TRABAJO FIN DE GRADO

CURSO ACADÉMICO 2023/2024

ANÁLISIS DE PROBLEMAS DE RUTAS DE VEHÍCULOS CON CAPACIDAD

UNIVERSIDAD MIGUEL HERNÁNDEZ

FACULTAD DE CIENCIAS SOCIALES Y JURÍDICAS DE ELCHE

GRADO EN ESTADÍSTICA EMPRESARIAL

Alumna: Mónica López Brufal

Tutora: Mercedes Landete Ruiz

Índice general

1.1 Introducción

Los problemas del viajante de comercio (TSP, por sus siglas en inglés Travelling Salesman Problem) y de rutas de vehículos son dos de los problemas más estudiados en Investigación Operativa. El primero se centra en estudiar problemas de la siguiente clase: un vendedor ambulante debe visitar varias ciudades para vender sus productos y desea conocer cuál es el camino que tiene que seguir para recorrer menos distancia, partiendo de la ciudad que reside, vaya a todas las ciudades y regrese a la ciudad de origen. Por otro lado, están los problemas de rutas de vehículos (más conocidos por sus siglas en inglés VRP, Vehicle Routing Problem) tratan de resolver problemas como el siguiente: una empresa (de correos/repartidor de comida,etc) deben repartir cierto producto entre sus clientes para satisfacer unas demandas y para ellos, se tendría que encontrar la ruta o rutas que repercutan menos costes,partiendo desde el almacén/restaurante, etc, visite cada cliente y regrese al almacén. Como se puede observar, ambos problemas están muy relacionados entre sí. De hecho, el VRP surgió como una extensión del problema del agente viajero. La diferencia básica es que el primero es de una ruta y el segundo de varias. Si en el segundo solo hay un vehículo, coinciden.

Estos dos tipos de problemas que parecen fáciles de resolver son conocidos por su gran complejidad computacional. Como veremos más adelante, ambos están dentro de la categoría NP-duro.La importancia de estos problemas es doble, no solo tienen una complejidad a la hora de resolverlos, si no también a los numerosas situaciones prácticas en las que se pueden aplicarlas.

La primera parte de este trabajo se centra en los problemas del agente viajero y problemas de rutas de vehículos: en su historia, sus aplicaciones, la modelización, en las formas de resolverlos y extensiones sobre estos problemas.

En la segunda parte del trabajo se hace una comparativa con The Waste Collection Vehicle Routing Problem with Time Windows: A Medical Waste Collection Case Study.

La tercera y última parte del trabajo realizó un algoritmo heurístico en el que intento buscar la mejor solución mediante el algoritmo realizado mediante un ejemplo.

1.2.1 Introducción

Los problemas de rutas tratan de encontrar la mejor ruta, la ruta óptima.Con el objetivo de satisfacer al cliente o clientes.

El problema del vendedor viajero (también conocido con estos nombres: problema del vendedor ambulante, problema del agente viajero o problema del viajante, *TSP* por sus siglas en inglés, Travelling Salesman Problem) pretende responder a la siguiente pregunta: ¿cuál es la posible ruta más corta que visita cada ciudad exactamente una vez y al finalizar regresa a la ciudad origen?. Dependiendo del objetivo del problema la pregunta es distinta (menos coste, menos kilómetros,menos tiempo, etc).

En el Problema del Agente Viajero , tiene como finalidad localizar un recorrido que conecte con todos los nodos de una red, pasando por cada nodo ellos tan solo una única vez y regresando al punto de partida, y que minimice la distancia total de la ruta, o el tiempo total del recorrido.

Este problema no necesariamente tiene una única solución. Además, tiene diversas aplicaciones. Un ejemplo es encontrar el camino más rápido para ir de una ciudad a otra en un mapa. En este caso, los vértices representan las ciudades y las aristas las carreteras que las unen.

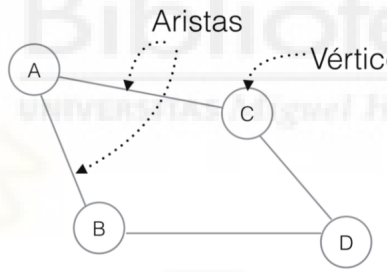


Fig 1: Grafo con aristas y vértices

Este problema tiene una variación importante, y esta depende de que las distancias entre un nodo y otro sean simétricas o no, es decir, que la distancia entre A y B sea igual a la distancia entre B y A.

La cantidad de rutas posibles en una red está determinada por la ecuación:

(n-1)!

Es decir que en una red de 7 nodos la cantidad de rutas probables es igual a (7-1)! =720 , y a medida que el número de nodos aumenta la cantidad de rutas posibles crece factorialmente. En el caso de que el problema sea simétrico la cantidad de rutas posibles se reduce a la mitad, es decir:

((n-1)!) / 2

Lo cual significa un ahorro en el tiempo de procesamiento de rutas de gran tamaño. Esto se encuentra explicado más detalladamente en el ejemplo que se encuentra a continuación.

Ejemplo práctico de cómo se calculan las rutas posibles. Nuestro objetivo en este caso es realizar una enumeración para encontrar la ruta óptima en el que se recorre menos distancia. Supongamos que tenemos 4 ciudades, entre cada par de ciudades tiene ciertos kilómetros entre ellas (los kilómetros se pueden ver en la figura 1).

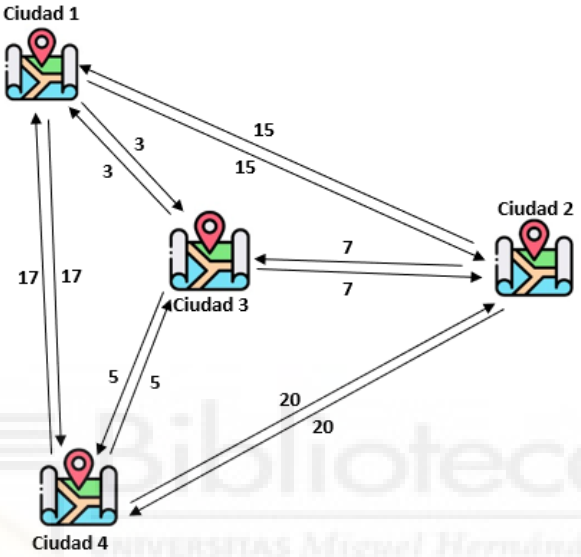


Fig 2: Imagen de ejemplo para calcular la mejor ruta

Como hemos visto anteriormente, como tenemos 4 ciudades para saber las soluciones posibles haríamos *(n-1)!,* (4-1)! = 6 soluciones/recorridos posibles. En este caso podríamos tener las siguientes soluciones:

●Recorrido 1: 1-2-3-4-1. El kilometraje que se recorrería en este primer caso sería: 15+7+5+17= 44 Km.

●Recorrido 2: 1-3-2-4-1. En este segundo caso sería: 3+7+20+17= 47 Km.

●Recorrido 3: 1-3-4-2-1.Tercer caso: 3+5+20+15=43 Km.

●Recorrido 4: 1-4-3-2-1.Cuarto caso: 17+5+7+15=44 Km.

●Recorrido 5: 1-4-2-3-1.Quinto caso: 17+20+7+3=47 Km.

●Recorrido 6: 1-2-4-3-1.Sexto y último caso: 15+20+5+3=43 Km.

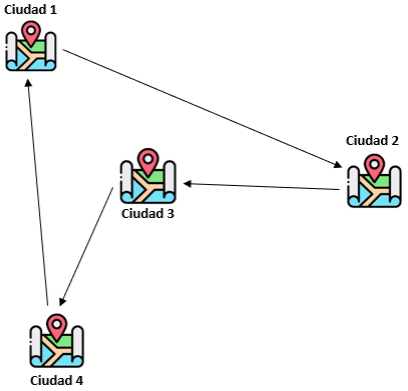


Fig 3: Recorrido 1

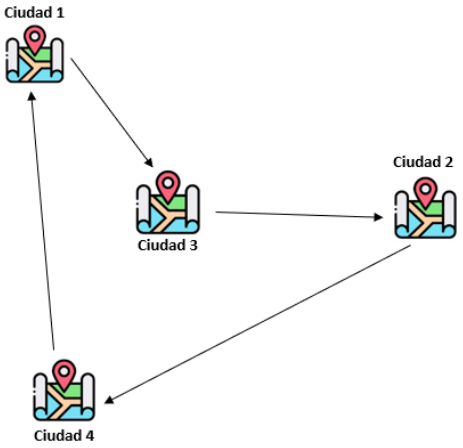


Fig 4: Recorrido 2

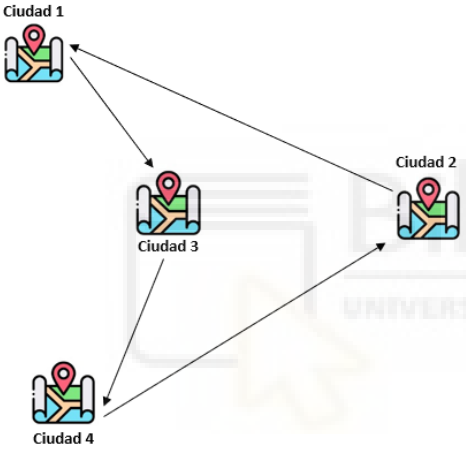


Fig 5: Recorrido 3

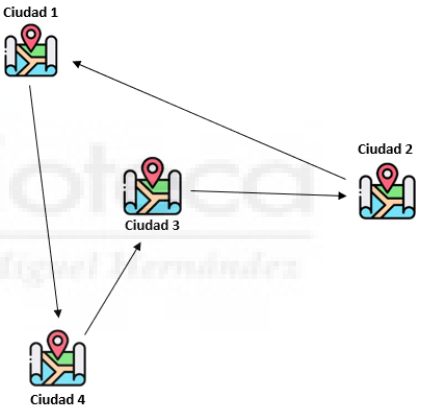


Fig 6: Recorrido 4

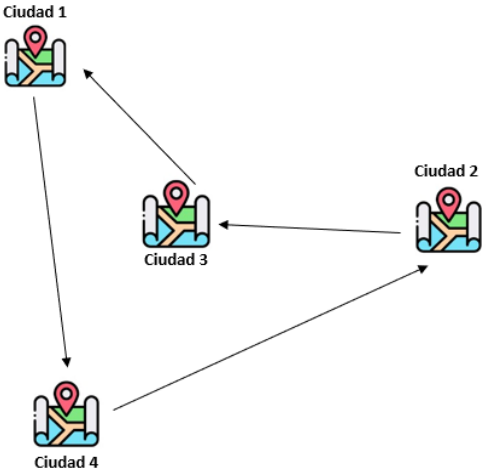


Fig 7: Recorrido 5

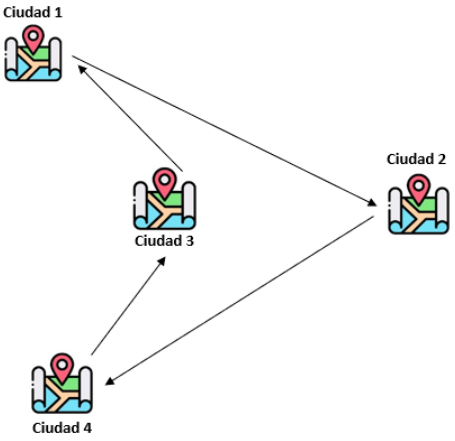


Fig 8: Recorrido 6

Podemos ver que la mejor solución sería las del recorrido 3 y la del recorrido 6. Ahora la pregunta que nos haríamos sería, ¿Sería posible resolver un problema del agente viajero, calculando todas sus posibles soluciones?. La respuesta sería que sí. Pero calcular las soluciones no es tan sencillo. Si en vez de 4 ciudades tuviésemos 10 ciudades, tendríamos que calcular el coste del recorrido de 362.880 soluciones. Y conforme aumenten las ciudades calcularlo computacionalmente se va complicando, y ya no hablar de hacerlo a mano.

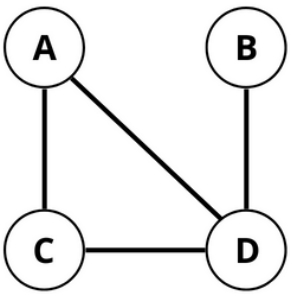
Pero si nos damos cuenta, el problema del agente viajero tiene muchas aplicaciones en la vida real, por ejemplo,recogida de basura, en una fábrica para saber el orden del proceso que se tiene que seguir para no perder tiempo;también en la paquetería, se trata de que los repartidores hagan la ruta más corta en el menor tiempo posible. Otro ejemplo ocurre en la impresión 3D, el cabezal parte de la posición 0, inyectando el material de toda una capa, por lo que tiene que ver cual es la ruta óptima dejando el material y después volver a la posición 0. De estos se podrían poner muchos ejemplos en los que son necesarios este tipo de problemas, y lo necesario que es encontrar la solución óptima.

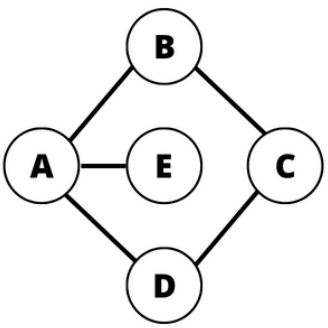
Estos problemas los representamos utilizando grafos, en los grafos tenemos los nodos o vértices, que en nuestro caso son las ciudades o los puntos por donde tiene que ir depositando el material en la impresora 3D. Y por otro lado, tenemos las aristas, también conocidas como arcos que son, la distancia, gasolina, tiempo y el coste. Esto dependerá de lo que queramos resolver.

El grafo que se utiliza normalmente en el problema del agente viajero es un grafo no dirigido, conexo y ponderado.

- No dirigido: sin dirección en las aristas.
- Conexo: para cualquier par de vértices existe al menos un camino posible sin repetir vértices.
- Ponderado: tiene algún valor en sus aristas

Lo que se trata es encontrar el ciclo Hamiltoniano, es decir, encontrar un camino que empieza desde un vértice y termina en ese mismo vértice y que pasa por el resto de vértices sin repetirlos y además con un coste mínimo.





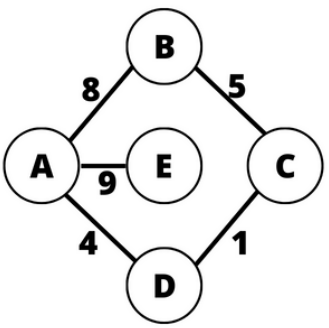


Fig 9: Grafo no dirigido Fig 10: Grafo conexo Fig 11: Grafo ponderado

1.2.2 Origen del problema del agente viajero

Hoy en día no está claro el origen del problema del viajante de comercio.En 1832 se menciona el problema en un manual sin ningún tratamiento matemático con ejemplos de tours por Alemania y Suiza.

Pero este problema fue definido en el siglo XIX, por el matemático irlandés W. R. Hamilton y por el matemático británico Thomas Kirkman.

Inventaron el juego llamado Icosian, que es un juego matemático cuyo objetivo era recorrer las aristas de un dodecaedro (cada vértice representa una ciudad) para pasar una sola vez por cada vértice y que el de partida y llegada tenía que ser el mismo.

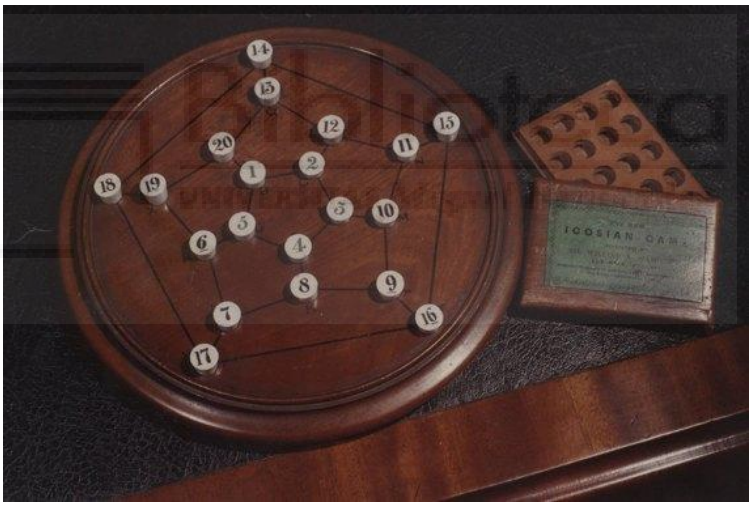


Fig 12:Imagen del juego original

Para que se entienda un poco mejor, pongo un ejemplo. Imaginemos que queremos salir desde la ciudad origen (T) y volver a esa misma ciudad, recorriendo el resto de ciudades una única vez.

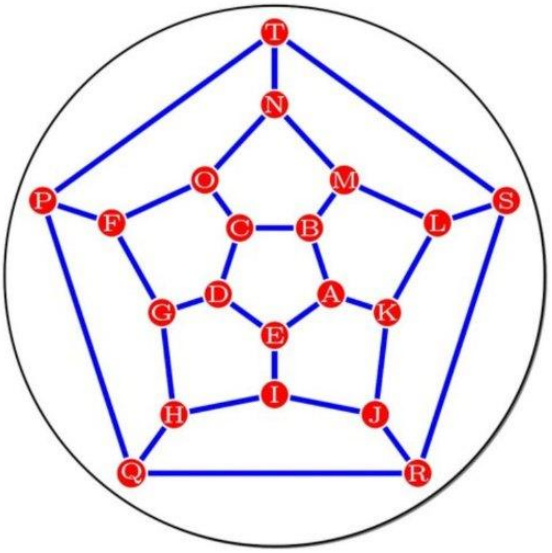


Fig 13: El plano del tablero para resolver el juego

Parece que resolverlo es sencillo, pero lo intente sin ver la solución y no pude evitar mirarla, después de un par de intentos. Os pongo la solución a continuación para que no os quedéis con la integra.

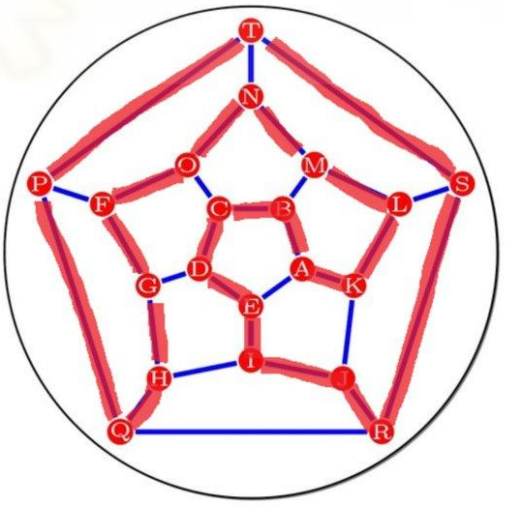


Fig 14: Solución del juego

1.2.3 Aplicaciones del problema del agente viajero

Hasta el día de hoy, el problema del agente viajero se ha aplicado sobre una gran variedad de problemas. A continuación, se comentan brevemente algunas de las aplicaciones más importantes del Problema del Viajante de Comercio.

●Logística: las aplicaciones más abundantes del problema del agente viajero se centran en el campo de la logística. El flujo de personas, mercancías y vehículos en torno a una serie de ciudades o clientes se adapta perfectamente a este tipo de problema. Destacamos:

- ➢ Empresas de paquetería
- ➢ Servicios de comida a domicilio
- ➢ Vendedores. Los comerciales que salen para vender sus productos tienen que salir de la oficina y volver a la misma.
- ➢ Turistas. Las agencias de viajes plantean la mejor ruta para la persona que viaja y regresa al hotel.

●Industria: sus aplicaciones son menores que la anterior, pero también se utiliza para reducir costes. Algunas de sus aplicaciones son:

- ➢ Secuencia de tareas
- ➢ Soldadura
- ➢ Impresión 3D
- ➢ Circuitos electrónicos

Cada día, las áreas donde trabaja el problema del agente viajero son más dispares.

1.2.4 Modelización y formulación del problema del agente viajero

A pesar de ser uno de los problemas más complejos de resolver.Podemos afirmar como hemos dicho anteriormente que el problema del agente viajero tiene solución, pues siempre pueden evaluarse todas las posibles soluciones y escoger la de menor coste. El problema es que el número de posibles soluciones aumenta de manera significativa cuando aumenta el tamaño del problema.

La modelización del problema del agente viajero se puede describir de la siguiente forma: Al tratarse de un grafo G = (V, A), donde V (vértice) = 1, ..., n y A (aristas). Los vértices i = 2, ...corresponden con las ciudades a visitar y el vértice 1 es la ciudad de origen y destino. A cada arco (i, j) se le asocia un valor no negativo cij , que representa el coste de viajar del vértice i al j .Si G es un grafo dirigido, la matriz de costes c es asimétrica mientras que, si cij = cji para todo (i, j) ∈ A, la matriz de costes será simétrica.En ese caso, el conjunto A se sustituye por un

conjunto E de arcos no dirigidos (i,j) tales que i < j. El objetivo del problema del viajante es encontrar una ruta que, comenzando y terminando en una ciudad, en este caso denotada por la ciudad 1, pase una sola vez por cada una de las ciudades y minimice la distancia recorrida. Si definimos las variables de decisión xij para todo (i, j) ∈ A, de forma que tomen el valor 1 si el arco (i, j) forma parte de la solución y 0 en otro caso; tenemos que el problema asociado al problema del viajante, consiste en minimizar la siguiente función objetivo:

$$\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} c_{ij} . x_{ij}$$

● De cada nodo parte uno y solo un arco

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

●A cada nodo llega uno y solo un arco

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

●Y evitando la formación de sub-ciclos (esta restricción indica que para todo subconjunto de los nodos debe haber al menos un arco que “salga”del subconjunto).

$$y_i - y_j + n.x_{ij} \leq n-1, \forall i \in \{1,...,n\}, x, y \in \{0,1\}$$

1.2.5 Algunos métodos de resolución

En general estos métodos se pueden clasificar en dos grandes grupos, por un lado tendríamos los algoritmos exactos y por el otro lado los algoritmos aproximados.

En el primer grupo se trata de realizar todas las posibles combinaciones (método de la fuerza bruta) un ejemplo sería a partir de la figura 2 . Esta posibilidad se convierte en impracticable a medida que aumenta el número de variables. Otra opción relacionada con este primer grupo es realizar la técnica de branch-and-bound (Ramificación y poda).

La técnica de Branch and Bound, se representa como un árbol de soluciones, en las que cada rama nos daría una solución posible. Lo que hace esta técnica es detectar qué ramificaciones no tienen solución óptima y descartarlas (podar esa rama del árbol), y descartarla ya que nos aleja de la solución óptima.

Veamos un ejemplo práctico de esta técnica.Vamos a suponer que tenemos 5 ciudades, cuyas distancias entre cada ciudad está representada en la siguiente matriz:

	Ciudad 1	Ciudad 2	Ciudad 3	Ciudad 4	Ciudad 5
Ciudad 1	0	11	9	10	7
Ciudad 2	11	0	11	6	7
Ciudad 3	9	11	0	9	10
Ciudad 4	10	6	9	0	7
Ciudad 5	7	7	10	7	0

Este algoritmo lo que hace, cual es la mínima distancia en cada una de las filas.Una posibilidad sería si estuviésemos en la ciudad 2, podríamos ir a la ciudad que esté más cerca en este caso la 4. La suma de estas mínimas distancias, será nuestro límite inferior, es decir ninguna ruta en este caso puede ser menor que 35. No puede haber ruta más corta a 35. Cuando ya tenemos este límite, empezamos con las ramificaciones. No tendría porque ser una matriz simétrica.

	Ciudad 1	Ciudad 2	Ciudad 3	Ciudad 4	Ciudad 5
Ciudad 1	0	11	9	10	7
Ciudad 2	11	0	11	6	7
Ciudad 3	9	11	0	9	10
Ciudad 4	10	6	9	0	7
Ciudad 5	7	7	10	7	0

Comenzamos a ramificar, desde la ciudad 1, podemos viajar al resto de ciudades (ciudad 2/3/4 y 5). Valoramos estás cuatro opciones e iremos evaluando.

Si empezamos de 1 a 2, después lo que tenemos que hacer es sumar el mínimo de las siguientes, teniendo en cuenta que no podemos volver, es decir, ir de la 2 a la 1.

Tendríamos una distancia de 40. Realizamos esto con todas las posibilidades:

	Ciudad 1	Ciudad 2	Ciudad 3	Ciudad 4	Ciudad 5
Ciudad 1	0	11	9	10	7
Ciudad 2	11	0	11	6	7
Ciudad 3	9	11	0	9	10
Ciudad 4	10	6	9	0	7
Ciudad 5	7	7	10	7	0

De la ciudad 1 a la 3 tendríamos una distancia de 37 .

	Ciudad 1	Ciudad 2	Ciudad 3	Ciudad 4	Ciudad 5
Ciudad 1	0	11	9	10	7
Ciudad 2	11	0	11	6	7
Ciudad 3	9	11	0	9	10
Ciudad 4	10	6	9	0	7
Ciudad 5	7	7	10	7	0

De la ciudad 1 a la 4 tenemos una distancia de 39.

	Ciudad 1	Ciudad 2	Ciudad 3	Ciudad 4	Ciudad 5
Ciudad 1	0	11	9	10	7
Ciudad 2	11	0	11	6	7
Ciudad 3	9	11	0	9	10
Ciudad 4	10	6	9	0	7
Ciudad 5	7	7	10	7	0

De la ciudad 1 a la ciudad 5 la distancia es de 35.

La ramificación se nos quedaría de la siguiente manera:

31 (LI)			
x12	x13	x14	x15
40	37	39	35

Una vez ya hemos ramificado desde la ciudad 1, tenemos que seguir ramificando, teniendo esto seguimos ramificando desde el número menor. En este caso la ciudad 5, seguimos ramificando.

x15		
x21 (40)	x23 (40)	x24 (38)

	Ciudad 1	Ciudad 2	Ciudad 3	Ciudad 4	Ciudad 5
Ciudad 1	0	11	9	10	7
Ciudad 2	11	0	11	6	7
Ciudad 3	9	11	0	9	10
Ciudad 4	10	6	9	0	7
Ciudad 5	7	7	10	7	0

	Ciudad 1	Ciudad 2	Ciudad 3	Ciudad 4	Ciudad 5
Ciudad 1	0	11	9	10	7
Ciudad 2	11	0	11	6	7
Ciudad 3	9	11	0	9	10
Ciudad 4	10	6	9	0	7
Ciudad 5	7	7	10	7	0

	Ciudad 1	Ciudad 2	Ciudad 3	Ciudad 4	Ciudad 5
Ciudad 1	0	11	9	10	7
Ciudad 2	11	0	11	6	7
Ciudad 3	9	11	0	9	10
Ciudad 4	10	6	9	0	7
Ciudad 5	7	7	10	7	0

También tendríamos que probar con el segundo número más pequeño, que es el x13. Quedaría de la siguiente manera:

x13		
x21 (42)	x24 (39)	x25 (38)

	Ciudad 1	Ciudad 2	Ciudad 3	Ciudad 4	Ciudad 5
Ciudad 1	0	11	9	10	7
Ciudad 2	11	0	11	6	7
Ciudad 3	9	11	0	9	10
Ciudad 4	10	6	9	0	7
Ciudad 5	7	7	10	7	0

	Ciuda	Ciuda	Ciuda	Ciuda	Ciuda

	d 1	d 2	d 3	d 4	d 5
Ciuda d 1	0	11	9	10	7
Ciuda d 2	11	0	11	6	7
Ciuda d 3	9	11	0	9	10
Ciuda d 4	10	6	9	0	7
Ciuda d 5	7	7	10	7	0

	Ciuda d 1	Ciuda d 2	Ciuda d 3	Ciuda d 4	Ciuda d 5
Ciuda d 1	0	11	9	10	7
Ciuda d 2	11	0	11	6	7
Ciuda d 3	9	11	0	9	10
Ciuda d 4	10	6	9	0	7
Ciuda d 5	7	7	10	7	0

Hemos obtenido dos ramas con el mismo coste, tendríamos que seguir analizando cada una, para ver cúal es el recorrido por el que tengo que seguir.

x24	
x35 (39)	x32 (40)

	Ciuda d 1	Ciuda d 2	Ciuda d 3	Ciuda d 4	Ciuda d 5

Ciudad 1	0	11	9	10	7
Ciudad 2	11	0	11	6	7
Ciudad 3	9	11	0	9	10
Ciudad 4	10	6	9	0	7
Ciudad 5	7	7	10	7	0

	Ciudad 1	Ciudad 2	Ciudad 3	Ciudad 4	Ciudad 5
Ciudad 1	0	11	9	10	7
Ciudad 2	11	0	11	6	7
Ciudad 3	9	11	0	9	10
Ciudad 4	10	6	9	0	7
Ciudad 5	7	7	10	7	0

x25	
x31 (41)	x32 (38)

	Ciudad 1	Ciudad 2	Ciudad 3	Ciudad 4	Ciudad 5

Ciudad 1	0	11	9	10	7
Ciudad 2	11	0	11	6	7
Ciudad 3	9	11	0	9	10
Ciudad 4	10	6	9	0	7
Ciudad 5	7	7	10	7	0

Nos da un total de 41.

	Ciudad 1	Ciudad 2	Ciudad 3	Ciudad 4	Ciudad 5
Ciudad 1	0	11	9	10	7
Ciudad 2	11	0	11	6	7
Ciudad 3	9	11	0	9	10
Ciudad 4	10	6	9	0	7
Ciudad 5	7	7	10	7	0

Nos da un total de 38.

Existen varias algoritmos de heurísticos, eston son algunos

●Heurístico del vecino más próximo. Este algoritmo no asegura conseguir una solución óptima, pero sí proporciona buenas soluciones.

1. Cuando ya tenemos de donde salimos, seleccionamos el vecino más cercano. (en este caso seleccionamos de A a la ciudad D).

Ciudad A	B	C	D
Distancia	4	3	2

2. En la siguiente iteración, seleccionamos el más cercano desde D(se excluye A porque es la ciudad origen).

Ciudad D	B	C
Distancia	5	3

3. Por lo tanto obtendremos esta ruta con esta técnica: A-D-C-B-A. Como sabemos que se tienen que visitar todos las ciudades, la colocación de la B es por descarte. Por ello obtendremos una buena solución, pero no quiere decir que sea la óptima.

●Algoritmo de Christofides. Este método es exclusivamente para el problema del viajante de comercio. Fue desarrollado en 1976 por Nicos Christofides. Este algoritmo lo vamos a explicar mediante un ejemplo. Si queremos viajar a varias ciudades, teniendo en cuenta que empezamos y terminamos en la misma ciudad, y nuestro objetivo es reducir costes. Para realizar este algoritmo tenemos que seguir unos pasos, lo vamos a hacer mediante un ejemplo:

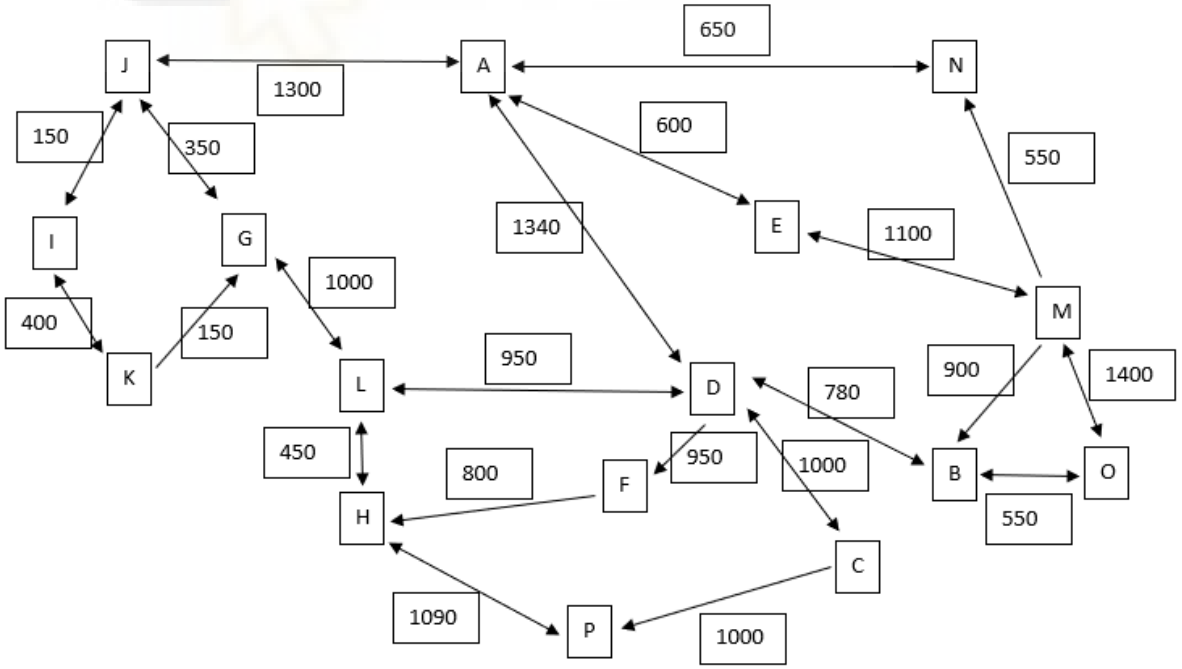


Fig 15: ejemplo del algoritmo de Christofides

1. Lo primero que hacemos es encontrar el árbol mínimo de envergadura máxima, que es el árbol que conecta todos los nodos (ciudades) al menor coste posible. De cada nodo miramos cada arista por su menor coste (teniendo en cuenta que todos los nodos tienen que estar conectados). Por ejemplo, del nodo A, la de menor coste se elegiría la que va a E, que es 600. Así con todos los nodos.

2. Una vez ya tenemos el árbol mínimo de envergadura máxima. Tenemos que conectar todos los nodos, por ello en algún caso no sucederá que sea el mínimo el que se escoja, como pasa por ejemplo en la K, que en vez de ir a la G, va a la I.

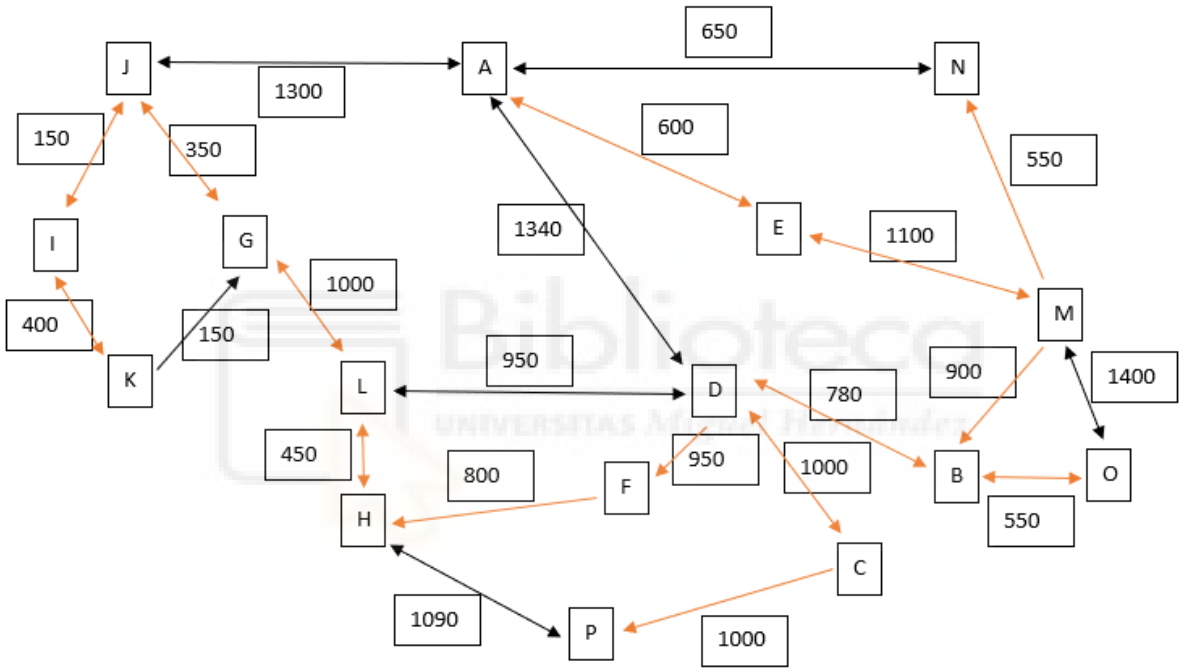


Fig 16: ejemplo algoritmo de Christofides 2

3. Después procedemos a conectar los nodos impares del árbol mínimo de envergadura máxima.

- Nodo A sale 1 ruta
- Nodo N sale 1 ruta
- Nodo M salen 3 rutas
- Nodo O sale 1 ruta
- Nodo B sale 3 rutas
- Nodo D salen 3 rutas
- Nodo P sale 1 ruta
- Nodo K sale 1 ruta

Una vez identificados esos nodos, ahora tenemos que eliminar la imparidad de este problema, esto lo hacemos mediante la conexión de los nodos impares. Nos quedaría de esta forma:

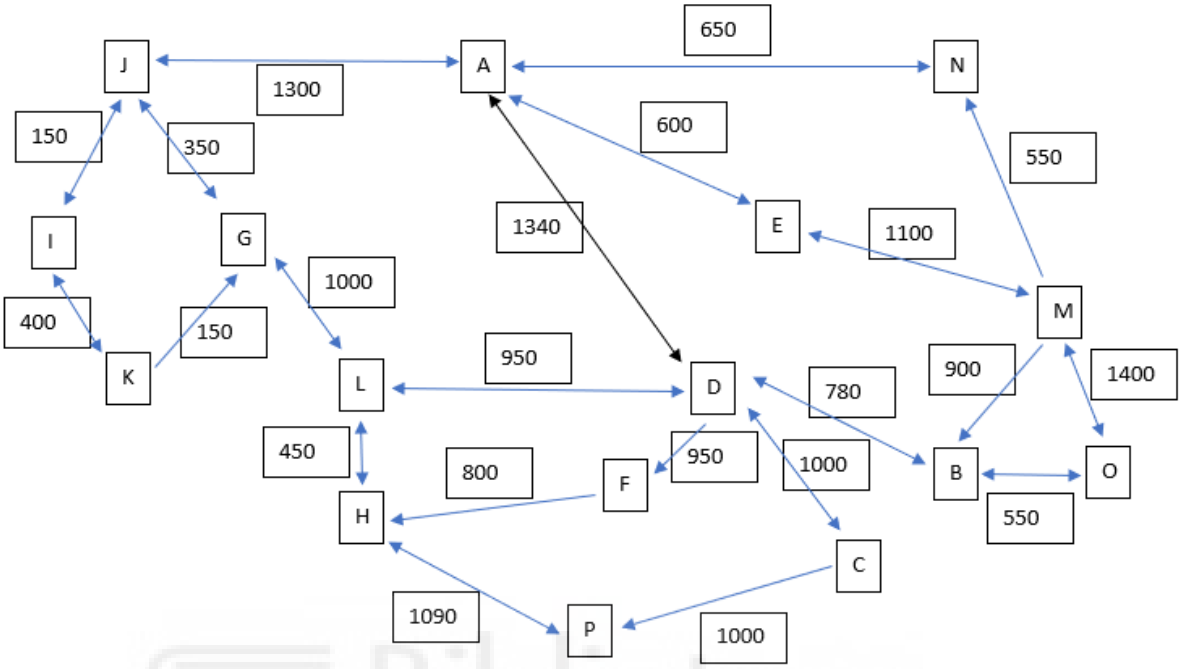


Fig 17: ejemplo de algoritmo de Christofides 3 (recorrido final)

La ruta cercana a la óptima es: A-N-A-E-M-O-B-D-C-P-H-L-D-F-H-L-G-J-I-K-G-J-A.

1.3 Problemas de rutas de vehículos

1.3.1 Introducción

El problema de rutas de vehículos (en inglés Vehicle Routing Problem VRP), es un problema de optimización combinatoria. Surgió como extensión del problema del agente viajero, en ciertos casos cuando la capacidad de los vehículos que realizan la ruta es limitada, y se necesite realizar varias rutas.

Como hemos comentado anteriormente se encuentra dentro de la denominada clase de problemas NP-completos, ya que se requiere mucho esfuerzo a la hora de encontrar una solución óptima, y este esfuerzo aumenta de manera exponencial con el tamaño del problema.

El objetivo del problema de rutas de vehículos es diseñar rutas para una flota de vehículos determinada, que a parte de satisfacer la demanda de los clientes que están dispersos geográficamente, se tiene que encontrar la ruta óptima (obtener la ruta que genere un menor coste a la empresa, ya sea dinero, tiempo, etc..), donde las partidas y llegadas de los vehículos son siempre el mismo lugar (origen). La flota de los vehículos puede ser homogénea o heterogénea.

Como en el problema del agente viajero tenemos las aristas y los vértices.

La pregunta que nos haríamos con este problema sería la siguiente: ¿Cuál es el conjunto óptimo de rutas para una flota de vehículos que debe satisfacer las demandas de un conjunto dado de clientes?.

El VRP tiene muchas aplicaciones, y la utilización de programas de optimización puede proporcionar grandes ahorros.

1.3.2 Origen del VRP

La primera definición del problema de rutas de vehículos, aparece en un artículo George Dantzig y John Ramser en 1959, en donde plantea una aproximación algorítmica y fue aplicado para entregas en gasolina, cuyo objetivo era asignar una ruta para que los camiones llegasen a las estaciones de servicio y la distancia de todos esos camiones fuese la mínima. En 1964, Clarke y Wright mejoraron la aproximación de Dantzig y Ramser utilizando una aproximación "*greedy*" conocido como algoritmo de ahorros.

1.3.3 Variantes del VRP

El problema de rutas de vehículos atiende a diferentes necesidades, esto hizo que surgieran diferentes variantes del problema, estas son algunas de las variantes:

●Problemas de enrutamiento de vehículos con recogida y entrega (VRPPD, por sus siglas en inglés). En esta variante se estudia cuando un número de productos se necesita mover de un lugar (recogida) a otro (entrega), teniendo en cuenta que se permite enviar y recibir cierta cantidad de productos entre ellos. El objetivo es encontrar la ruta óptima para que una flota de vehículos parte de la recogida y entregue los productos, considerando que la carga del vehículo al llegar a una parada será la carga inicial menos las demandas ya repartidas más las demandas recogidas.

●Problema de Enrutamiento del Vehículo con Ventanas de Tiempo (VRPTW, por sus siglas en inglés). Se añadiría una restricción, en la que cada cliente tiene que ser atendido dentro de un cierto horario, a esto se le denomina ventanas de tiempo.

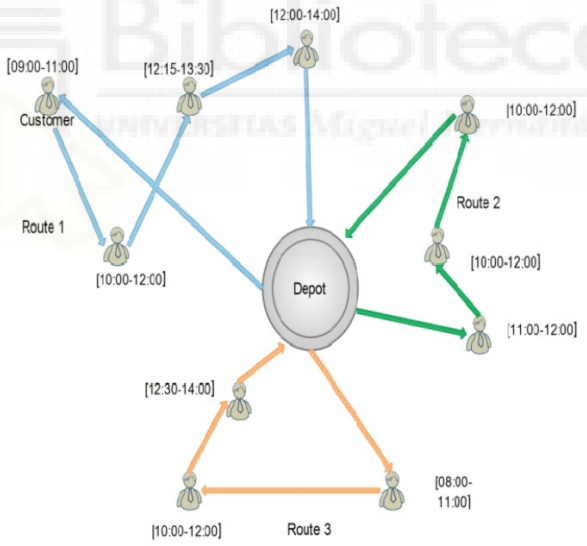


Fig 18: ejemplo VRPTW

●Problema de Enrutamiento de Vehículo Periódico (PVRP, por sus siglas en inglés). En este caso los pedidos pueden ser entregados sólo en ciertos días.

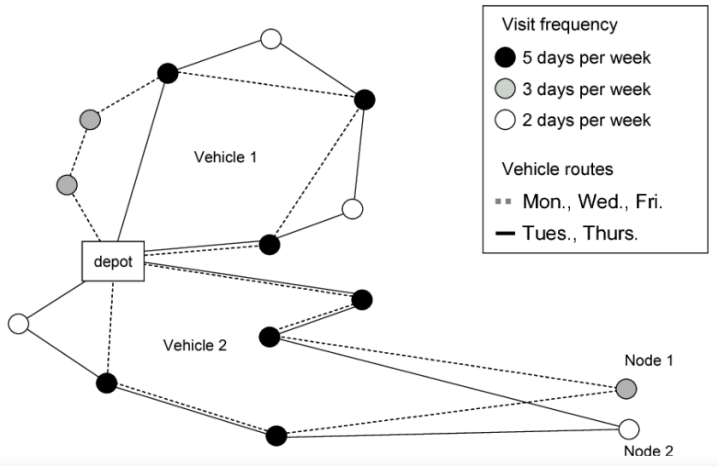


Fig 19: ejemplo del PVRP

●Problema de Enrutamiento de Vehículo con Múltiples Depósitos (MDVRP, por sus siglas en inglés). Este problema tiene varios depósitos con una flota de vehículos por cada depósito, los cuales deben atender la demanda de todos los clientes.

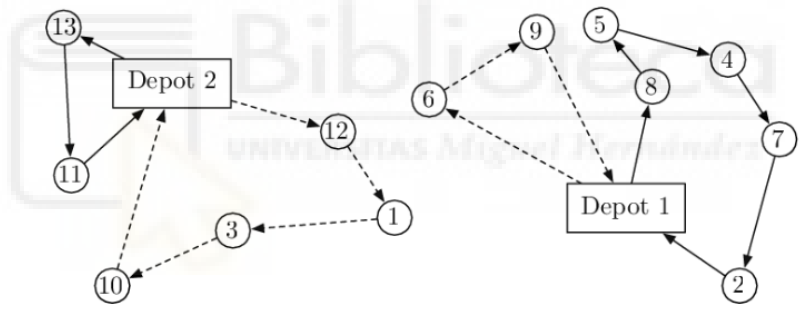


Fig 20: ejemplo MDVRP

●Problema de Enrutamiento del Vehículo con Capacidad (CVRP, por sus siglas en inglés). Todos los vehículos de la flota tienen capacidad determinada todos los vehículos por igual. La restricción que añadimos en este caso es la de capacidad, es decir, la suma de las demandas no puede exceder la capacidad del vehículo.

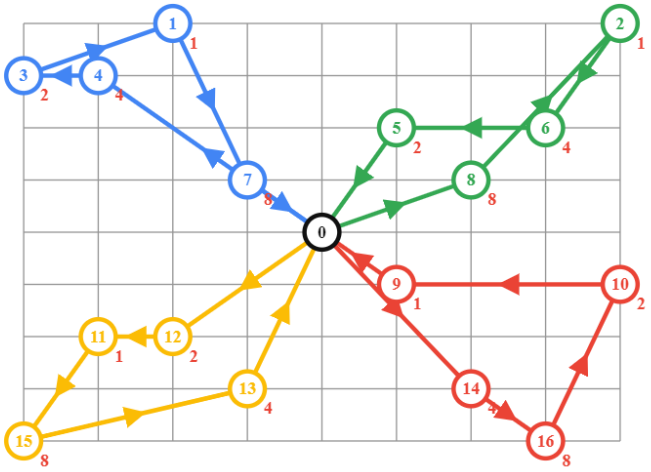


Fig 21: ejemplo de CVRP

1.3.4 Modelización y formulación del VRP

Como ocurre con el problema del agente viajero, existen múltiples formas de plantear el problema de rutas de vehículos. En este modelo K es el número de vehículos, C es la capacidad de los vehículos y d_j son las demandas.La función es minimizar el coste, ya sea coste económico, tiempo, distancia, etc.

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

Las restricciones son las siguientes:

sujeto a

$$\sum_{i \in V} x_{ij} = 1 \quad \forall j \in V \setminus \{0\}$$

$$\sum_{j \in V} x_{ij} = 1 \quad \forall i \in V \setminus \{0\}$$

$$\sum_{i \in V \setminus \{0\}} x_{i0} = K$$

$$\sum_{j \in V \setminus \{0\}} x_{0j} = K$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S), \quad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in V$$

El significado de cada restricción, se explica a continuación:

- Estas restricciones establecen que exactamente un arco entra y exactamente uno sale de cada vértice asociado con un cliente, respectivamente.

$$\sum_{i \in V} x_{ij} = 1 \quad \forall j \in V \setminus \{0\}$$

$$\sum_{j \in V} x_{ij} = 1 \quad \forall i \in V \setminus \{0\}$$

- El número de vehículos que salen del depósito es el mismo que el número que entra.

$$\sum_{i \in V \setminus \{0\}} x_{i0} = K$$

$$\sum_{j \in V \setminus \{0\}} x_{0j} = K$$

- Imponen que las rutas deben estar conectadas y que la demanda en cada ruta no debe exceder la capacidad del vehículo

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S), \ \ \forall S \subseteq V \setminus \{0\}, S \neq \emptyset$$

- Restricción de integralidad

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in V$$

1.3.5 Algunos métodos de resolución del VRP

Las técnicas que más se utilizan a la hora de resolver estos problemas son las heurísticas y meta heurísticos, ya que los métodos exactos no garantizan encontrar la solución óptima en un tiempo razonable hablando computacionalmente cuando el número de la instancia es grande.

Las técnicas de solución se clasifican de este modo:

- Métodos exactos.
- Heurísticos.
- Metaheurísticos

1. Métodos Exactos

Entre los métodos exactos se destacan los algoritmos de ramificación y acotamiento, muchos de ellos se basan en la relación del problema del agente viajero y VRP, la filosofía de este problema se explicó anteriormente.También tenemos el método simplex que consiste en ir mejorando la solución en cada paso, consta en ir del vértice de un poliedro a un vértice vecino de manera que disminuya o aumente la función objetivo. Dado que el número de vértices que presenta un poliedro solución es finito, en la medida en que se pueda satisfacer el conjunto de restricciones, siempre se hallará como mínimo una solución óptima.

2. Métodos Heurísticos

Estos métodos, son técnicas que examinan el espacio de búsqueda de una forma limitada, generando soluciones aceptables, normalmente en tiempos cortos de ejecución.

Algunos de los métodos heurísticos que se utilizan para resolverlo son los siguientes:

- El Algoritmo de Ahorros: este algoritmo de ahorros de Clark & Wright, es una de las técnicas más utilizadas para resolver el problema de VRP a través de heurísticas, consiste en el principio de combinar una solución de dos rutas diferentes para formar una nueva ruta donde se pueda disminuir los ahorros.

- Método de Inserción: consiste como su propio nombre indica insertar, los algoritmos empiezan realizando una selección con n nodos, después se añade una

o varias restricciones, que esto serían los j, para que la distancia del ciclo no aumente.

●Heurístico de dos fases: este tipo de heurísticos permiten componer soluciones viables que probablemente no resulten soluciones óptimas, se divide el problema en dos fases. La primera fase es para agrupar los nodos y la segunda fase es para asignar la ruta de vehículos. Dentro de esta se reconocen dos métodos, que son los siguientes:

- ❖ Método De Rutear Primero Y Asignar Después (Routing First Clustering Second). Se desarrolla en dos fases. En la primera se usa un algoritmo que genere una ruta X donde se incluyen todos los nodos como si fueran a ser visitados por un solo vehículo sin tener en cuenta las restricciones de capacidad del problema. Esta ruta resulta ser la solución óptima que en la segunda fase va a ser agrupada (clustering) de forma que la capacidad del vehículos sea optimizada y no se exceda.

- ❖ Método De Asignar Primero Y Rutear Después (Clustering First, Routing Second). Este método se desarrolla en dos fases:

    1. La primera fase consiste en crear grupos de clientes, llamados clusters (estos se mantienen constantes).
    2. La segunda fase consiste en construir la ruta para cada uno de los clientes (crear algoritmo permita solucionar el problema de forma exacta o de la manera más aproximada posible, ajustándose a las características).

       Las más utilizadas por este método son:

        - ➢ Heurística de Barrido o Sweep
        - ➢ Heurística de Asignación Generalizada de Fisher y Jaikumar
        - ➢ Heurística de Localización de Bramel y Simchi-Levi

3. Métodos meta heurísticos

Estos métodos permiten obtener soluciones mejores que las heurísticas pero con un tiempo mayor de ejecución que los métodos heurísticos, pero aún menor que los métodos exactos.

Algunos de los métodos son los siguientes:

●Búsqueda Tabú: es un algoritmo que evalúa secuencias de soluciones mejorada, permitiendo navegar en un campo de posibles soluciones más amplio sin tener en cuenta las que ya se han encontrado (evitar ciclos) es lo que se denomina tabú. Con este método no se obtienen muy buenos resultados.

●Optimización de colonias de hormigas: este método está inspirado en las colonias reales de hormigas. En la búsqueda de comida, las hormigas dejan un rastro de feromonas, la cantidad de estas feromonas depende de la calidad de la comida y su longitud. La finalidad es encontrar el camino o caminos más próximos al hormiguero, y estos serán los más frecuentados.

Esto hizo que se creara este algoritmo creando hormigas artificiales que exploren el conjunto de soluciones, y variando del valor que tome la función objetivo, se señale valores mediante una función que imite a las feromonas.

●Algoritmo de GRASP (sus siglas en inglés Greedy Randomized Adaptive Search Procedure), es conocido como “Procedimiento de Búsqueda Voraz Aleatorio y Adaptativo”. La realización de este algoritmo se divide en dos fases, por un lado tenemos la fase constructiva que emplea heurísticas constructivas para obtener una solución inicial, y por otro lado está la fase de mejoramiento que es mejorar la anterior con un algoritmo de búsqueda local.

●Algoritmo genético: este algoritmo se basa en la teoría de la evolución de especies de Darwin, donde se originan soluciones a partir de otras guardando algunas características de estas dependiendo del grado de mutación que se quiere lograr en cada iteración, las mutación se realizan de manera aleatoria, lo que resuelve la problemática de óptimos locales presentes en algunas soluciones.

1.4 Un problema de rutas para la recogida de residuos con ventanas de tiempo: un caso de estudio médico

Relación del trabajo con una revista de impacto publicada: The Waste Collection Vehicle Routing Problem with Time Windows: A Medical Waste Collection Case Study.

El estudio nombrado anteriormente defiende que los residuos deben ser recolectados/transportados y eliminados en el menor tiempo posible y de la mejor manera posible. Esto requiere unos costes bastante elevados, y se requiere de una política y gestión adecuadas para conseguir minimizar estos costes.

Existen otros tipos de problemas, y esto hizo que surgiera el problema de enrutamiento de la contaminación (WCPRP).

La mayoría de investigadores tratan de encontrar soluciones para reducir los costes de transporte, pero muy pocos incluyen las emisiones de dióxido de carbono/ventanas de tiempo etc, ya que es bastante complicado introducirlo como variables.

Para la realización de problemas de logística se necesita superar problemas reales como la congestión del tráfico, llegadas tardías, tiempo de carga, reducción de emisiones etc.Se quiere maximizar el beneficio reduciendo la contaminación.

El proyecto actual tiene como objetivo encontrar una solución a este problema, y poder encontrar soluciones reduciendo la contaminación y los costes.

En la última década varios investigadores se han interesado en encontrar estrategias para generar beneficios para gobiernos y empresas, no solo beneficios económicos sino también con contribuciones a la sociedad y medio ambiente.

Se sabe que los problemas de enrutamiento tienen unos niveles altos de CO2 debido al peso y kilómetros. El estudio de contaminación de 2018 the WMO GREENHOUSE GAS, afirmó que las variaciones de los contaminantes más importantes han aumentado de la siguiente forma: dióxido de carbono 81,6%, metano 10,2% y nitroso 5,6%. Descubrieron que la mayoría de emisiones del efecto invernadero provienen de las actividades del transporte. Por ello es importante reducir la contaminación ambiental y generar ahorros para la empresa.

El modelo que comenzó en esta área, es el problema del agente viajero, introducido por Dantzig y Ramser. Posteriormente comenzó otro estudio ya que se iban añadiendo restricciones aquí empezó el problema de generación de rutas.En el inicio de este segundo problema Clarke y Wright implementaron un método de

optimización lineal. Después estos problemas se utilizaron para analizar escenarios de la vida real, cada vez con una mayor complejidad.

Comenzó a ganar popularidad el algoritmo genético, enjambre de partículas, búsqueda del vecindario variable, optimización de colonias de abejas, entre otros.

Las recientes contribuciones han realizado mezcla entre heurística y metaheurística, de esta forma se genera una mejor forma los algoritmos de búsqueda local.

Los problemas de la base de recogida de residuos, pertenecen a la logística inversa. Intentaron resolverlo donde el proceso de captación se basó únicamente en los nodos de generación, mejorando al agregar estaciones de transferencia, generando así rutas más fáciles con menos cruces para reducir distancia y optimizar la capacidad del vehículo.

Estos son casos que resaltan en el estudio:

Alshraideh y Qdais recalcaron la importancia de optimizar rutas,que contribuye a una menor producción de emisiones de efecto invernadero cuando se reduce la distancia recorrida. También intentaron minimizar la totalidad del viaje, reduciendo las emisiones de CO2 y satisfaciendo las demandas de los clientes en términos de tiempo.

Shih y Chang resolvieron el problema de la recolección de desechos,basado en un método de 2 fases.

S. Úbeda en su estudio para una empresa de distribución de alimentos, abordan problemas mejorando los horarios de las rutas de entrega y reduciendo el número de viajes (consiguió reducir el CO2 acortando la distancia), esto utilizando el método heurístico del vecino más cercano.

Molina en el VRP que utilizaba funciones multiobjetivo para reducir, costes generales, emisiones de CO2 y NOX. En el ámbito de la formulación, Molina consideró una flota heterogénea, restricciones dependientes del tiempo y dos contaminantes principales a minimizar.

Sawik introdujo una programación de enteros mixtos para un minorista español, incluía objetivos para reducir el ruido, la contaminación y consumo de combustible.

En Xiao demostró que la suposición clásica que dice que a menos kilómetros recorridos menos contaminación se produce no siempre es correcta. Similar enfoque de Suzuki, descomponiendo la función objetivo basada en el consumo de combustible de carga y descarga de combustible por vehículo.

En Gaoyuan , su investigación enfatiza el intercambio entre llegar a satisfacer al cliente y la reducción de CO2. Preguntas importantes para resolver el problema de ruteo de vehículos.

Otros enfoques, como en Jabali analizó la limitación de la velocidad del vehículo como parte del proceso de optimización.

Niu implementó un modelo de optimización para el problema de enrutamiento de vehículos abiertos (OVRP) incluyendo un algoritmo de optimización de velocidad como estrategia para encontrar la velocidad óptima por ruta para limitar el consumo de combustible y los costos salariales.

El modelo matemático para el WCHFPRPTW tiene como objetivo minimizar las emisiones de CO2 transportadas en una ruta. Sea G = (V, A) un grafo con un conjunto de nodos N = {0, 1, 2. . . , n, n + 1}.

La función objetivo tiene como prioridad minimizar kg de CO2 en función del consumo de combustible debido a la distancia y peso transportados en una ruta propuesta por Xiao .Esta ecuación tiene flota heterogénea.

Algoritmo genético

Para resolver el WCHFPRPTW se implementó un Algoritmo Genético (GA). La AG, se basa en las leyes de selección natural y supervivencia de la prueba. El algoritmo básico consta de tres operadores genéticos (selección, cruce y mutación) que se utilizan para transformar una población generada aleatoriamente en una óptima o casi óptima.

En el inicio el algoritmo comienza con una población inicial que son posibles soluciones que se representan como una cadena de bits con longitud fija. Una ruta es una secuencia que comienza en el 0 (depósito) hasta el siguiente 0. Por ejemplo:

0, 2, 3, 5, 8, 0, 1, 7, 0, 6, 0, 4, 9, 0

Esto quiere decir que el primer camión va desde el depósito al cliente 2, y posteriormente al resto de clientes hasta llegar al cliente 7. El segundo camión va desde el depósito hasta el cliente 1, y sucesivamente al resto de clientes.

Aunque, al principio, las soluciones de GA no siempre cumplen con las ventanas de tiempo, el algoritmo lo supera usando una penalización en la función objetivo.

El siguiente paso es el cálculo de la función objetivo, en este caso viene dada por los kilogramos de CO2, se añaden las siguientes constantes:

-Penalización por cada unidad de tiempo si el camión llega después de la última ventana de tiempo.

-Penalización por cada kilogramo de sobrecarga.

El algoritmo tenderá a encontrar las rutas que generan las menores emisiones en cada iteración, hasta que no se pueda realizar ninguna mejora.

La parte del cruce y mutación, dónde después de que cada par de padres sea apareado, el proceso de cruzamiento combina los genes de cada padre para crear un nuevo hijo. Se necesita un cruce de dos puntos,donde los genes entre dos puntos se combinan entre padres.

Una vez se genera la nueva descendencia, algunos genes pueden sufrir mutaciones, esto significa que alguno de los bits se pueden intercambiar aleatoriamente en función de una probabilidad preestablecida por cada investigador.

Algunos investigadores añaden la tasa de elitismo, que es una probabilidad de dejar intactos algunos de los genes más fuertes de la población, y no se ven afectados por ninguna mutación.

En el operador de dos opciones lo utilizaron varios investigadores. Se aplicó solo a arcos de la misma ruta para evitar que se supere la capacidad de los vehículos, y también el algoritmo evalúa que el intercambio 2opt no incumple las restricciones de ventanas de tiempo para hacer el movimiento. Finalmente, el algoritmo termina si el número de generaciones alcanza la generación máxima número (parámetro).

En el caso de estudio real basado en M.Mera, la empresa de interés recolecta desechos hospitalarios de un conjunto de 177 clientes, ubicados en Guayaquil, Ecuador. Importante para el proyecto actual es que no hay una técnica de enrutamiento de vehículos, lo que significa que algunos camiones inevitablemente regresarán con recogidas canceladas o clientes no atendidos por culpa de la empresa.

Route	Clients	Km Travelled	Travelled Time	Fuel	CO_2	Waiting Time	Veh.Utilization
1	31	283	14	37.66	100.56	31%	46%
2	28	195	10	34.20	91.31	51%	91%
3	25	217	13	38.73	103.42	53%	37%
4	35	166	15	21.71	57.97	32%	25%
5	19	146	12	24.65	65.81	71%	10%
6	23	146	10	24.60	65.69	35%	10%
7	8	177	14	33.20	88.65	65%	9%
8	8	123	5	20.49	54.70	33%	7%
Total	177.0	1453	93	235.25	628.11	-	-

Table 3: Company's initial situation

En la tabla anterior se muestra el análisis de la situación inicial de la empresa que se realizó considerando variables específicas. La tabla compuesta por 8

columnas que son el número de ruta, el número de clientes atendidos en cada ruta, el kilómetro total recorrido por ruta, el tiempo total recorrido (si se excede de 8 horas, hora extra será 1.5 dólares más cara, aumentando los costes totales).

Las columnas 5, 6 y 7 representan el consumo de combustible y las emisiones de CO2, tiempo de espera, representa las llegadas anticipadas (calculadas por la diferencia entre el tiempo real de llegada, resultado del tiempo de salida más el tiempo de tránsito tiempo desde el nodo anterior hasta el destino, y el tiempo real de inicio del servicio permitido en ese nodo). Y en la última columna de utilización del vehículo, representa el porcentaje del vehículo utilizado, considerando su capacidad total, al finalizar el recorrido.

La empresa posee una flota heterogénea detallada en la Tabla 4 que consta de tres camiones de 3000 kg, cinco de 5.000 kg y uno de 10.000 kg.

Vehicle Kg	ρ_{0k}	ρ_k^*
3000	12.9	14.1
5000	16.6	19
10000	18.7	21.6

Table 4: Vehicle Parameters

La propuesta de optimización es buscar reducir la producción de CO2 en peso/distancia/kilómetro, lo que también sirve para mejorar la satisfacción del cliente mientras se mitiga la contaminación ambiental. Además, se consideró la introducción de una segunda función objetivo en el apartado computacional para evaluar el impacto de penalizar el número de vehículos utilizados. En esta función se incluyó un factor constante y tomó el valor de 10 para el caso de estudio (representando el costo de uso de cada vehículo, por lo tanto, evalúe los beneficios de penalizar el número de vehículos utilizados en un viaje completo).

El algoritmo logró resultados satisfactorios, cumpliendo con todas las restricciones y parámetros involucrados. Para garantizar resultados de calidad, se realizaron 30 pruebas por combinación. Las combinaciones se basaron en las dos funciones objetivo, 10 y 12 horas de trabajo permitidas por vehículo, 100 y 200 generaciones, que corresponden a las iteraciones que se permite al algoritmo para generar posibles resultados. Se utilizaron 342 clientes y, una relajación de las ventanas de tiempo con brechas mínimas de 5 horas. Además, se penaliza el tiempo, permitiendo 10 y 12 horas de trabajo y, se consideró un parámetro extra que estipula el rango de ejecución de 100 y 200 generaciones, para asegurar que cuanto

mayor sea el número de pruebas realizadas por el programa, más precisas serán las respuestas. Los resultados son los siguientes:

Time Window	Objective Function	Generations	Working Hours	Distance	Vehicles Used	CO_2	Waiting time	Vehicle Utilization
Relaxed	CO2 and Vehicle Use	100	10	560.83	6.5	240.44	27%	33%
Original	CO2 and Vehicle Use	100	10	601.43	7.0	259.38	31%	31%
Relaxed	CO2	100	10	568.27	6.9	241.20	33%	31%
Original	CO2	100	10	607.1	7.0	260.67	30%	30%
Relaxed	CO2 and Vehicle Use	200	10	520.07	5.9	220.86	23%	37%
Original	CO2 and Vehicle Use	200	10	570.17	6.8	245.28	30%	31%
Relaxed	CO2	200	10	549.1	6.7	231.93	30%	33%
Original	CO2	200	10	576.53	7.0	247.63	32%	30%
Relaxed	CO2 and Vehicle Use	100	12	534.43	5.3	221.80	28%	43%
Original	CO2 and Vehicle Use	100	12	560.67	5.8	238.45	32%	38%
Relaxed	CO2	100	12	542.77	6.4	231.13	39%	34%
Original	CO2	100	12	583.83	6.6	247.93	39%	33%
Relaxed	CO2 and Vehicle Use	200	12	499.63	5.4	212.04	30%	41%
Original	CO2 and Vehicle Use	200	12	534.5	5.5	229.76	30%	39%
Relaxed	CO2	200	12	545.77	6.7	230.47	42%	33%
Original	CO2	200	12	556.37	6.5	236.80	40%	34%

Table 5: Average Results per combination

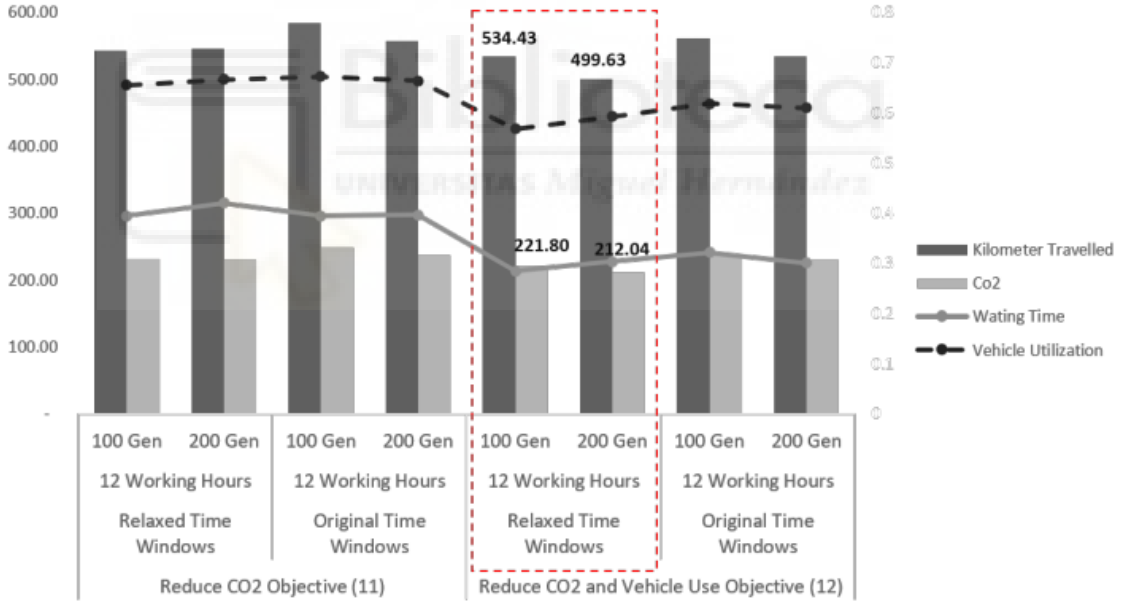


Figure 2: Comparison between the two Objective functions and their performance

De los resultados se pudo observar, que la función objetivo que incluye la sanción adicional por circulación de vehículos genera los mejores resultados tanto con ventanas de tiempo originales del problema y su versión relajada con 5 horas de diferencia.Además, 12 horas de tiempo máximo de viaje y 200 generaciones llevan el algoritmo genético para lograr resultados buenos para el problema.

Los resultados que se muestran en la Tabla 5 concluyeron que, la función objetivo que minimiza el CO2 y el uso del vehículo es mejor que con la función que

solo reduce el CO2. La función objetivo que minimiza el CO2 y el uso de vehículos, considerando ventanas de tiempo relajadas y 200 generaciones; recorre una media de 500 km generando 212 gr de CO2 por Kg. Sin embargo, usando el original, todavía encuentra buenos resultados, aunque, un poco más alto que usando ventanas de tiempo relajadas, los resultados todavía se consideraron buenos alcanzando 534.50 km recorridos y produciendo 229,76 gr CO2, estos resultados optimizan recursos y alcanzan un 100% de satisfacción del cliente. En la Tabla 6, los mejores resultados consideraron una política que relaja las ventanas de tiempo del cliente, 50 cromosomas, 200 generaciones y 12 horas máximas de trabajo, alcanzó los 459 km recorridos 189 gr CO2 por km recorrido, 24% del tiempo de espera y 50% de la utilización del vehículo. A pesar de esto beneficia a la empresa, reduciendo kilómetros recorridos y desperdicio de vehículos.

Route	Clients	Km. Travelled	Fuel	CO_2	Wating Time	Veh. Utilization
V1	0,128,68,37,71,39,48,46,42,96,93,108,147, 146,142,129,127,175,0	53	7.02	18.75	56%	44%
V2	0,2,1,8,4,3,7,9,11,16,17,12,13,23,26,33,34, 41,45,51,54,69,75,74,70,58,59,44,47,49,60, 80,77,84,102,92,101,106,99,133,130,176, 122,113,112,116,119,135,134,141,150,151, 155,159, 0	89	15.07	40.24	11%	34%
V3	0,143,132,87,28, 86,83,79,144,78,117,115, 111,107, 100,97,98,95,109,123,125, 140, 145,170, 177,169, 168,0	112	14.65	39.11	44%	30%
V4	0,5,6,10,20,24,30,32,35,40,38,50,76,65,63, 67,66,61,104,90,110,118,120,121,124,131, 139,154,163,160,114,105,173,174,138,156, 158,164,166,165,0	99	17.19	45.90	18%	58%
V5	0,19,18, 14,15,22,31,27,21,25,29,36,43,53, 52,56,55,64,57,62,72,73,85,82,88,81,89,91, 94,103,126,149,148,136,137,152,153,157, 167,172,171,161,162,0	118	22.74	60.71	8%	31%

Table 7: Best solution for original time windows with the reducing CO_2 and vehicle use function

En la Tabla 7, el mejor resultado usando la función minimizar CO2 y uso del vehículo considerando ventanas de tiempo originales, 50 cromosomas, 200 generaciones y 12 horas máximas de trabajo, alcanzó un viaje completo de 471 kilómetros recorridos generando 204 kg. de CO2, una media del 28% de tiempo de espera y 39% de utilización de vehículos. La solución resultó en 5 vehículos

utilizados para el viaje completo y trabajó 51,37 horas en comparación con la situación original de la empresa de 8 vehículos y 93 horas de trabajo.

Conclusiones a las que llegaron mientras implementa la función objetivo propuesta con las ventanas de tiempo originales:

• Se logró una reducción significativa en la producción de CO2, de 628 kg. a 204,7 kg para el recorrido total.

• El tiempo medio de trabajo se redujo de 12 horas a 10,3.

• Se redujo el tiempo de espera del 34% al 28%.

• La utilización de vehículos se mejoró al 39 % en comparación con el 29 % original.

• El cronograma logró 100% de satisfacción del cliente bajo este escenario sin ninguna modificación en las ventanas de tiempo.

• El viaje completo cubrió 471 km en comparación con 1453 km originalmente.

• La reducción de 8 vehículos a 5 vehículos generó un gran impacto en la estructura de costos, reduciendo el uso de vehículos, y el ahorro por contratar menos conductores. encontró que la capacidad del vehículo no está ocupada eficientemente.

1.5 Realización del algoritmo

Explicación del algoritmo que hemos realizado para buscar la mejor solución a nuestro problema, mostrando algunos resultados que nos dan. El algoritmo está basado en una función (“calculoRutas”) a través de la cual se le pasan dos parámetros: el número de muestras que queremos y, la capacidad tienen nuestros vehículos. En el ejemplo que se mostrará a continuación indica que queremos 3 muestras distintas (se puede poner cualquier número pero, para mostrar los datos y facilitar el entendimiento, se usarán 3 muestras) y, que nuestros vehículos tienen una capacidad máxima de 60

```
calculoRutas(3,60)
```

Hay que saber que la función devuelve una serie de valores en forma de lista que son: las cantidades de paquetes de cada vehículo de todas las muestras (las variables demandaN ([[1]],[[2]] y [[3]]), la matriz ([[4]]), el coste más rentable ([[5]]) y su ruta ([[6]]), los costes de todas las muestras ([[7]]) y, las rutas seguidas por todos los vehículos de todas las muestras ([[8]]), los números indicados después de las variables indican su posición en la lista. Por otro lado, hay que conocer que las variables que se devuelven, excepto la de demandaN (siendo N=1,2,3), tienen 7 columnas con el objetivo de hacer que en caso de que haya traspaso de paquetes inesperados (se explicará posteriormente) quepan siempre en la matriz.

Nuestra empresa de transportes tiene su sucursal en Alicante, que este sería nuestro almacén donde los camiones descansan y recogen la mercancía para repartirla a los respectivos clientes. Al comienzo la empresa tiene 15 clientes a los que tenemos que repartirles cierta mercancía. Tenemos que tener en cuenta que ir de un cliente a otro tiene cierto coste, que los hemos generado de forma aleatoria.

Esto se realizó mediante el programa Rstudio de esta forma:

```
calculoRutas=function(numMuestras,capacidad){
  mas_rentable=10000
  costes=vector()
  n=numMuestras       #Cantidad de muestras que quiero
  coche1_rutas=matrix(nrow = n,ncol = 7)
  coche2_rutas=matrix(nrow = n,ncol = 7)
  coche3_rutas=matrix(nrow = n,ncol = 7)
  rutaMasRentable=matrix(nrow = 3,ncol=7)
  matriz = diag(0,nrow=16)
  diag=1
  fila=1

  #Se crea la matriz con diagonal 0 y el resto de números aleatorios gracias a la función sample
  for (i in 1:16){
    matriz[i,]=sample(1:30,16)
  }
  for (i in 1:16){
    matriz[i,i]=0
  }

  for (i in 1:16){
    for (j in 1:16){
      matriz[i,j]<-matriz[j,i]
    }
  }
```

Hemos generado una matriz simétrica 16x16 números aleatorios con la función sample, siendo la primera fila y columna el almacén, y que la diagonal contenga ceros, para que no cuente como coste estar en el mismo sitio. Y por ejemplo obtendremos esta matriz:

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]	[,13]	[,14]	[,15]	[,16]
[1,]	0	16	26	19	9	20	1	24	14	18	27	5	11	16	16	25
[2,]	16	0	4	6	23	16	18	14	22	1	15	24	26	13	6	18
[3,]	26	4	0	20	17	1	27	22	18	10	21	6	20	5	26	4
[4,]	19	6	20	0	3	27	24	5	2	3	17	29	18	4	3	6
[5,]	9	23	17	3	0	24	4	23	7	26	8	21	1	29	20	16
[6,]	20	16	1	27	24	0	8	10	29	30	2	9	12	12	27	12
[7,]	1	18	27	24	4	8	0	20	28	5	26	28	15	20	8	28
[8,]	24	14	22	5	23	10	20	0	24	13	13	13	13	14	7	5
[9,]	14	22	18	2	7	29	28	24	0	20	9	14	27	30	12	24
[10,]	18	1	10	3	26	30	5	13	20	0	29	30	4	7	23	20
[11,]	27	15	21	17	8	2	26	13	9	29	0	27	3	6	4	10
[12,]	5	24	6	29	21	9	28	13	14	30	27	0	28	15	30	1
[13,]	11	26	20	18	1	12	15	13	27	4	3	28	0	3	21	15
[14,]	16	13	5	4	29	12	20	14	30	7	6	15	3	0	5	13
[15,]	16	6	26	3	20	27	8	7	12	23	4	30	21	5	0	19
[16,]	25	18	4	6	16	12	28	5	24	20	10	1	15	13	19	0

La empresa tiene tres vehículos (en el código se denominan “coche” en los que repartimos la mercancía de los clientes para su posterior reparto. La capacidad que tiene cada camión es la indicada por parámetro, y lo que se hace es repartir 5 números aleatorios (del 1 al 20, sin repeticiones) en cada camión, 15 números en total, que corresponden a los 15 clientes, que posteriormente sabremos a qué clientes corresponden el número de paquetes. La función “conocer_total” permitirá posteriormente conocer si los paquetes que hay en un vehículo superan la capacidad previamente establecida:

```
cap=capacidad    #Capacidad de cada vehículoS
demanda=sample(1:20,15,FALSE)
demanda1=demanda[0:5]
demanda2=demanda[6:10]
demanda3=demanda[11:15]


conocer_total=function(demanda){  #Función para conocer la cantidad que posee cada coche
  total=0
  for (i in 1:length(demanda)){
    total=total+demanda[i]
  }
  return (total)}
```

La cantidad de paquetes por vehículo podría superar la capacidad, es por eso que surge la siguiente pregunta:¿Cómo resolvemos que no supere la capacidad preestablecida de cada camión? Se realizará de la siguiente manera:

```
#Modificar las demandas si superan la capacidad preestablecida y, pasar dichas demandas a otros coches
if(conocer_total(demanda1)>cap){
  mi_numero=100
  for(i in 1:length(demanda1)){
    if (demanda1[i]<mi_numero & conocer_total(demanda1)-demanda1[i]<=cap){
      mi_numero=demanda1[i]
    }
  }
  demanda1=demanda1[! demanda1 %in% c(mi_numero)]
  demanda2=c(demanda2,mi_numero)
}

if(conocer_total(demanda2)>cap){
  mi_numero=100
  for(i in 1:length(demanda2)){
    if (demanda2[i]<mi_numero & conocer_total(demanda2)-demanda2[i]<=cap){
      mi_numero=demanda2[i]
    }
  }
  demanda2=demanda2[! demanda2 %in% c(mi_numero)]
  demanda3=c(demanda3,mi_numero)
}

if (conocer_total(demanda1)-conocer_total(demanda2)>0){
  if(conocer_total(demanda3)>cap){
    mi_numero=100
    for(i in 1:5){
      if (demanda3[i]<mi_numero & conocer_total(demanda3)-demanda3[i]<=cap){
        mi_numero=demanda3[i]
      }
      }
      demanda3=demanda3[! demanda3 %in% c(mi_numero)]
      demanda2=c(demanda2,mi_numero)
    }
  }
  else{
    if(conocer_total(demanda3)>cap){
      mi_numero=100
      for(i in 1:length(demanda3)){
        if (demanda3[i]<mi_numero & conocer_total(demanda3)-demanda3[i]<=cap){
          mi_numero=demanda3[i]
        }
      }
      demanda3=demanda3[! demanda3 %in% c(mi_numero)]
      demanda1=c(demanda1,mi_numero)
    }
  }
  #Fin modificación demandas
```

En este fragmento de código se hace una control para comprobar que ninguno de los vehículos supere la capacidad establecida. En caso de que en el vehículo uno o dos se supere dicha capacidad, se pasará el número más bajo necesario para que no se supere la capacidad. En el ejemplo que se observa a continuación, se observa como de la demanda2 solo tiene 4 números, eso es porque se ha pasado del número 13 a la siguiente demanda (demanda3).

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    7    6    5   13    2   16    0
[2,]   15    8    4    9    0    0    0
[3,]   14   12   10    3   11    0    0
```

En el caso en que fuese la demanda 3 la que necesita reducir su cantidad de paquetes, dicha cantidad se pasará a la demanda con menor cantidad.

Ahora lo que hacemos es que se nos reparta de forma aleatoria dentro de esos furgones a que cliente pertenece dicha cantidad de paquetes para posteriormente seguir la ruta en la matriz mostrada al principio del apartado.

```
#Se crean tablas de las rutas que sigue cada coche en la matriz
for(i in 1:7){
  if( is.na(coche1[i])){
    coche1_rutas[e,i]=0
  }
  else{
    coche1_rutas[e,i]=coche1[i]
  }
}
for(i in 1:7){
  if( is.na(coche2[i])){
    coche2_rutas[e,i]=0
  }
  else{
    coche2_rutas[e,i]=coche2[i]
  }
}
for(i in 1:7){
  if( is.na(coche3[i])){
    coche3_rutas[e,i]=0
  }
  else{
    coche3_rutas[e,i]=coche3[i]
  }
}
#Fin de crear tablas de las rutas
```

Estos serían los resultados (para observar los datos y entenderlos hay que fijarse en las filas, por ejemplo, la fila 1 de las tres rutas corresponden a la misma muestra):

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   15    6    2   11    3    7    0
[2,]    7    6    5   13    2   16    0
[3,]   15    3   11   12    4   14    0

[[9]]
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   12    8   14   16    0    0    0
[2,]   15    8    4    9    0    0    0
[3,]    8    9    5   16    0    0    0

[[10]]
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   13    9    4   10    5    0    0
[2,]   14   12   10    3   11    0    0
[3,]    7    2   10    6   13    0    0
```

Estos resultados muestran qué trayectoria seguirá el vehículo por la matriz, Ahora lo que nos interesa es saber el coste que genera cada vehículo. Para ello se ha usado la función “calcular_coste”:

```
    #Se calcula el coste total, es decir, el coste sumando todos los costes de todos los coches
    coste_total=calcular_coste(coche1)+calcular_coste(coche2)+calcular_coste(coche3) #Coste de los tres coches juntos
    if(coste_total<mas_rentable){    #Se compara el coste con el más rentable y, si es menor, entonces se define como el más rentable
      mas_rentable=coste_total
      for (i in 1:7){
        if( is.na(coche1[i])){
          rutaMasRentable[1,i]=0
        }
        else{
          rutaMasRentable[1,i]=coche1[i]
        }
        if( is.na(coche2[i])){
          rutaMasRentable[2,i]=0
        }
        else{
          rutaMasRentable[2,i]=coche2[i]
        }
        if( is.na(coche3[i])){
          rutaMasRentable[3,i]=0
        }
        else{
          rutaMasRentable[3,i]=coche3[i]
        }
      }
    }
    costes[e]=coste_total #Se añade el coste total a un lista de todos los costes totales del bucle
  }
  return(list(demanda1,demanda2,demanda3,matriz,mas_rentable,rutaMasRentable,costes,coche1_rutas,coche2_rutas,coche3_rutas))
}

calculoRutas(3,60)
```

Una vez calculado el coste de los vehículos por separado, querremos saber cuanto cuesta la movilización de los tres. A la vez que se hace esto, se compara el coste total con costes anteriores para poder determinar que coste es el más rentable y que rutas han seguido dichos coches y, por último, se añaden todos los costes totales a una lista, el código utilizado y los resultados se muestran a continuación:

Por último, queda conocer que la función devuelve una serie de valores que son: las cantidades de paquetes de cada vehículo de todas las muestras, la matriz, el coste más rentable y su ruta, los costes de todas las muestras y, las rutas seguidas por todos los vehículos de todas las muestras.

```
return(list(demanda1,demanda2,demanda3,matriz,mas_rentable,rutaMasRentable,costes,coche1_rutas,coche2_rutas,coche3_rutas))
```

En estos momentos utilizamos la matriz que calculamos inicialmente que equivalen a los costes de ir de un cliente a otro. En este caso sería lo que queremos es conseguir el que menos coste tenga. Para este ejemplo el coste más bajo de estas tres pruebas es:

Obtenemos los siguientes resultados:

```
[[5]]
[1] 200

[[6]]
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    7    6    5   13    2   16    0
[2,]   15    8    4    9    0    0    0
[3,]   14   12   10    3   11    0    0

[[7]]
[1] 236 200 256
```

1.6 BIBLIOGRAFÍA

Eleazar Puente. (2020, 26 mayo). *Método de Ahorros para ruteo de vehículos.* [Vídeo]. YouTube. https://www.youtube.com/watch?v=Aj1XIiGN6WU

Juliana Castañeda Jimenez y Jaime Andrés Cardona Arias (2014). *Implementación del método del ahorrar para resolver el VRP aplicado al diseño de una red logística inversa para la resolección del aceite vehicular usado generado en los puntos de acopio ubicados en Pereira.* [Trabajo final de grado título Ingeniería Industrial, Universidad Tecnológica de Pereira]. https://core.ac.uk/download/pdf/71397901.pdf

M.Sc. José Jesús Mendoza Canovas (2017). Diseño de algoritmos heurísticos y metaheurísticos eficientes para resolver el problema del Agente Viajero. [Tesis doctoral para optar al grado de doctor en matemáticas aplicadas en la Universidad Nacional Autónoma de Nicaragua, Managua]. https://repositorio.unan.edu.ni/8779/1/11129.pdf

Yessica Liceth Velázquez Castiblanco (2015). Análisis de las características y aplicaciones de los sistemas de ruteo de vehículos. [Trabajo final de grado, especialización en gerencia en logística integral de Bogotá, en la Universidad Militar Nueva Granada, facultad de ingeniería]. https://repository.unimilitar.edu.co/bitstream/handle/10654/13308/An%C3%A1lisis%20de%20las%20caracter%C3%ADsticas%20y%20aplicaciones%20de%20los%20sistemas%20de%20ruteo%20de%20veh%C3%ADculos.pdf?sequence=1&isAllowed=y#:~:text=El%20Problema%20de%20Ruteo%20de,satisfacci%C3%B3n%20al%20cliente%2C%20entre%20otros.

Beatriz Moreno Pérez de Vargas (2015). Resolución del problema del viajante de comerci (TSP) y su variante con ventanas de tiempo (TSPTW) usando métodos heurísticos de búsqueda local. [Trabajo del grado en Ingeniería de Organización Industrial, en la Universidad de Valladolid]. https://uvadoc.uva.es/bitstream/handle/10324/13378/TFG-I-252.pdf?sequence=1&isAllowed=y

Carlos Rafael Pérez Cabrera (2016, 13 julio). Problemas de rutas de vehículos algunas de sus variantes más conocidas. [Trabajo de fin de grado matemáticas, estadística e investigación operativa, en la universidad de la Laguna]. https://riull.ull.es/xmlui/bitstream/handle/915/3046/Problemas+de+Rutas+de+Vehiculos.+Algunas+de+sus+variantes+mas+conocidas..pdf;jsessionid=3E6AB37AD730092067B160EA7B209665?sequence=1

Aida Calviño Martínez (2011, 28 junio). Cooperación en los problemas del viajante (TSP) y de rutas de vehículos (VRP). [Proyecto fin de máster interuniversitario en técnicas estadísticas, en la Universidad de Vigo]. http://eio.usc.es/pub/mte/descargas/proyectosfinmaster/proyecto_762.pdf

De Cádiz, U. (s. f.). *TSP: Algoritmos de resolución*. https://knuth.uca.es/moodle/mod/page/view.php?id=3417

Wikipedia contributors. (2023). Vehicle routing problem. *Wikipedia*. https://en.wikipedia.org/wiki/Vehicle_routing_problem

Universitat Politècnica de València - UPV. (2016, 28 enero). *Algoritmo Branch and Bound para secuenciación en configuraciones 1/ri di/Lmax | | UPV* [Vídeo]. YouTube. https://www.youtube.com/watch?v=m1IcXO64-n4

colaboradores de Wikipedia. (2023). Problema de enrutamiento de vehículos. *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/Problema_de_enrutamiento_de_veh%C3%ADculos

Oscar Sotomayor. (2020, 18 julio). *Problema del Agente viajero algoritmo de ramificación y acotamiento B&B* [Vídeo]. YouTube. https://www.youtube.com/watch?v=mQxiuM_w5ZQ

Jorge Romero. (2020, 22 abril). *Algoritmo de Clarke y Wright* [Vídeo]. YouTube. https://www.youtube.com/watch?v=d6WHquQVPVU

Fig. 1. A MDVRP example with 2 depots, 4 routes and 13 customers. (s. f.). ResearchGate. https://www.researchgate.net/figure/A-MDVRP-example-with-2-depots-4-routes-and-13-customers_fig11_225444551

Francis, P. (2006). *The Period Vehicle Routing Problem with Service Choice*. https://www.semanticscholar.org/paper/The-Period-Vehicle-Routing-Problem-with-Service-Francis-Smilowitz/e437698b6e3c34769c92017445661411 00b9be3f/figure/0

Fig. 1. Illustration of the VRPTW. (s. f.). ResearchGate. https://www.researchgate.net/figure/Illustration-of-the-VRPTW_fig1_324726221

Capacity Constraints. (s. f.). *Google Developers*. https://developers.google.com/optimization/routing/cvrp#example

colaboradores de Wikipedia. (2023c). Problema del viajante. *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/Problema_del_viajante#Algoritmos_Exactos

López, B. S. (2020). Problema del agente viajero – TSP. *Ingenieria Industrial Online*. https://www.ingenieriaindustrialonline.com/investigacion-de-operaciones/problema-del-agente-viajero-tsp/

Felipe Gutiérrez Cerda. (2021, 28 abril). *Heuristica de Christófides (Problema del Vendedor Viajero) | Felipe Gutiérrez Cerda* [Vídeo]. YouTube. https://www.youtube.com/watch?v=uPS1I2mO00s

Matemáticas Para Niños Geniales. (2020, 17 octubre). *El Problema del Agente Viajero (TSP)* [Vídeo]. YouTube. https://www.youtube.com/watch?v=LUuSLdBSejQ

Cristian González García. (2020, 18 abril). *Problema del viajero: ciclo Hamiltoniano, algoritmo voraz y Backtracking* [Vídeo]. YouTube. https://www.youtube.com/watch?v=W16E5qzzCxU

Pedro Antonio Teppa Garran. (2021, 16 septiembre). *7. Introducción al problema del agente viajero* [Vídeo]. YouTube. https://www.youtube.com/watch?v=edDGIeqYH4I

1. (s. f.). http://www.fdi.ucm.es/profesor/jjruz/WebProyectos/Carlos%20Salvador%20Vazquez/Capitulo%201.htm

García, A. (2022). Optimización de Rutas de Vehículos. Orígenes y variantes. *oga*. https://www.oga.ai/blog/optimizacion-de-rutas-de-vehiculos-origenes-y-variantes/

Algoritmo de Christofides - Google Search. (s. f.). https://www.google.com/search?q=Algoritmo+de+Christofides&sxsrf=ALiCzsaASjzBKA911TuiNxH_dqJGYNB1Lg:1670773224715&source=lnms&tbm=vid&sa=X&ved=2ahUKEwiJvLyk8_H7AhUT_IUKHdk5Dm8Q_AUoA3oECAIQBQ&cshid=1670773289725077&biw=1366&bih=657&dpr=1#fpstate=ive&vld=cid:e4c951d5,vid:uPS1I2mO00s

El Problema del Agente viajero: un recorrido sobre su historia, sus aplicaciones y problemas relacionados. - PDF Free Download. (s. f.). https://docplayer.es/73167152-El-problema-del-agente-viajero-un-recorrido-sobre-su-historia-sus-aplicaciones-y-problemas-relacionados.html

Conocimiento, V. A. (2020, 3 septiembre). *Hamilton, vándalo matemático e inventor | Pasatiempos OpenMind*. OpenMind. https://www.bbvaopenmind.com/ciencia/matematicas/william-hamilton-vandalo-matematico-invento-juego-mesa/

García, A. (2022). Optimización de Rutas de Vehículos. Orígenes y variantes. *oga*. https://www.oga.ai/blog/optimizacion-de-rutas-de-vehiculos-origenes-y-variantes/

Frederick S. Hillier y Gerald J. Lieberman. 5ª. Eds.(1994). [Introducción a la investigación operativa]

Toth, P. y Vigo, D. eds. (2022) [The vehicle routing problem].