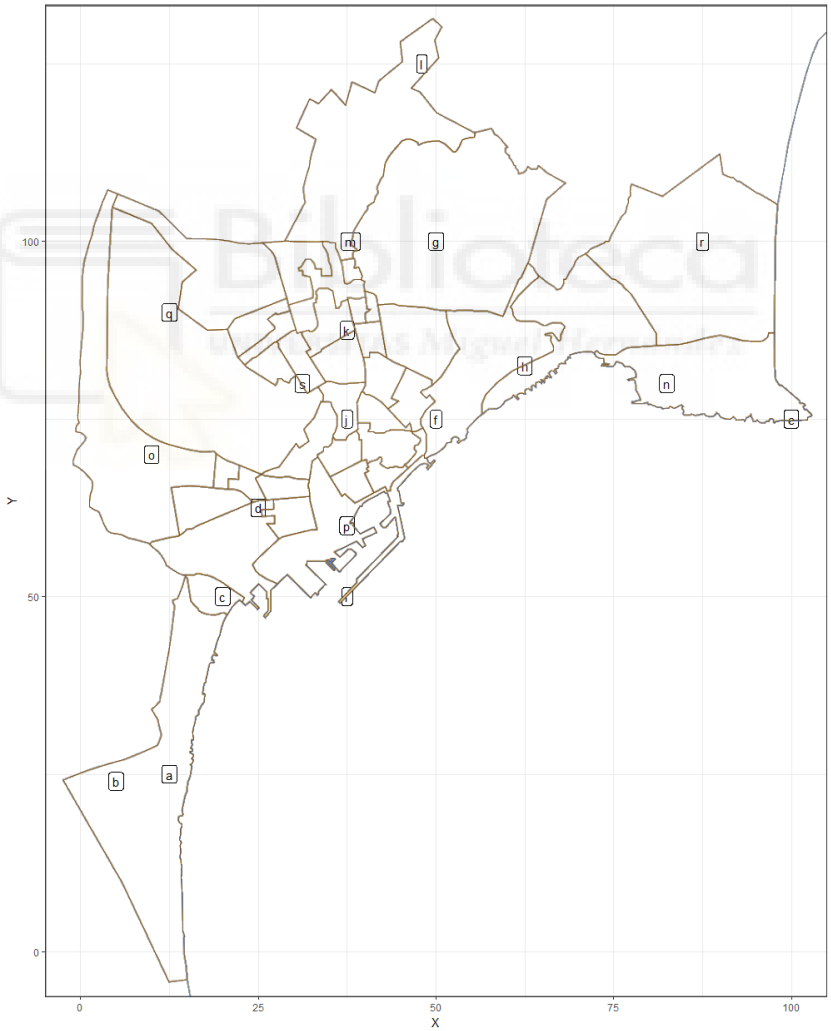

Figure 24: Ubicación de los puntos del problema. Fuente: elaboración propia.

Las coordenadas de cada punto son las siguientes:

	latitud	longitud	x	y
a)	38.299013	−0.522561	12.5	25.0
b)	38.297689	−0.533875	5.0	24.0
c)	38.325683	−0.513668	20.0	50.0
d)	38.341071	−0.504865	25.0	62.5
e)	38.352536	−0.414109	100.0	75.0
f)	38.352159	−0.474099	50.0	75.0
g)	38.382179	−0.471840	50.0	100.0
h)	38.360521	−0.456077	62.5	82.5
i)	38.328057	−0.489530	37.5	50.0
j)	38.352385	−0.485892	37.5	75.0
k)	38.366670	−0.486089	37.5	87.5
l)	38.406283	−0.469104	48.0	125.0
m)	38.383780	−0.487452	38.0	100.0
n)	38.359505	−0.426986	82.5	80.0
o)	38.363491	−0.507077	10.0	70.0
p)	38.339004	−0.488163	37.5	60.0
q)	38.371580	−0.516603	12.5	90.0
r)	38.374319	−0.423377	87.5	100.0
s)	38.365836	−0.496322	31.2	80.0

Como se puede observar, tenemos 19 puntos con sus coordenadas geográficas en formato grado decimal y sus respectivas en un plano 100x127. Se ha tomado la decisión de trasladar las coordenadas reales a un plano de 100x127 con la intención de que los resultados sean lo más parecidos a los ejemplos anteriormente expuestos.

Tras introducir todos los datos en el sistema, la solución es la siguiente (Figura 25):

Esta ruta, tiene una distancia vectorial de 751.7473, cuyo punto de partida es el nodo A. Las soluciones haciendo referencia al modelo matemático serían las siguientes:

$X_{ai} = 1$; $X_{ip} = 1$; $X_{pd} = 1$; $X_{ds} = 1$; $X_{sj} = 1$; $X_{jf} = 1$; $X_{fh} = 1$; $X_{hn} = 1$; $X_{ne} = 1$; $X_{er} = 1$; $X_{rl} = 1$; $X_{lg} = 1$; $X_{gm} = 1$; $X_{mk} = 1$; $X_{kq} = 1$; $X_{qo} = 1$; $X_{oc} = 1$; $X_{cb} = 1$; $X_{ba} = 1$

$U_a = 1$; $U_i = 2$; $U_p = 3$; $U_d = 4$; $U_s = 5$; $U_j = 6$; $U_f = 7$; $U_h = 8$; $U_n = 9$; $U_e = 10$; $U_r = 11$; $U_l = 12$; $U_g = 13$; $U_m = 14$; $U_k = 15$; $U_q = 16$; $U_o = 17$; $U_c = 18$; $U_b = 19$

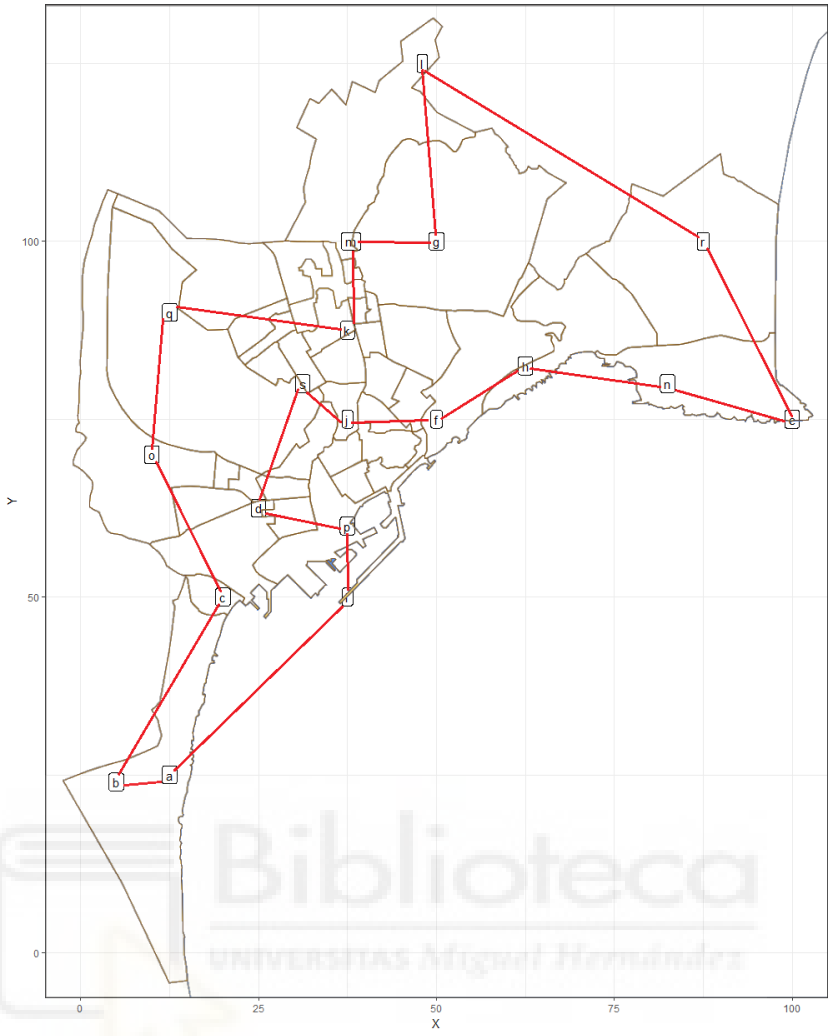

Figure 25: Solución al problema mediante un TSP. Fuente: elaboración propia.

Aplicación real Close Enough TSP

En este caso, se desea obtener la ruta más corta que pase suficientemente cerca de los nodos anteriormente señalados. La empresa precisa saber cuánto se minimizaría la distancia si sus clientes estuviesen dispuestos a desplazarse a un punto cercano. Concretamente, se ha establecido una distancia de 20 unidades de distancia por nodo. Esta pequeña variación, podría suponer un aumento en el número de clientes con distancias similares a un TSP, lo que supondría un aumento de ingresos para la empresa.

Representemos los nodos con su radio para ver claramente las zonas de Steiner y los nodos a tener en cuenta. (Figura 26):

Aparentemente, tenemos varias zonas de Steiner. Al igual que hemos hecho en el apartado que resolvíamos este problema, eliminamos los puntos que pertenecen a alguna zona de Steiner,

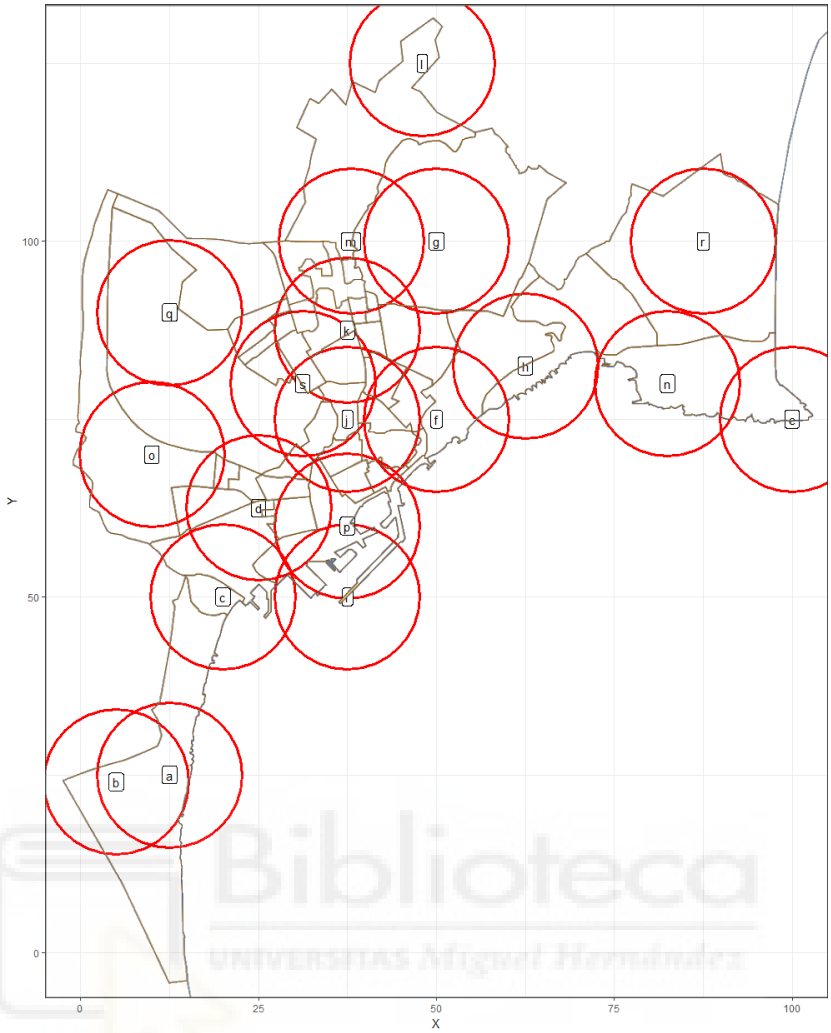

Figure 26: Representación de los nodos con su disponibilidad de desplazamiento. Fuente: elaboración propia.

dejando uno de ellos de tal forma que la ruta pasará por los eliminados. Los nodos omitidos son los siguientes: B,C,D,E,F,H,K,M,O,P,S. Tras realizar el proceso de selección, la ruta es la siguiente: (Figura 27):

La distancia ahora es de 520.6256. La longitud de la ruta se ha reducido considerablemente. Esto puede suponer múltiples ventajas empresariales. Esta estrategia novedosa, reduce considerablemente los costes, hecho que supone un avance frente a la competencia.

Aplicación real modelo de aproximación continua

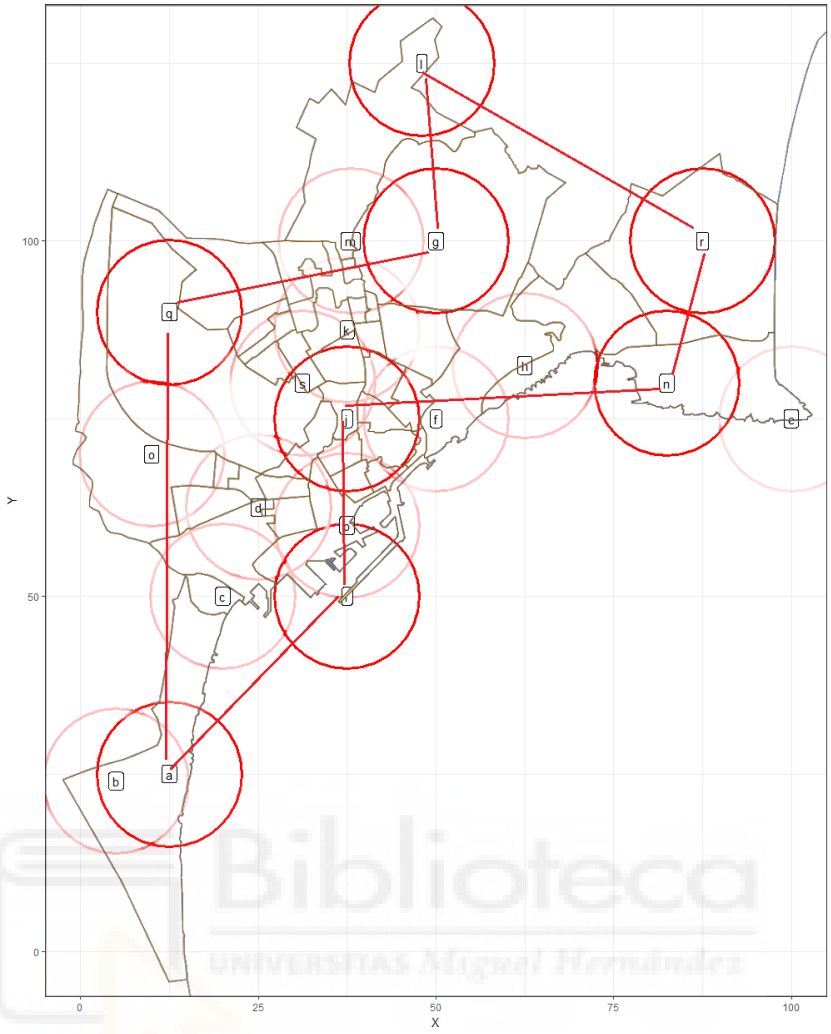

Figure 27: Ruta suficientemente próxima más corta. Fuente: elaboración propia.

Al igual que hicimos anteriormente, ahora vamos a testear qué distancia estiman los modelos de aproximación continua que hemos definido.

Recordemos que nuestra propuesta es la siguiente:

$$L^* = -51.0315696 + 3.9706978 * sqrt(A) - 9.1966490 * r + 0.5911935 * D\bar{x} + 0.7635737 * D\bar{y}$$

Tras calcular los valores de los parámetros, los resultados son estos:

$A = 9595$
$r = 20$
$D\bar{x} = 382.8$
$D\bar{y} = 357.0526$
$D = 60.35206$
$R = 9595$
$n = 19$

Con estos valores, los resultados de ambas fórmulas son los siguientes:

Regresión:

$$L^* = -51.0315696 + 3.9706978 * sqrt(9595) - 9.1966490 * 20 + 0.5911935 * 382.8 + 0.7635737 * 357.0526 = 652.9263$$

Adaptación de Chien:

$$L^* = 2.1 * D + 0.67 * sqrt(R * (n-1)) - r * n = 25.18055$$

El modelo, estima una distancia de 652.9263, que aunque indique más que la estimación del close enough, sigue siendo menor que la de un TSP normal, por tanto sería una buena estimación como punto de partida. En cambio, la fórmula de Chien adaptada, predice una distancia de 25.18055.

Como bien vimos anteriormente, esta adaptación depende en gran medida de como estén dispuestos los nodos, ya que si hay una nube de puntos, no se va a pasar suficientemente cerca de todos, sino que se pueden visitar varios en un solo servicio. Por tanto, en este caso, la adaptación de Chien, no nos ofrecería unos resultados demasiado concluyentes. Nos quedamos con la ecuación que hemos realizado mediante la regresión, que aunque no muy certera, nos devuelve un resultado más realista de la distancia total.

Tanto el método del close enough como el de los modelos de aproximación continua son válidos para estimar el coste como paso previo a calcularlo específicamente. Quizás la fórmula sea más adecuada en aquellas situaciones en las que se use como un paso previo a calcular concretamente el coste de la ruta, ya que es mucho más rápido calcular este número que evaluar todas las posibles rutas y escoger la más corta si no se tiene el software y las herramientas adecuadas. A pesar de esto, resulta muy útil por la simpleza que posee.

4 Conclusiones

A lo largo de este trabajo, se han obtenido multitud de resultados. Tras definir varias formas de abordar un TSP, hemos visto cómo es posible resolverlo de diferentes formas, ya sea con la ayuda de un solver, o modelando variables y parámetros paso a paso. Ambas formas son igual de eficientes, pues llegan a la misma solución.

Tras resolver también una de las adaptaciones del TSP, hemos visto como esta reducía en gran medida la distancia frente al TSP. Esta adaptación supone un gran avance en la reducción de los costes para cualquier empresa que emplee este sistema.
En nuestro caso en concreto, a pesar de no haber podido hallar la solución exacta al close enough TSP, nuestra aproximación ofrece resultados favorables, teniendo en cuenta que nuestro objetivo es minimizar la distancia. De hecho, hemos comparado los resultados frente a un TSP y efectivamente, reducimos la distancia considerablemente.

Por otra parte, los modelos de aproximación continua, nos ofrecen unos resultados lo bastante certeros como para tenerlos en cuenta. Como bien hemos comprobado al comparar ambos métodos anteriores con este, obtenemos resultados lo suficientemente próximos como para tenerlos en cuenta. Por ejemplo, como paso previo a calcular rutas en concreto, ver si es viable o no el proyecto con este método.

Además, tras evaluar las aportaciones de los diferentes autores, conseguimos unificar las diferentes notaciones y probar la eficiencia de estos modelos con un ejemplo concreto. Hemos visto cuan certeras son las aportaciones de los diferentes autores frente a la solución real (calculando la ruta TSP). Para nuestro ejemplo, Kwon, Chien, Daganzo y Stein fueron los que más se acercaron a la solución real del problema.

Otro de nuestros objetivos era ofrecer un modelo de aproximación continua para el close enough TSP. Tras generar una base de datos de entrenamiento con múltiples valores en los parámetros, realizamos una regresión con la que estimar el coste de la ruta. Con estos resultados, comparamos con las estimaciones con las distancias que mostraba el close enough TSP, para ver la eficiencia de nuestro modelo. Finalmente, los resultados fueron favorables. Aunque las estimaciones sobreestiman en algunas ocasiones y subestiman en otras, siempre son razonablemente próximas, por tanto, son resultados a tener en cuenta.

Como bien hemos visto, el problema TSP está muy presente en el día a día. Desde servicios de logística, de transporte o nosotros mismos, utilizamos este sistema, ya sea mediante alguna herramienta o mentalmente. Evidentemente, cada caso, hace su propia adaptación sujeta a sus necesidades, pero en esencia, el problema es el mismo.

Este proyecto puede servir de ayuda en futuras investigaciones en el campo del cálculo de rutas. Con la incipiente innovación en el mundo de los drones, la adaptación close enough TSP, es una perfecta base para implantar esta tecnología en ellos, sobretodo con la automatización de este sector. A partir de aquí, se deberán añadir muchas más restricciones al problema, ya que el dron, a pesar de no tener que ir por carreteras, si tiene obstáculos que evitar, como edificios u otros drones.

Como hemos visto, la librería rmpk de RStudio, está todavía en construcción y no nos ha permitido realizar el cálculo exacto del close enough TSP. En un futuro, este podría ser uno de los puntos de partida para dar una solución precisa y no una aproximación.

5 Anexo: Código R

En esta sección del proyecto, está reflejado todo el código elaborado a lo largo del trabajo. Como ya se ha mencionado anteriormente, el lenguaje de programación está basado en R. Las diferentes librerías que se han utilizado, también están reflejadas en el código. Además de ello, dos de las librerías más importantes para la realización de este trabajo, vienen reflejadas en la bibliografía para un análisis más exhaustivo, como son las librerías "TSP" y la "rmpk".

Ejemplo con datos aleatorios y el paquete de R, TSP.

```
#Librerias:
library(TSP)
library(tspmeta)

#Numero de puntos:
n <- 10

#Extremos del espacio euclideo:
max_x<-100
max_y<-100

set.seed(123456)
 cities  <- data.frame(id = 1:n, x = runif(n, max = max_x), y = runif(n, max = max_y))

#Convertimos las coordenadas en matriz:
coords.mx <- as.matrix(datos)

#Matriz de distancias
dist.mx <- dist(coords.mx)

#Construimos el TSP
tsp.ins <- tsp_instance(coords.mx, dist.mx)
tour <- run_solver(tsp.ins, method="2-opt")

autoplot(tsp.ins, tour)

#Distancia ruta
tour_length(tsp.ins)
```

Ejemplo con datos aleatorios sin el paquete TSP.

```
library(knitr)
library(dplyr)
library(ggplot2)
library(tidyverse)
```

```
#numero de nodos
n <- 10

#Extremos del espacio euclideo
max_x<-100
max_y<-100

set.seed(123456)
cities <- data.frame(id = 1:n, x = runif(n, max = max_x), y = runif(n, max = max_y))

ggplot( cities , aes(x, y)) +
geom_point()

#matriz de distancias
distance <- as.matrix(stats::dist(select(cities , x, y), diag = TRUE, upper = TRUE))
dist_fun <- function(i, j) {
vapply(seq_along(i), function(k) distance[i[k], j[k]], numeric(1L))
}

library(ompr)
model <- MIPModel() %>%
        add_variable(x[i, j], i = 1:n, j = 1:n,
        type = "integer", lb = 0, ub = 1) %>%
        add_variable(u[i], i = 1:n, lb = 1, ub = n) %>%
        set_objective(sum_expr(dist_fun(i, j) * x[i, j], i = 1:n, j = 1:n), "min") %>%
        set_bounds(x[i, i], ub = 0, i = 1:n) %>%
        add_constraint(sum_expr(x[i, j], j = 1:n) == 1, i = 1:n) %>%
        add_constraint(sum_expr(x[i, j], i = 1:n) == 1, j = 1:n) %>%
        add_constraint(u[i] >= 2, i = 2:n) %>%
        add_constraint(u[i] - u[j] + 1 <= (n - 1) * (1 - x[i, j]), i = 2:n, j = 2:n)

model

library(ompr.roi)
library(ROI.plugin.glpk)

#Longitud de la ruta
result <- solve_model(model, with_ROI(solver = "glpk", verbose = TRUE))
result

solution <- get_solution(result, x[i, j]) %>%
filter(value > 0)
kable(head(solution, 3))

paths <- select(solution, i, j) %>%
rename(from = i, to = j) %>%
mutate(trip_id = row_number()) %>%
```

```
tidyr::gather(property, idx_val, from:to) %>%
mutate(idx_val = as.integer(idx_val)) %>%
inner_join(cities, by = c("idx_val" = "id"))
kable(head(arrange(paths, trip_id, 4)))

ggplot(cities, aes(x, y)) +
geom_point() +
geom_line(data = paths, aes(group = trip_id)) +
ggtitle(paste0("Optimal route with cost: ", round(objective_value(result), 2)))
```

Ejemplo con datos proporcionados por el paquete de R, TSP.

```
library("TSP")
#Archivos de datos
data("USCA312")
data("USCA312_map")
data("USCA50")

#Calculamos la ruta
tour <- solve_TSP(USCA312)
tour

#Cargamos herramientas de mapas
library("maps")
library("sp")
library("maptools")

#Dibujamos el mapa
plot(as(USCA312_coords, "Spatial"), axes=TRUE)
plot(USCA312_basemap, add=TRUE, col = "gray")

#Dibujamos la ruta y las ciudades
tour_line <- SpatialLines(list(Lines(list(
Line(USCA312_coords[c(tour, tour[1]),])), ID="1")))
plot(tour_line, add=TRUE, col = "red")
points(USCA312_coords, pch=3, cex=0.4, col="black")
```

Ejemplos para los modelos de aproximación continua

```
#Formula General :
b = 0.765
A = 100^2
n = 10
L = b*sqrt(A*n)

#Marks(1948)
L = sqrt(n*A) - 1/(2*sqrt(n))
```

```
#Verblunsky (1951)
L = 2 + sqrt(2.8*n)

#Few(1995)
L=sqrt(2*n) + 1.75

#Stein (1978)
b = 0.765
L = b*sqrt(A*n)

#Daganzo
L = 2*D + 0.57*sqrt(n*A)

#Chien
L = 2.1*D + 0.67*sqrt(R*(n-1))

#Kwon
L = 0.41*D + (0.77 - 0.0088*(n+1) + 0.9*S/(n+1))*sqrt(n*A)

#cavdar y Sokol
library(stats)
x <- cities$x
y <- cities$y

sigmax <- mean(dt(x, df = Inf))
sigmay <- mean(dt(y, df = Inf))

Cx <- mean(distance[1,])
Cy <- mean(distance[,1])

L= 2.791* sqrt(n*(sigmax * sigmay)) + 0.2669*sqrt(n*(sigmax * sigmay)*A/(Cx*Cy))

n <- 10

# Extremos del espacio euclideo
max_x<-100
max_y<-100

set.seed(123456)
 cities  <- data.frame(id = 1:n, x = runif(n, max = max_x), y = runif(n, max = max_y))
distance <- as.matrix(stats::dist(select(cities, x, y), diag = TRUE, upper = TRUE))
D <- mean(distance[1,])

auxR <- data.frame(cities[2:10,])

#Area rectangulo
planox <- max(auxR$x)-min(auxR$x)
planoy <- max(auxR$y)-min(auxR$y)
```

```
R <- planox * planoy

S <- mean(distance[1,]/100)
```

Propuesta de aproximación al Close Enough TSP

```
#Librerias
library(ggplot2)
library(TSP)
library(tspmeta)
library(tidyverse)

n <- 10
max_x<-100
max_y<-100
r <- 10
D = 2*r

#Generacion de coordenadas aleatorias
#cities <- data.frame(x = runif(n, max = max_x), y = runif(n, max = max_y))

datograf <- data.frame(a=cities$x, b=cities$y, id= 1:n)
ggplot(datograf, aes(a, b))+
        geom_point(size = D, pch = 1)+
        geom_label(aes(label = id), data = datograf, nudge_y = 2, alpha = 0.5) +
        labs(x = "X", y = "Y")

#Desechamos los nodos de la zona de Steiner
cities <- cities[-c(1, 5),]

coords.mx <- as.matrix(cities)
dist.mx <- dist(coords.mx)

tsp.ins <- tsp_instance(coords.mx, dist.mx)
tour <- run_solver(tsp.ins, method="2-opt")
autoplot(tsp.ins, tour)
tour_length(tsp.ins)

R <- ((max(cities$x) - min(cities$x))*(max(cities$y)-min(cities$y)))

sum(abs(cities$x - mean(cities$x)))
sum(abs(cities$y - mean(cities$y)))

distance <- as.matrix(stats::dist(select(cities, x, y), diag = TRUE, upper = TRUE))
D <- mean(distance[1,])
```

Propuesta del método de aproximación continua al Close Enough TSP

```
#Lectura de datos
datos <- read.table("C:/Users/Juan/Desktop/TFG/bancodatos.txt", sep = ";")

#Construccion del modelo:
models <- lm(y ~ sqrt(R) + sqrt(A) + n + r + Devx + Devy + D, data = datos)
summary(models)
anova(models)

#Hayamos el modelo:
modstep <- step(models, direction = "both", trace = 0)
summary(modstep)

#Coeficientes:
modstep$coefficients
```

Caso real

```
library(ggplot2)
library(TSP)
library(tspmeta)
library(tidyverse)
library(readxl)
library("png")

packs <- c("png","grid")
img <- readPNG("map.png")
g <- rasterGrob(img, interpolate=TRUE)

datos <- read_excel("C:/Users/Juan/Desktop/TFG/coordreal.xlsx")
r=20
D=2*r

ggplot(datos, aes(x,y))+
        geom_point(color = "red")+
        geom_point(size = D, pch=1, color = "red", shape=21, stroke = 2)+
        labs(x = "X", y = "Y") +
        geom_label(aes(label = nodo), data = datos)+
        annotation_custom(g, xmin=-Inf, xmax=Inf, ymin=-Inf, ymax=Inf)+
        theme_bw() +
        xlim(0, 100)+
        ylim(0,127)

### TSP
cities <- data.frame(x = datos$x, y = datos$y)
coords.df <- data.frame(long=cities$x, lat=cities$y)
coords.mx <- as.matrix(coords.df)
dist.mx <- dist(coords.mx)
tsp.ins <- tsp_instance(coords.mx, dist.mx)
```

```
tour <- run_solver(tsp.ins, method="2-opt")
autoplot(tsp.ins, tour)
tour_length(tsp.ins)


### Close enough
cities <- cities[-c(2,3,4,5,6,8,11,13,15,16,19),]
coords.df <- data.frame(long=cities$x, lat=cities$y)
coords.mx <- as.matrix(coords.df)
dist.mx <- dist(coords.mx)
tsp.ins <- tsp_instance(coords.mx, dist.mx)
tour <- run_solver(tsp.ins, method="2-opt")
autoplot(tsp.ins, tour)
tour_length(tsp.ins)

### Modelo aprox continua
A <- ((max(cities$x) - min(cities$x))*(max(cities$y)-min(cities$y)))
Devx <- sum(abs(cities$x - mean(cities$x)))
Devy <- sum(abs(cities$y - mean(cities$y)))
-51.0315696 + 3.9706978*sqrt(A) -9.1966490*r + 0.5911935*Devx + 0.7635737*Devy

### Formula chien
distance <- as.matrix(stats::dist(select(cities, x, y), diag = TRUE, upper = TRUE))
D <- mean(distance[1,])
R <- ((max(cities$x) - min(cities$x))*(max(cities$y)-min(cities$y)))
n = length(datos$x)
L = 2.1*D + 0.67*sqrt(R*(n-1))
L-r*n
```

6 Bibliografía

https://dspace.mit.edu/handle/1721.1/5363

https://drum.lib.umd.edu/handle/1903/9822

https://www.redalyc.org/pdf/849/84912053047.pdf

https://cran.r−project.org/web/packages/TSP/TSP.pdf

Belenky, Alexander S. 1998. Operations research in transportation systems ideas and schemes of optimization methods for strategic planning and operations management. 1998. Kluwer Academic Publishers (125−144).

Gregory Gutin and Abraham P. Punnen, 2002. The Traveling salesman problem and its variation. Kluwer Academic.

https://link.springer.com/article/10.1007\%2Fs11750−017−0456−1

https://github.com/dirkschumacher/rmpk/

https://www.sciencedirect.com/science/article/pii/030505489400093N

http://www.terpconnect.umd.edu/~bgolden/recent_presentation_pdfs_links/steiner_problem.pdf

https://revistas.utp.edu.co/index.php/revistaciencia/article/viewFile/7279/4311

https://www.sciencedirect.com/science/article/pii/S0305054818302673

http://www.dipmat2.unisa.it/people/carrabs/www/pdf/2016_CETSP.pdf