TRABAJO FIN DE MÁSTER
2009 - 2010

PROGRAMA OFICIAL DE POSGRADO EN
INGENIERÍA MECÁNICA Y DE MATERIALES

MÁSTER EN SISTEMAS MECÁNICOS

MODELIZACIÓN DE PROBLEMAS DE CÁLCULO DE RUTAS DE LARGA DISTANCIA CON LLENADO ÓPTIMO DE VEHÍCULOS (VRP + BPP)

Centro Politécnico Superior

Universidad de Zaragoza

EUITIZ

Autor

Dª. Beatriz Royo Agustín

Director

Catedrático D. Emilio Larrodé Pellicer

Codirector

Dr. D. David Escuín Finol

Septiembre 2010

Área de Ingeniería e Infraestructura de los Transportes
Departamento de Ingeniería Mecánica

Anexos

Contenido

Anexo I: NP- Completo

1) Introducción

Para hablar de los problemas NP-Completos se tiene que entender en primer lugar lo que se conoce como la *Teoría de la Computabilidad*. Esta disciplina es una de las cuatro partes que constituyen la lógica matemática: teoría de conjuntos, teoría de modelos, teoría de la demostración y la teoría de la computabilidad y se ocupa del estudio y clasificación de las relaciones y aplicaciones computables. Además, la teoría de la computabilidad, junto con la teoría de autómatas, lenguajes y máquinas, es el fundamento de la informática teórica y esta, a su vez, de la industria de los ordenadores.

La importancia de esta teoría radica en que permite dividir los problemas en decidibles y no decidibles. Lo que es muy útil teniendo en cuenta que el objetivo de este trabajo es resolver un problema que se corresponde con la definición de problema indecidible. De modo que se identifica con la combinación de dos problemas típicos de estudio dentro de esta disciplina. Uno de ellos es el problema TSP o problema del viajante y el segundo es el problema de la mochila.

2) Teoría de la computabilidad

La teoría de la computabilidad es una de las cuatro ramas de la lógica matemática, la cual engloba: la teoría de conjuntos, la teoría de la demostración, la teoría de modelos y la teoría de la computabilidad. Esta disciplina también es una de las partes fundamentales de la informática teórica y a su vez de la industria de los ordenadores, junto a la teoría de autómatas, lenguajes y máquinas.

En esta rama de la lógica matemática se estudian los problemas decidibles y no decidibles. Si se supone el problema del máximo común divisor mediante el algoritmo de Euclides o la determinación de dos números primos mediante la criba de Eratóstenes, se puede concluir que existen algoritmos o procedimientos mecánicos que permiten obtener la solución del problema en cuestión. De forma que, hasta principios del siglo XX, se daba por hecho que existían algoritmos capaces de resolver cualquier tipo de problema, la única dificultad era encontrarlos. Por tanto, si lo que se pretende es determinar un algoritmo, no es necesario definir la clase de todos los algoritmos, esto sólo es necesario si se pretende demostrar que algún problema no es algorítmicamente soluble. Lo que se traduce como que para dicho problema no hay ningún algoritmo que lo resuelva.

Posiblemente Tietze, en 1908, fuera el primero en afirmar la no existencia de un algoritmo. En su caso se refirió a los grupos de representación finita: "*...la cuestión de cuándo dos grupos son isomorfos no es soluble en general*".

Pero hasta 1928, momento en el que por un lado Hilbert y Ackermann en su libro de lógica, con el planteamiento de la decibilidad de la lógica de predicados y por otro lado a partir de 1930, con el asunto de la solubilidad de todo problema matemático con Gödel, Church y Turing, no se había planteado la formalización del concepto informal de función matemáticamente computable. Estas formalizaciones junto con otras realizadas por Kleene, Post y Markoff, dieron lugar a la hipótesis conocida como *Hipótesis de Church-Turing-Post-Kleene,* que afirma la coincidencia entre el concepto informal de función parcial mecánica o algorítmicamente computable y el concepto formal, matemático, de aplicación recursiva.

3) Tesis de Church- Turing

Al hablar de funciones, este documento se refiere a valores naturales (es decir, $f : N^n \rightarrow N$). Una función es *computable* si existe algún método efectivo, es decir, un humano con papel, lápiz y tiempo es capaz de resolverla.

La tesis de Church se basó en conjeturar que cualquier función computable podía escribirse en términos de ciertos operadores matemáticos: el cero, el sucesor de un número, la composición, la recursión primitiva y la minimización. Estas funciones se denominaron *funciones recursivas generales.*

Simultáneamente, Turing lanzó otra hipótesis. Está predice que cualquier función computable podía calcularse con unas maquinas muy simples conocidas como *máquinas de Turing.*

Finalmente, el poder expresivo de las funciones recursivas de Church y la máquina de Turing resultaron ser lo mismo. Por tanto, y tras conocerse este hecho se probó que cualquier función recursiva de Church podía calcularse con una máquina de Turing y viceversa. A estos resultados se les bautizó con *Tesis de Church-Turing.*

El hecho de que cualquier método efectivo pueda ser representado por una función recursiva general o máquina de Turing impone un límite en la computabilidad, puesto que existen funciones que no pueden ser calculadas con estas expresiones o con la máquina de Turing. Estas funciones, según la Tesis de Church-Turing, no deberían ser calculadas con ningún método existente. De hecho, en la actualidad no se conoce ningún método para la resolución de las funciones no computables.

4) Decibilidad

Según la Tesis de Church- Turing, un problema es decidible si y sólo si es computable. Es decir, si existe una maquina de Turing o expresión recursiva capaz de resolverlo. La cuestión de si, para una clase de problemas, existe un algoritmo o procedimiento efectivo que los resuelva se le denomina *problema de decisión.* Se dice que un problema es indecible si se puede demostrar que no existe ningún algoritmo que pueda responder a todas las preguntas que el problema pueda plantear. Ejemplos:

- ¿Hay enteros tales que satisfagan la ecuación 3x + 6y = 151? Este problema tiene una respuesta negativa, NO. Por tanto, se trata de un problema de decisión.
- ¿Hay enteros (x,y) tales que se cumple la ecuación ax+by = c? La respuesta en este caso sería SI, tratándose de nuevo de un problema decisional.
- ¿Para cada asignación de valores (a, b, c), se tiene un problema distinto? SI, si el máximo común divisor de a y b divide a c, para ello se tiene el algoritmo de Euclides para resolverlo.

A los problemas decisionales con respuesta negativa se les conoce como problemas *indecibles* o *no decidibles.* Sin embargo, existen problemas elementales que el ser humano es capaz de resolver que la máquina de Turing no (modelización cognitiva), por tanto se requieren recursos de modelización no algorítmicos sino aproximados o *heurísticos.*

5**) Complejidad Computacional**

El criterio de decibilidad no es suficiente para establecer si una modelización cognitiva puede ser estrictamente algorítmica. Además los recursos físicos, espacio y tiempo, son limitados, a lo que se une que ni siquiera todos los problemas decidibles son *manejables*.

Una manera de estimar la manejabilidad de un problema es medir la complejidad computacional. Los parámetros utilizados para determinar esta medida son:

- Tiempo de computación requerido (número de operaciones primitivas)
- Espacio de memoria requerido:
  - Tamaño del input.
  - Tamaño del output.
  - Resultados intermedios.

La complejidad vendrá determinada por el tiempo requerido para su resolución, dada como una función del tamaño del problema. Los resultados que se obtienen al evaluar la complejidad son dos:

- Problema solubles en tiempo *Polinómico,* vulgarmente buenos
- Problemas solubles en tiempo *Exponencial,* vulgarmente malos

Ejemplo: Determinar la complejidad de la multiplicación.

- Sean dos enteros:
  - n, formado por *i* dígitos.
  - m, *k* dígitos.
- Para obtener mxn resultan i filas que se sumarán. En cada una de las filas se tendrán aproximadamente j ($\geq$k) operaciones. Por tanto para conseguir las i filas se requieren ixj operaciones.
  - Por último para sumar las filas se realizan alrededor de ixj sumas de dos números de un dígito cada uno.

- El tiempo de ejecución será proporcional al tiempo de ejecución de todas las operaciones. Como se sabe que el número de operaciones es de 2(ixj), se concluye que la complejidad será proporcional al factor variable de la expresión ixj.

A la hora de realizar la medida de la complejidad algorítmica es importante determinar su *tasa funcional de crecimiento.* Esta puede ser:

- Lineal (buena)
- Cuadrática (admisible en algunos casos)
- Exponencial (no manejable)
- …

La denotación de la complejidad se realiza a través de la expresión *O (f(n)),* la cual significa que la complejidad es de orden n o proporcional a n. En la determinación de la complejidad se ignoran las contantes y términos de menor orden. Es decir, si la complejidad algorítmica de un problema resulta en la siguiente expresión, $f=2n^2 + 3n + 5$, la denotación utilizada sería O (f(n)) = n^2, o de orden cuadrático.

Los problemas se clasifican según su complejidad de la siguiente forma:

- **Problemas P:** Existen algoritmos eficientes para su resolución en tiempo polinómico.
- **Problemas NP:** No se puede demostrar que sean problemas P, es decir, no se consigue encontrar un algoritmo que sea capaz de resolverlos en tiempo polinómico. Sin embargo, si existe una algoritmo eficiente (polinómico) para la verificación de su solución.
- **Problemas NP-Completos:** No son problemas P ni NP. En este caso si uno de la clase se solucionara mediante un algoritmo polinómico, todos tendrían un algoritmo de ese tipo, pero hasta hoy no se ha encontrado.

-

Anexo II: VRP "Vehicle Routing Problem"

1) Introducción

El VRP (vehicle routing problem), es el problema de enrutamiento de vehículos, data del año 1959 y fue introducido por Dantzig y Ramser, quienes describieron una aplicación real de la entrega de gasolina a las estaciones de servicio y propusieron una formulación matemática. Cinco años después, Clarke y Wright propusieron el primer algoritmo que resultó efectivo para resolverlo. De este modo, se dio comienzo a grandes investigaciones y trabajos en el área de ruteo de vehículos.

El problema VRP se corresponde con el problema del viajero (TSP, travelling salesman problema). El problema TSP pertenece al grupo de problemas NP-completos, es decir no existe una función matemática que lo resuelva en tiempo polinomial. Esto ha llevado a muchos investigadores a explorar diversos métodos para abordarlos. La mayoría de estos métodos pueden clasificarse como "exactos" o de "optimización" (Aarts y Lenstra, 2003). Estos algoritmos son descritos en el anexo IV.

Durante los años sesenta, los investigadores trataban de responder la siguiente pregunta: ¿existe un algoritmo de optimización con tiempo de ejecución polinomial para un problema como el TSP? Hasta ahora, nadie ha conseguido encontrar respuesta a esta pregunta. Sin embargo Karp, en 1972 mostró que si la respuesta es "si "para el TSP, hay también otros problemas difíciles para los cuales podría hallarse un algoritmo polinomial. Como no se ha encontrado solución para ninguno de estos problemas, Reeves en 1996 dice que esto sugiere categóricamente que la respuesta a la pregunta original es "no". Por ello mismo, el área de optimización combinatoria resulta cada vez más atrayente para investigadores y académicos, ya que cualquier contribución en este aspecto tiene repercusiones directas en la industria.

2) Problema del agente viajero (TSP)

El problema TSP constituye la situación general y de partida para formula otros problemas combinatorios más complejos pero más prácticos, como el ruteo de vehículos. En el TSP se dispone de un solo vehículo que debe visitar a todos los clientes en una sola ruta y a costo mínimo.

No suele haber un depósito (y si lo hubiera, no se distinguiría de los clientes), no hay demanda asociada a los clientes y tampoco hay restricciones temporales.

Si se denota por Δ+(i) y Δ–(i) al conjunto de nodos adyacentes e incidentes al nodo i, es decir, Δ+(i) = {j_V | (i, j)_E} y Δ–(i)={j_V | (j, i)_E}. De manera similar, el conjunto de arcos incidentes hacia el exterior e interior del nodo i se definen como δ+(i)= {(i, j)_E} y δ-(i) = {(j, i) _E}. El problema puede formularse matemáticamente mediante programación lineal entera (PLE) como sigue (Clarke y Wright, 1964):

$\min \sum_{i,j \in E} c_{ij} x_{ij}$, sujeto a:

$$\sum_{j \in \Delta^+(i)} x_{ij} = 1, \forall i \in V$$

$$\sum_{j \in \Delta-(j)} x_{ij} = 1, \forall j \in V$$

$$\sum_{i \in S, j \in \Delta^+(i)/S} x_{ij} \geq 1, \forall S \subset V$$

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

3) Complejidad del TSP y aproximaciones

La mayor parte de los problemas de ruteo de vehículos son generalizaciones del TSP. En ese sentido, puede considerarse el VRP más simple.

El tiempo de cálculo necesario para resolver el TSP se incrementa con rapidez a medida que aumenta el número de ciudades n. En un caso general el número de rutas factibles que debe considerarse es (n – 1)!/2, puesto que hay (n – 1) posibilidades para la primera ciudad después de la ciudad de residencia del agente, (n – 2) posibilidades para la siguiente ciudad y así sucesivamente. El denominador 2 surge porque cada ruta presenta una ruta inversa equivalente con la misma distancia (TSP simétrico). Así, mientras un TSP con 10 ciudades tiene no menos de 200.000 soluciones factibles que deben ser consideradas, un problema con 20 ciudades tiene alrededor de 1016 soluciones factibles, mientras que un problema con 50 ciudades tiene alrededor 1062 (Hillier y Lieberman, 2001). Es decir, se genera un espacio de búsqueda que crece exponencialmente, donde encontrar la solución óptima requiere de técnicas de cálculo especiales para dar con ella.

Para resolver el TSP normalmente se utilizan algoritmos de aproximación o heurísticos, la diferencia radica en que éstos nos dan una garantía de cómo podemos obtener malas soluciones. Normalmente especificada como un tiempo c del valor óptimo. El algoritmo que mejor solución ha propuesto, es el de Arora (1998). El algoritmo garantiza una aproximación de (1+1/c) veces el valor óptimo, para todo c > 1. Esto se basa en partición geométrica y árboles de expansión. Aunque teóricamente c puede ser muy grande, esto tendrá un efecto negativo en su tiempo de corrida (O(n(log2n)O(c) para instancias bidimensionales.

4) Problema de ruteo de vehículos (VRP)

A grandes rasgos un problema de ruteo de vehículos (VRP) consiste en, dado un conjunto de clientes y depósitos dispersos geográficamente y una flota de vehículos, determinar un conjunto de rutas de costo mínimo que comiencen y terminen en los depósitos, para que los vehículos visiten a los clientes máximo una vez. Dentro de esta definición, el problema se ubica en un amplio conjunto de variantes:

- CVRP (Capacitated VRP) (Ralphs, Hartman yGalati, 2001).
- MDVRP (Multi-Depot VRP) (Hjorring, 1995).
- PVRP (Periodic VRP) (Baptista, Oliveira yZúquete, 2002).

- SDVRP (Split Delivery VRP) (Dror, Laporte y Trudeau, 1994; Archetti, Mansini y Speranza, 2001).
- SVRP (Stochastic VRP) (Laporte y Louveaux, 1998).
- VRPB (VRP with Backhauls) (Ralplhs, Hartman y Galati, 2001); Jacobs-Blecha y Goetschalckx, 1992).
- VRPPD (VRP with Pick-Up and Delivering) (Righini, 2000).
- VRPSF (VRP with Satellite Facilities) (Bard et al., 1997).
- VRPTW (VRP with Time Windows) (Cordeau et al., 2002).

5) VRPTW

El VRPTW es el mismo problema que el VRP con la restricción adicional de que en el VRPTW se asocia una ventana temporal a cada cliente, que se define como el intervalo de tiempo en el que un cliente puede ser servido. El intervalo $[e_0, l_0]$ corresponde al almacén. Una descripción formal del problema sería la siguiente:

Objetivo: Minimizar la flota de vehículos y la suma del tiempo total de viaje y espera necesitado para suministrar a todos los clientes en sus horas solicitadas.

Viabilidad: El VRPTW respecto del VRP se caracteriza por las siguientes restricciones adicionales:

- Una solución pasa a ser inviable si un cliente es suministrado tras la cota superior de su ventana temporal.
- Si un vehículo llega antes del límite inferior de la ventana temporal del cliente genera un tiempo de espera adicional en la ruta.
- Cada ruta debe empezar y terminar dentro de la ventana temporal del almacén.
- En el caso de ventanas temporales amplias, un servicio tardío no afecta a la viabilidad de la solución, pero es penalizado añadiendo un valor a la función objetivo.

Formulación: Se denota b_v el comienzo del servicio para un cliente v. Una ruta $Ri = \{v_0, \ldots, v_m, v_{m+1}\}$, es viable si se satisface $e_{vi} \leq b_{vi} \leq l_{vi}$, $1 \leq i \leq m$ y $b_{vm} + \delta_{vm} + c_{vm,0} \leq l_0$, donde δ_{vi} es el tiempo de operación y $c_{vi,0}$ la distancia recorrida desde el almacén hasta el cliente v_i. Siempre que un vehículo viaje al siguiente cliente tan pronto como haya terminado en el cliente actual, b_{vi}, puede ser calculado recursivamente como $b_{vi} = max\{ e_{vi}, b_{vi-1} + \delta_{vi-1} + c_{vi-1,vi}\}$ con $b_0 = e_0$ y $\delta_0 = 0$. De este modo, un tiempo de espera $w_{vi} = \max\{0, b_{vi} - b_{vi-1} - \delta_{vi-1} - c_{vi-1,vi}\}$, podría ser incluido en el

cliente vi. El coste de la ruta i es ahora dado por $C_{VRPTW}(R_i) = \sum_{i=0}^{m} c_{i,i+1} + \sum_{i=1}^{m} \delta_i + \sum_{i=0}^{m} w_i$. Para una solución S con las rutas $R_1,\ldots, R_m$, el coste de S viene dado por $F_{VRPTW}(S) = \sum_{i=1}^{m} (c_{VRPTW}(R_i) + M)$, donde M es una constante. M es añadido ya que el principal objetivo es reducir el tamaño de la flota. S se dice que es viable, si todas las rutas contenidas en S son viables y sus clientes son servidos por una única ruta. Como describió Solomon [Solomon, 1995], se asume que todos los vehículos dejan el almacén tan pronto como es posible según e_0.Una vez obtenida una solución del VRPTW, se ajusta la salida del almacén de cada vehículo para eliminar tiempos de espera.

En la figura 1 se muestra un grafo de una posible solución del VRPTW. Las barras blancas y azules representan la ventana temporal, donde la parte blanca representa el momento en el que puede ser servido el cliente, la línea roja muestra el momento en el que es servido un cliente.

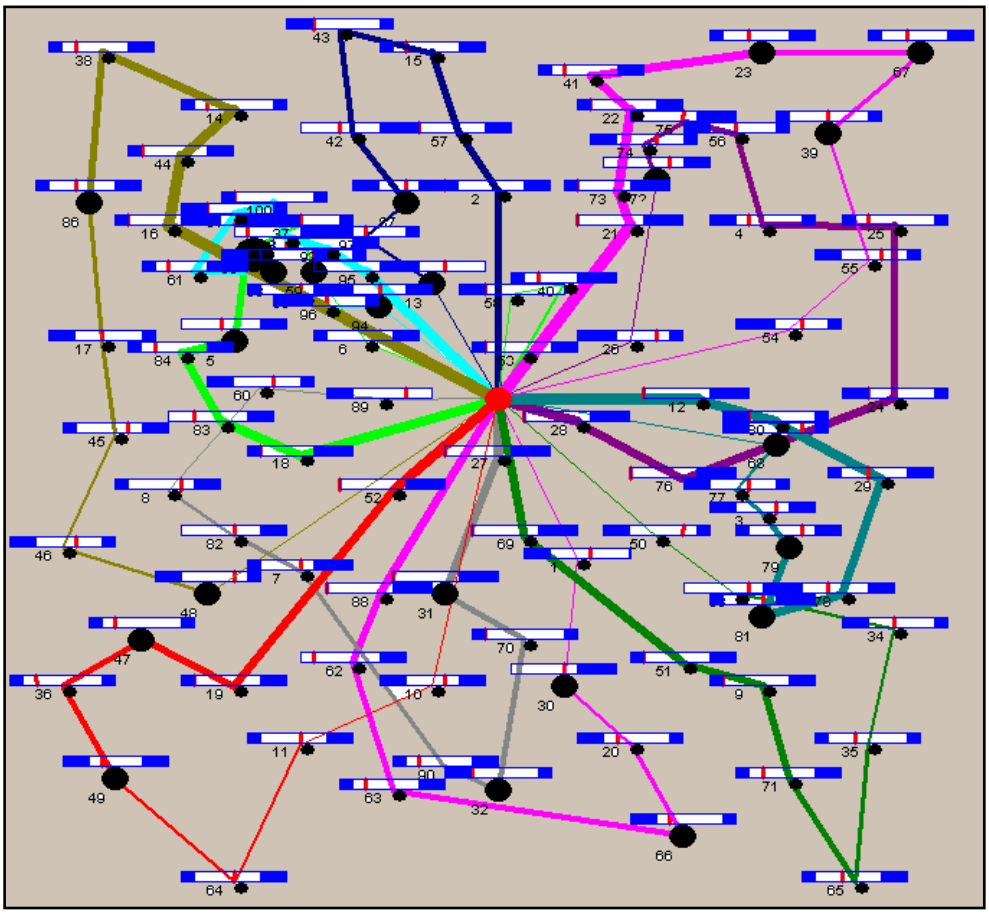

Figura 1. Solución VRPTW

6) 2L – CVRP

Este problema es una generalización del CVRP, el cual considera la demanda de los clientes según dos dimensiones, el peso y la forma rectangular de los paquetes. Es un problema combinado de dos NP-completos (CVRP, Anexo II sección 1.6) y el BPP (Anexo III).

Objetivo: Generar un conjunto de rutas que satisfagan las restricciones del CVRP, pero que además garantice la viabilidad de la carga dentro del espacio libre del camión.

Viabilidad: Una solución es viable si satisface las siguientes restricciones:

- El tamaño de la ruta generada no excede el número de vehículos disponibles vh.
- Todas las rutas comienzan y finalizan en el almacén.
- La demanda de todos los clientes es satisfecha.
- Cada cliente es visitado una sola vez.
- El peso de todos los ítems del conjunto de clientes pertenecientes a una ruta no excede la capacidad de un vehículo Q.
- No debe haber solapamientos en ninguno de los ítems demandados por el conjunto de de clientes cubiertos por una ruta dentro de la dimensión LxW de la superficie de carga disponible del camión, es decir, no se permite remontabilidad.

Formulación: Si se define N como el número de clientes junto con el almacén, *A* el número de arcos que unen todos clientes de N y c_{ij} el coste del arco que une el cliente i con el cliente j, con $i \neq j$.

- Número de vehículos que se disponen el almacén es de vh.
- Cada vehículo tiene una capacidad máxima de Q y unas dimensiones de LxW.
- La demanda de cada uno de los clientes, consiste de un conjunto ITi de mi ítems rectangulares de peso conocido.
- Cada ítem Iik en ITi (k=1..mi) tiene una longitud lik y una anchura wik.
- Se define ai, como la superficie total de todos los paquetes del cliente ITi.
- Los ítems deben ser colocados sobre las superficies de carga del camión: sus l- u w-esquinas deben ser paralelas a las L- y W-dimensiones de las superficies del vehículo, respectivamente.

Anexo III: BPP "Bin Packing Problem"

1) Introducción

El BPP (bin packing problem), o problema de la mochila, trata de colocar un cierto número de artículos en un determinado número de compartimentos, con el fin de minimizar el número total de compartimentos utilizados y el costo requerido para operar estos. Existen investigadores como Günther R. Raidl y S. Martello que se han centrado en este tipo de problemas en dos y tres dimensiones.

Este también pertenece al grupo de problemas NP-completos. Del mismo modo, el problema CVRP, o el problema de cálculo de rutas con capacidad limitada, se identifica con el problema del BPP o problema de la mochila.

2) Bin Packing Problem (BPP)

El problema consiste en embalar un conjunto de objetos en varias cajas o contenedores tal que el peso o el volumen total no exceda un valor máximo de las cajas. De una manera precisa, se define un problema de empaquetamiento en compartimentos (BPP, bin packing problem) como sigue. Se tiene un conjunto finito de artículos, cada uno de los cuales tienen un peso w y una restricción de precedencia entre estos, incurriendo en un costo

c_{ij} (tal vez infinito). Posteriormente se define un grupo ordenado para ser un subconjunto de artículos de modo que el peso total del grupo pedido no exceda la capacidad de la caja y ningún costo entre los artículos adyacentes en el grupo sea infinito. La meta primaria es crear una solución factible con el número mínimo de grupos ordenados. Cuando dos soluciones tienen el mismo número de grupos ordenados, el que posea costo mínimo se escoge.

Matemáticamente la formulación del problema puede ser así: dado un conjunto finito de elementos $E = \{e_1,\ldots, e_n\}$ con pesos asociados $W = \{w_1,\ldots,w_n\}$ tales que $0 \leq w_i \leq w(Bin)$, se divide E en N subconjuntos, de forma que la suma de pesos en cada partición sea a lo sumo w(Bin), teniendo en cuenta que N sea mínimo.

Una entrada y un resultado posible para el problema del bin packing so muestran en las figuras 1 y 2.

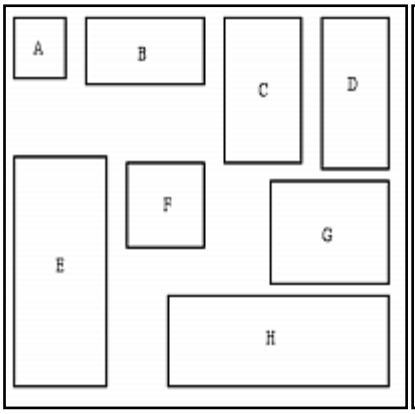

Figura 1: Entrada del BPP

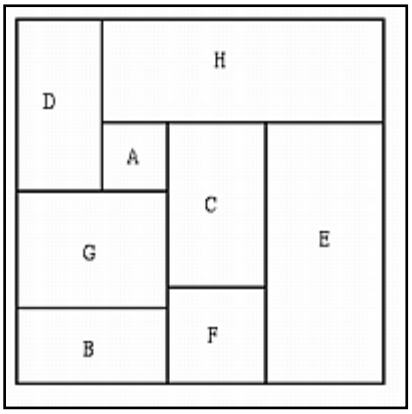

Figura 2: Salida del BPP

Anexo IV: Metaheurísticas

1) Introducción

Las heurísticas y metaheurísticas son métodos de solución que se basan en la optimización combinatoria. En la actualidad existe un incipiente interés en esta rama de las matemáticas debido a la complejidad de estos problemas en la obtención de soluciones óptimas en tiempo polinomial (anexo I).

El objetivo de estas técnicas, independientemente del campo en el que se apliquen, es que en general, tratan de construir sistemas que sean capaces de tomar decisiones en un entorno adverso.

Para entender el significado de terreno adverso en este contexto, se describe un ejemplo del problema TSP ya explicado en el anexo II. Sin embargo, aquí se incide en la generación del espacio de búsqueda para entender la importancia de la aplicación de estos métodos de optimización combinatoria.

- Se tiene un estado inicial: Clientes distribuidos en distintas localizaciones. En este caso como estado inicial: *{S0} = {cx, cy, cw, cz},* representado en la figura 1

- Se quiere llegar a un estado objetivo: Visitar a todos los clientes recorriendo la menor distancia posible, generando el menor coste, es decir, como estado final: *{SF} = {c1, c2, c3, c3},* representado en la figura 3
- Se necesita especificar una función objetivo que define las características del estado objetivo. La función objetivo es: min ($\sum_{i=1}^{4}\sum_{j=1}^{4} d_{i,j}$ donde i ≠j).
- Matriz con las distancias entre cada punto. Como se trata de la distancia euclídea, la distancia de i a j es la misma que de j a i figura 2.
- Para conocer la mejor solución se deben realizar todas las combinaciones posibles y devolver la que nos dé el valor óptimo de la función objetivo, representado en la figura 4.

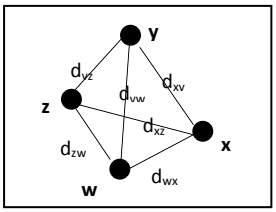

Figura 1. Estado Inicial .

	X	Y	W	Z
X		d_{yx}	d_{wx}	d_{zx}
Y	d_{yx}		d_{yw}	d_{yz}
W	d_{wx}	d_{yw}		d_{wz}
Z	d_{zx}	d_{yz}	d_{wz}	

Figura 2. Matriz de distancias.

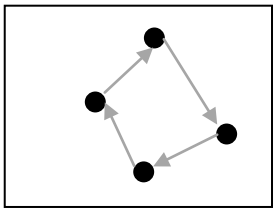
Figura 3. Estado Objetivo.

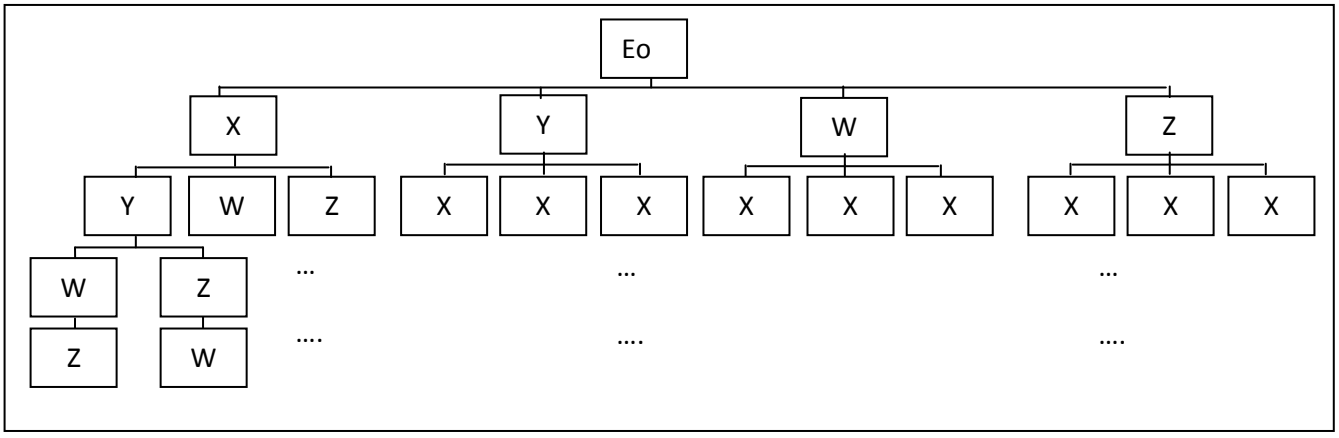

Figura 4. Combinaciones resultantes de TSP con 4 clientes

Tal y como se observa en la figura 4, los nodos o combinaciones resultantes son 24, es decir, factorial de 4, por tanto el número de combinaciones posibles dado un número de clientes n, es de factorial de n. Si n es muy grande el número de alternativas generadas a explorar es muy elevado, lo que hace que sea imposible realizar una exploración de todas ellas debido a que los recursos (tiempo, memoria) son limitados.

Con los métodos de optimización combinatoria se pretende reducir el espacio de búsqueda, de forma que algunos estados no son explorados. Esto deriva en que la solución encontrada no tiene que ser la óptima, pero si será una buena solución próxima a la mejor. La forma de realizar esto es a través de distintas técnicas, como la poda de ramificaciones que no

se consideran buenas. Por ejemplo en el caso anterior, solo se expanden los estados donde la distancia entre el estado i-1 y el estado i es la menor de todas las posibles combinaciones.

Estas técnicas de optimización combinatoria, se dividen en técnicas de optimización local convencional *(heurísticas)* y técnicas de optimización local inteligente *(metaheurísticas)*. A diferencia de un enfoque algorítmico "exacto", un método de optimización no tiene una base de matemática formal que lo sustente, es desarrollado más o menos por intuición (Ignizio y Cavalier, 1994).

2) Heurísticas

La idea más genérica del término heurística está relacionada con la tarea de resolver inteligentemente problemas reales usando el conocimiento disponible (Narducci, 2005). Heurística proviene de una palabra griega con un significado relacionado con el concepto de encontrar y se vincula a la supuesta exclamación "*¡heuriskein! "o* "*¡eureka! "de* Arquímedes al descubrir su famoso principio (De la Cruz, 2003). Reeves (1996) define el término heurística de la siguiente forma: *"Una técnica heurística (o simplemente una heurística) es un método que busca buenas soluciones (es decir, soluciones cercanas al óptimo) a un costo computacional razonable sin poder garantizar optimalidad"*.

3) Técnicas Heurísticas

Las técnicas heurísticas más relevantes se listan a continuación. Todas estas generan un árbol de búsqueda. La diferencia es el orden o criterio de expansión.

- Búsqueda en profundidad: expandir primero una rama hasta al final, según el valor de la función objetivo desde el inicio hasta el nodo a examinar.
- Búsqueda en anchura: expansión primero un nivel y pasar al siguiente, según el valor de la función objetivo desde el inicio hasta el nodo a examinar.
- Búsqueda A*: basada en una función objetivo que mide el costo de llegar desde el inicio hasta el nodo examinado, al que se le suma, una previsión del coste de alcanzar el estado objetivo desde el nodo examinado.

4) Técnicas Heurísticas para el VRP

Las técnicas heurísticas para el VRP, en general, pueden ser clasificadas dentro de cuatro categorías (Gaskell, 1967), así:

- *Constructivas*, como el método de los ahorros de Clarke y Wright, con base en el ahorro generado por insertar nuevos clientes en cada vehículo hasta completar una solución final.
- Métodos de *agrupar primero, luego enrutar*, que agrupan los clientes en varios subconjuntos, asignan cada subconjunto a un vehículo y luego resuelven cada TSP correspondiente (por ejemplo, el método de Fisher y Jaikumar, basado en el problema de asignación generalizado y el algoritmo de barrido de Gillet y Miller).
- Métodos heurísticos de *enrutar primero, luego agrupar*, que empiezan resolviendo el TSP definido por todos los clientes y luego parten la ruta hallada para asignar un tramo a cada vehículo (como el método de curvas de llenado de Bowerman, Calamai y Brenthall, y el método de partición óptima de Beasley).
- Los *métodos de mejoramiento*, como los intercambios Or–Opt.

5) Técnicas Metaheurísticas

Las metaheurísticas, también llamadas heurísticas modernas, han aparecido durante las últimas dos décadas (Yu, 1998). Su finalidad es tomar inicialmente una solución factible, para luego mejorarla usando heurísticas de mejoramiento embebidas en una estructura más general. La característica común de estos enfoques es el uso de mecanismos para evadir óptimos locales (Moraga, 2002). Glover y Laguna (1997) definen el término "metaheurística" como: "*una estrategia maestra que guía y modifica otras heurísticas para producir soluciones más allá de aquéllas que son normalmente generadas en una solicitud por optimalidad local. Las heurísticas guiadas por tal metaestrategia pueden ser procedimientos de alto nivel o nada más que una descripción de operaciones disponibles para transformar una solución en otra, junto con reglas de evaluación asociadas.* "

En la figura 5 se puede observar un compendio de las técnicas metaheurísticas utilizadas para resolver los problemas de optimización combinatoria. Se puede observar que se han empleado varias estrategias para resolver el problema que se pueden agrupar en tres grandes categorías: búsqueda secuencial por entornos (o vecindarios), redes neuronales y algoritmos evolutivos. Dentro de cada categoría se encuentran subclasificaciones, con el fin de especificar las características de los procedimientos, según sean probabilistas o deterministas, con uno o varios operadores, constructivos, con perturbaciones, con cruzamiento de información o sin él, etc.

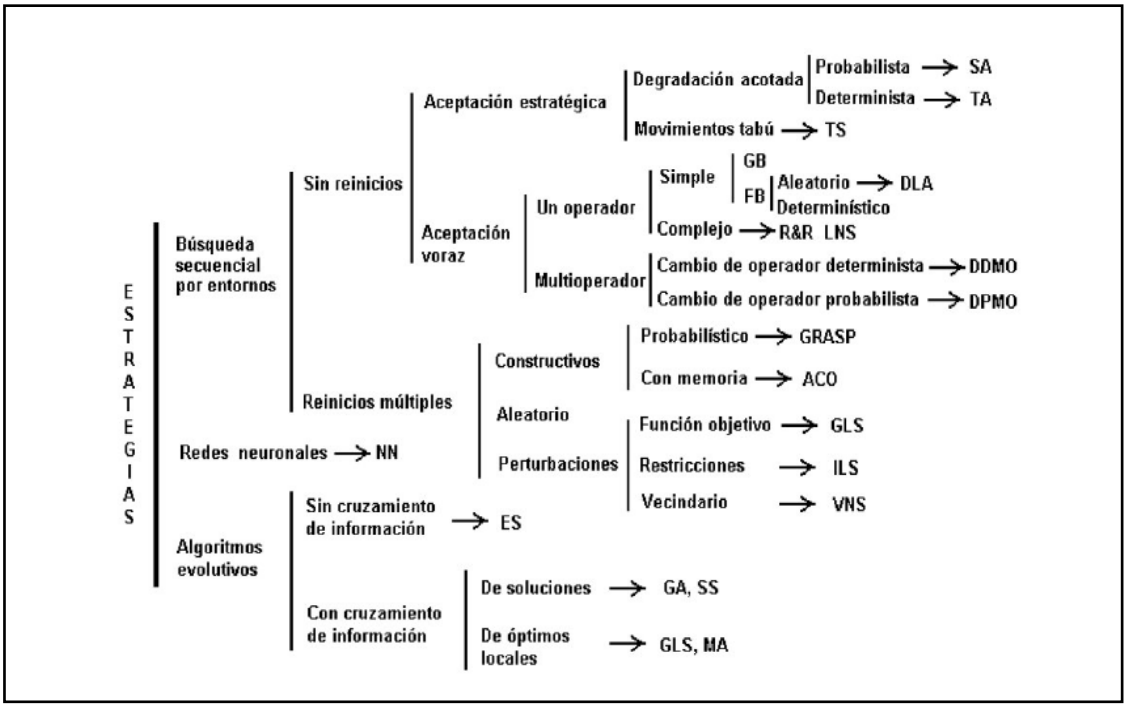

Figura 5. Técnicas para resolver problemas de optimización combinatoria.

6) Técnicas Metaheurísticas para el VRP

Ente las técnicas metaheurísticas para el VRP se encuentran las colonias de hormigas, búsqueda dispersa, algoritmos genéticos y la búsqueda tabú.

Cabe comentar que según Laporte et al. (2000), el procedimiento de búsqueda tabú ha sido la más exitosa metaheurística, en especial para resolver el VRP. En su libro Glover y Laguna (1997) presentan una muy buena discusión sobre la aplicabilidad de búsqueda tabú en problemas de optimización reales.

Anexo V: Colonias de hormigas

1) Introducción

En este anexo se explican los fundamentos básicos de la técnica metaheurística usada en este TFM, denominada colonia de hormigas.

En el comportamiento de las hormigas se distingue la búsqueda de alimentos, trazando el camino más corto entre el hormiguero y el emplazamiento de alimentos.

El eje de esa búsqueda es el depósito de *feromona* como *rastro* que orienta el recorrido. Las hormigas prefieren seguir la mayor concentración de feromona, la misma que se consigue por el recorrido más corto hecho por hormigas (que por esta razón, pueden repetirlo en forma más seguida). En principio siguen rutas aleatorias, pero las que han utilizado las de menor longitud pueden regresar más rápidamente (pues transitan a una velocidad uniforme), y repetir el camino, dejando una mayor densidad de huellas de feromona. Las depositadas en las otras rutas se van evaporando, y dejan de tener interés en los nuevos recorridos. Ello se ilustra en figura 1:

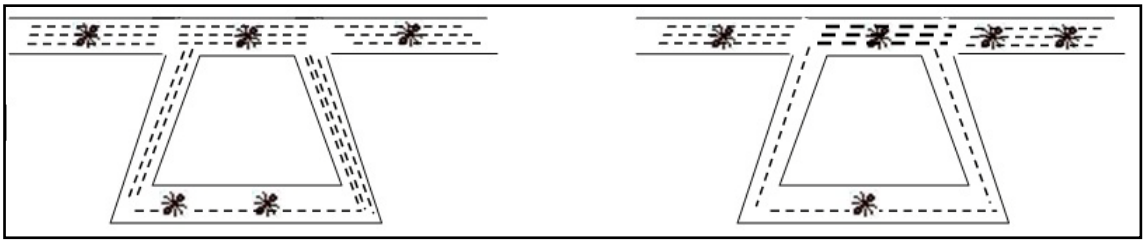

Figura 1. Selección de ruta más corta por mayor densidad de feromona.

El algoritmo basado en colonias de hormigas o ACO del inglés "*Ants colony optimitation"*, simula una colonia de hormigas artificial que trabajan en grupo y se comunican a través de de rastros de feromona artificial. Cada hormiga del problema construye una solución al problema recorriendo un grafo de construcción. Cada arista o tramo del grafo contiene dos tipos de información que guían el movimiento de la hormiga:

Parámetro heurístico: Denota la preferencia de moverse desde el nodo r al nodo s en la arista rs. Esta preferencia se va a designar en este TFM como u_{rs}, su valor no se modifica en el proceso y es función de la distancia entre los nodos.

Rastro de la feromona: Determina la deseabilidad aprendida en el movimiento de r a s, este valor su que se modifica y se representa con ζ_{rs}.

2) Optimización en el diseño de una estructura

2.1) Optimización de colonias de hormigas

La base fundamental del algoritmo es la simulación de la reacción que exhibe una colonia de hormigas. La sustancia virtual es denominada *rastro*, como símil a la feromona. El algoritmo base sigue la estructura computacional que se muestra en la figura 2:

Inicio de rutina o ciclo de rastros
Do While (*hasta que el criterio no sea satisfecho)*
Do Until (*cada hormiga completa un viaje*)
Construcción solución (decisión) de la hormiga
Mecanismo de decisión de la hormiga
Actualización del rastro local
End Do
Actualización del ciclo global de rastros
End Do

Figura 2. Estructura base del ACO.

La construcción de la solución necesita de un criterio para seleccionar la siguiente opción, esta *decisión* se basa en la intensidad de rastro presente en cada recorrido entre dos puntos adyacentes.

- El que tiene más valor de rastro o feromona, tiene la mayor probabilidad de ser elegido.

- Si no hay rastros, la probabilidad de elección es cero.

- Si todas las rutas tienen igual cantidad de rastros, la decisión es aleatoria.

Es decir la hormiga selecciona la ruta a través del mecanismo de decisión descrito, por tanto elegirá el siguiente punto del itinerario hasta haber visitado todos los nodos Una vez se han considerado todos los puntos, la hormiga regresa al nodo inicial habiendo completado un viaje. Cuando se finaliza la expedición se evalúa la calidad de la solución, de forma que las hormigas o soluciones que mejor lo resuelven renuevan, por un proceso de actualización global, el rastro asignando un valor mayor. De esta forma, cuando todas las hormigas de un ciclo han finalizado el viaje, son analizados y actualizados los valores de las feromonas. Entonces, un nuevo ciclo comienza y se repite el proceso. Casi todas las hormigas seguirán el mismo viaje en cada ciclo y convergerá la solución. Se compara la mejor solución en el último ciclo para obtener la solución global.

2.2) Algoritmo de colonias de hormigas para el VRP

Las soluciones para el VRP con colonias de hormigas están basadas en la propuesta original para el *caso del viajero* (Marco Dorigo, Milano, Italia, en 1992 y siguientes), quien debe recorrer desde su casa todas las ciudades buscando el recorrido más corto.

- La distancia entre las ciudades *i* y *j*, de coordenadas *(xi, yi)(xj,yj)*, respectivamente, se calcula como:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

- Siendo *n* el número de ciudades, *i* alguna ciudad, $b_i(t)$ el número de hormigas en *i* en el tiempo *t*, el total de hormigas *m* resulta:

$$m = \sum_{i=1}^{n} b_i(t)$$

- Cada hormiga tiene las siguientes características:

- Selecciona la siguiente ciudad a visitar. Esta selección está basada en una probabilidad que es función de la *distancia* a la ciudad (o parámetro heurístico) y la cantidad de *rastro de feromona* presente en cada tramo que conecta las ciudades, tal y como se observa en la figura 3.

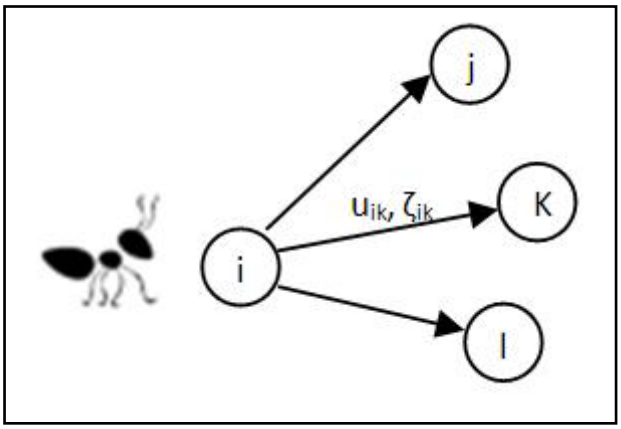

Figura 3. Parámetros de decisión.

- Tiene memoria sobre las ciudades que ha visitado (según una lista tabú) y las que no ha visitado.

- Cuando ha completado el viaje, deja una sustancia o rastro de feromona en cada tramo.

2.3) Rastro de la feromona

La *intensidad del rastro* en la ruta en cada *tramo* que conecta *i* con *j* en el tiempo *t*, se designa con $\zeta_{ij}(t)$.

En el tiempo *0*, $\zeta_{ij}(0)$ es tan pequeño como una constante pequeña positiva $\varepsilon > 0$. En el tiempo *t*, cada hormiga elige una ruta y viaja a la siguiente ciudad en el tiempo *t+1*.

En cada *iteración* hay *m* movimientos provocados por las *m* hormigas (cada una haciendo un movimiento) en el intervalo *(t, t+1)*. Un *ciclo* es definido como *n* iteraciones, y significa que las *m* hormigas han completado un viaje. Al final de cada iteración, el algoritmo implementa una *actualización local*: reduce el nivel de rastro en los tramos seleccionados por la colonia en la iteración precedente. Cuando una hormiga viaja a la ciudad *j* desde la ciudad *i*, la regla de actualización ajusta la intensidad del rastro del tramo conectado por:

$$\zeta_{ij} = (1-\psi)\zeta_{ij}(t) + \psi\zeta_0$$

Donde Ψ es un parámetro ajustable entre 0 y 1 que representa la persistencia del rastro de la feromona. Además:

$$\zeta_0 = \frac{1}{nL_{nn}}$$

Donde nLnn es la longitud calculada con el algoritmo heurístico *el vecino más cercano* (se comienza aleatoriamente en una ciudad y se selecciona la siguiente basada en la distancia más corta hasta visitar todas las ciudades).

Al final de cada ciclo la intensidad del rastro es ajustada usando una *actualización global*. El primer paso es calcular la cantidad de rastro dejada en cada tramo, como:

$$\Delta\zeta_{ij}{}^{k} = \frac{1}{L_k}$$

Donde *k* representa alguna hormiga (de *1* a *m*). Y *Lk* es la longitud del ciclo elegida por una hormiga *k*.

El valor total de actualización del rastro para cada tramo es la suma de las cantidades dejadas por cada hormiga que haya seleccionado dicho tramo *ij*, y será:

$$\Delta\zeta_{ij}{}^{k} = \sum_{i=1}^{m} \Delta\zeta_{ij}{}^{k}$$

El valor de la actualización global está determinado por la siguiente expresión:

$$\zeta_{ij}(t+n) = (1-\rho)\zeta_{ij}(t) + \rho\Delta\zeta_{ij}$$

Donde ρ es una constante entre 0 y 1, tal que (1 – ρ) representa la evaporación del rastro entre t y t+n (tiempo requerido para completar un ciclo).

2.4) Parámetro heurístico

El sistema admite que las hormigas posean un grado de visibilidad determinado por el parámetro heurístico uij, asociado a cada tramo, que depende de la distancia del mismo, teniendo preferencia los cercanos:

$$u_{ij} = \frac{1}{d_{ij}}$$

2.5) Probabilidad de preferencia

La combinación de la probabilidad heurística con el rastro de la feromona resultan en:

$$a_{ij}(t) = \frac{[\zeta_{ij}]^{\alpha}[u_{ij}]^{\beta}}{\sum_{l \in permitidas}^{n} [\zeta_{il}]^{\alpha}[u_{il}]^{\beta}}$$

Se trata de la proporción respecto a las ciudades vecinas permitidas desde la ciudad i. Los parámetros α y β son constantes usadas para controlar la importancia de los valores de los rastros de las feromonas locales y visibles.

La probabilidad $p_{ij}^{k}(t)$ de que la hormiga K elija ir de i a j en el tiempo t es:

$$p_{ij}{}^{k}(t) = \frac{a_{ij}(t)}{\sum_{l \in permitidas}^{n} a_{il}(t)}$$

Si la hormiga no está permitida viajar a j entonces, $p_{ij}^{k}(t) = 0$.

Anexo VI: Descripción del problema

1) Introducción

En este anexo se presenta un caso práctico del problema nacional de transporte con las singularidades que se han planteado en el documento principal. Para facilitar la compresión al lector se adjunta una serie de imágenes y descripciones de los pasos.

2) Presentación del caso práctico

Con ayuda del mapa representado en la figura 1 se pueden observar los pedidos a servir por una hipotética empresa de transporte. Las cruces simulan a los clientes internos donde se realizan labores de picking y cuyas ventanas temporales son más flexibles. Los clientes externos se encuentran esparcidos por la geografía española y simplemente se identifican con la operación a realizar en ese punto.

Para la realización de este ejemplo se han figurado una serie de pedidos (como se ha comentado a lo largo de esta memoria, un pedido P_i, viene representado por el par (r_i, e_i)), las recogidas de cada pedido se han dibujado en el mapa de color rojo y las entregas en color verde. Gracias al mapa se obtiene una foto de las localizaciones de las operaciones a realizar que facilita la comprobación de los resultados.

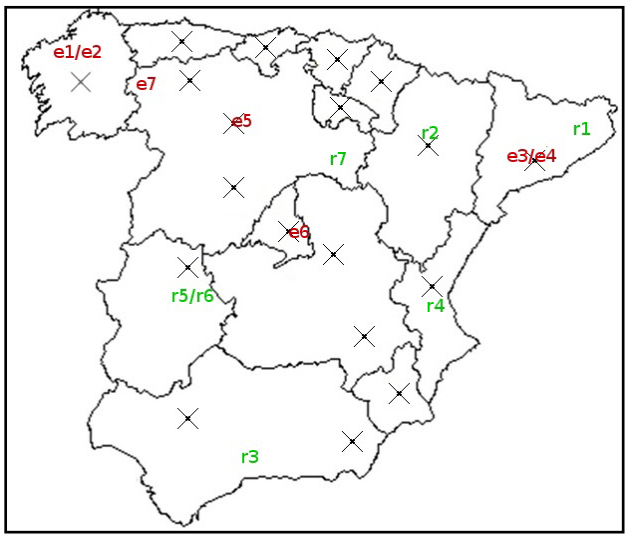

Figura 1. Pedidos a realizar por la empresa en un día.

Como se puede observar algunos pedidos se realizan sobre clientes internos y otros sobre clientes externos. En esta imagen se han presentado clientes donde una operación del pedido se realiza sobre un tipo de cliente y la otra parte de la operación se realiza sobre otro tipo de cliente. Por ejemplo, el $P_2(r_2, e_2)$, r_2 es un almacén interno y la e_2 es un cliente externo, o al revés, como $P_3(r_3, e_3)$, donde r_3 es un cliente externo y e_3 interno. Además también se observa que dos pedidos pueden tener operaciones que se produzcan en el mismo cliente, como en el caso de $P_5(r_5, e_5)$ y $P_6(r_6, e_6)$ donde la localización de r_5 es la misma que la de r_6; para las entregas aparecen dos casos $P_1(r_1, e_1)$ con $P_2(r_2, e_2)$ y $P_3(r_3, e_3)$ con $P_4(r_4, e_4)$ donde coinciden las localizaciones de e_1 con e_2 y de e_3 respecto de e_4. No se ha presentado el caso en que los orígenes y los destinos de un par de pedidos coincidan pero sí que se contempla como posibilidad. Por último el $P_7(r_7, e_7)$, tiene ambas operaciones aisladas. Además a la hora de realizar la agrupación de pedidos se tienen que considerar si esa agrupación es viable, es decir, si va a satisfacer las siguientes restricciones:

- Satisfacción de ventanas temporales.
- Compatibilidad entre las mercancías.
- Compatibilidad del vehículo con la mercancía.
- Restricción de peso.
- Restricción de disponibilidad, es decir, si el conjunto de ítems de un cliente cabe en la superficie disponible del camión.
- Estructura Lifo, el último pedido recogido será el primero en entregar.

Para entender cómo actúan estas restricciones se presenta la tabla 1 que contiene:

- Las ventanas temporales de las recogidas y entregas de los clientes.
- Tipo de carga: frio, isotermo o normal.
- Kilogramos.

Además se suponen los siguientes datos:

- Capacidad del camión de C=24.000 kg, la cual no puede ser excedida.
- Las recogidas se realizan todas el mismo día y las entregas al día siguiente de forma que se satisfagan las limitaciones en cuanto al tiempo de disco, tiempo de descanso del conductor,….

No se ha incluido información sobre la superficie de los ítems para la restricción de disponibilidad, (se encuentra una configuración en el camión que permite introducir todos los paquetes de un pedido), ya que con la expuesta en la tabla es suficiente para comprender la complejidad de este problema combinatorio. Sin embargo, merece la pena comentar que esta restricción permite optimizar el llenado al realizar un mejor aprovechamiento de la superficie del vehículo y por ello se ha tratado en este TFM.

PAR	**HORA RECOGIDA**	**HORA ENTREGA**	**TIPOLOGÍA**	**VEHÍCULO ESPECIAL**	**KG**
(r1,e1)	07:00	07:00	Alimentación	Frío	12.000
(r2,e2)	07:00	07:00	Alimentación	Frío	6.000
(r3,e3)	08:00	07:00	Industrial		20.000
(r4,e4)	12:30	07:00	Industrial		2.000
(r5,e5)	07:30	08:00	Alimentación		10.000
(r6,e6)	07 n:30	08:00	Droguería		12.000
(r7,e7)	08:30	07:00	Alimentación	Frío	15.000

Tabla 1. Información pedidos de un día para un caso hipotético.

3) Solución del caso práctico

A primera vista parece un problema trivial, es decir, a priori la agrupación que se realizaría sería coger r_1 y r_2, ya que el destino es el mismo. Lo mismo ocurriría con r_3 y r_4 y con r_5 y r_6. Pero, *¿Pueden realizarse estas agrupaciones?, ¿satisfacen todas las restricciones?,* es decir, *¿se cumplen las fechas pactadas, compatibilidad carga-camión, compatibilidad entre la carga?, ¿esta agrupación resulta óptima?, ¿dónde metemos a* r_7*? Etc.*

Con la información listada en el apartado anterior y los datos de la tabla 1 se analiza la solución que por intuición se formularía:

- r_1 junto a r_2: no se permite ya que no satisface la hora de recogida en r2.
- r_3 junto a r_4: se permite que vayan juntos.
- r_5 junto a r_6: no se permite ya que la carga r5 es incompatible con r6.
- r_7: solo.

En caso de que la solución fuera viable, las rutas resultantes se muestran en la figura 2. Los desplazamientos entre las localizaciones se han dibujado en línea recta, aunque en la vida real el trazado se realizaría siguiendo el trazado de carreteras vigente. Analizando el mapa se observa que el número de expediciones que se obtiene es de 6. El aprovechamiento global de cada camión y el global se muestran en la tabla 2.

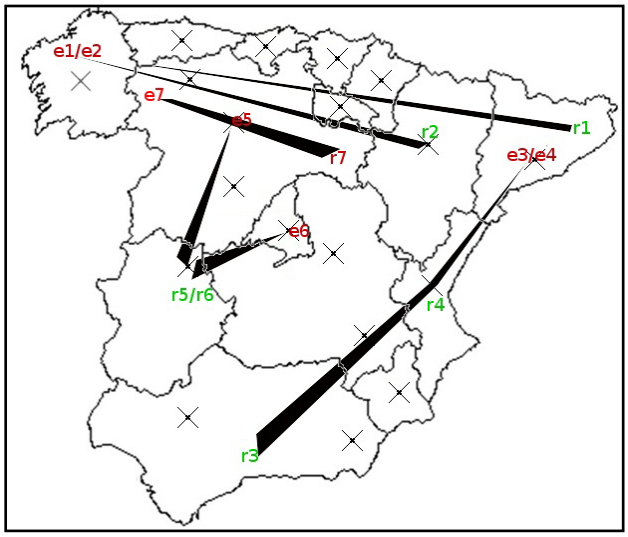

Figura 2. Rutas resultantes de la solución obtenida a priori por intuición no viable.

RUTA	**APROVECHAMIENTO (%)**
r1,e1	50
r2,e2	25
r3,r4,e3,e4	92
r5,e5	41,7
r6,e6	50
r7,e7	62,5
Global	**53,53**

Tabla 2. Aprovechamiento camión de la solución obtenida a priori por intuición no viable.

Para obtener una solución que resulte viable y que de las viables sea la que genere un menor coste, es decir, que sea óptima, se debería de realizar la combinación de todos los pares con todos los pares. Si se realizara esta combinación se obtendría la solución mostrada en la figura 3 y el rendimiento de cada vehículo resultante para cada camión sería el representado en la tabla 3.

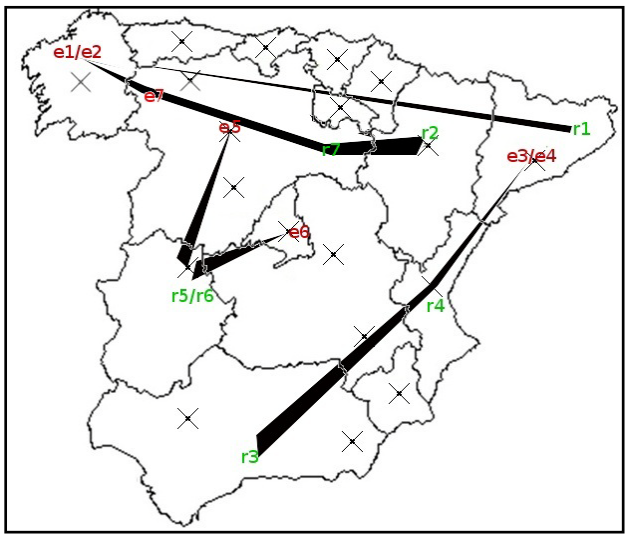

Figura 3.Rutas solución óptima.

RUTA	**APROVECHAMIENTO (%)**
r1,e1	50
r2,r7,e7,e2	87,5
r3,r4,e3,e4	92
r5,e5	41,7
r6,e6	50
Global	**64,24**

Tabla 3. Aprovechamiento camión de la solución óptima

La solución óptima obtenida además de resultar viable respecto de la anterior, consigue un mayor aprovechamiento en global y reduce el número de camiones a 5. Sin embargo la complejidad de este problema crece exponencialmente con el número de clientes, ya que las combinaciones crecen en esta medida. Esta complejidad es la que hace que se necesiten de métodos metaheurísticos (anexo IV), de aproximación a la solución óptima. Estas técnicas necesitan de la definición de una función objetivo, que es el objeto de estudio de este TFM

.

Anexo VII: Metodología

1) Introducción

En este anexo se detallan las explicaciones realizadas en el apartado 3 del documento principal, referidas a la metodología propuesta para resolver el problema. Para ello se incluye la información aportada en ese documento con las extensiones oportunas.

2) Datos

Los datos necesarios para la resolución del problema se listan a continuación. Una descripción más precisa sobre la estructura, tipos, etc. es realizada en el anexo IX.

- **Punto**: Con este dato se caracteriza la estructura de datos utilizada para representar las localizaciones correspondientes al origen y al destino de un pedido, es decir, posición geográfica e intervalo temporal de operación (entrega o recogida).

- **Ítem**: Contendrá las dimensiones de un paquete correspondiente a un pedido.
- **Pedido**: Esta estructura define los atributos que determinan a un pedido. Es decir, un pedido será recogido en un punto origen, transportado a un punto destino, los kilos a transportar, el conjunto de ítems, …
- **Matriz Distancias**: Cada celda de la matriz será la distancia real (carretera) que separa el punto de la columna i con el punto de la columna j, siendo j distinta de i.
- **Tipología**: Tipos de carga a transportar: Frio, Isotermo, Normal.
- **Ruta**: Secuencia de pedidos.
- **Solución**: Secuencia de rutas.
- **Vehículo**: Características que definen al medio de transporte: capacidad, dimensiones, tipo de carga que transporta.

3) Estructura del algoritmo global

El algoritmo general planteado, se divide en tres etapas tal y como se muestra en la figura 1. A diferencia del documento principal, en este caso se realiza una descripción detallada cada una de las etapas.

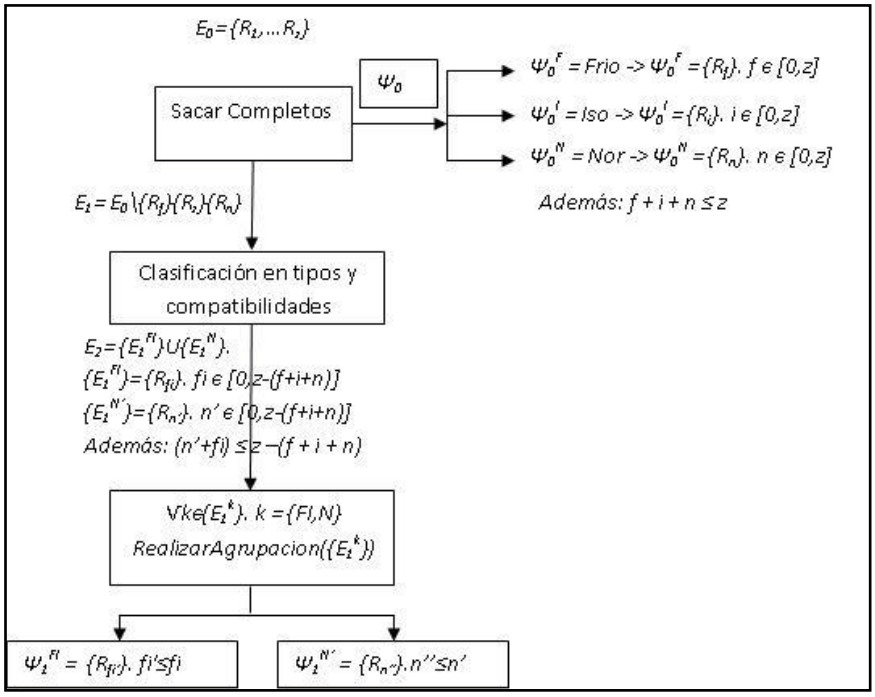

Figura 1. Estructura algoritmo general.

- **Sacar Completos**: Tiene como objeto sacar de la agrupación o del cálculo aquellos pedidos que se consideran carga completa. Como salida se obtienen las rutas directas Ψ_0 y las cargas sobrantes que van al paso siguiente para ser agrupadas. El comportamiento y las entradas y salidas del mismo serán:

- Como entrada tiene el conjunto de rutas "$E_0 = \{R_1,\ldots, R_z\}$".
- Cada una de las rutas se corresponde con un pedido, es decir, *"*$\{R_j\} = \{P_j\}.\ j \in [1, z]$".
- Como salida de este proceso del algoritmo de cálculo se tiene a Ψ_0, que contienen las rutas directas cuyos camiones contienen un solo pedido que corresponde a una carga completa.
- El conjunto de soluciones $\Psi_0 = \{R_i,\ldots, R_k\}$, a su vez se divide en tres grupos según la tipología de la mercancía del pedido. Es decir, por un lado se tiene $\Psi_0^F = \{R_f\}$ *con* $f \in [0, z]$, para tipo frio, $\Psi_0^I = \{R_i\}$ *con* $i \in [0, z]$ *para* isotermo y $\Psi_0^N = \{R_n\}$ *con* $n \in [0, z]$, para el resto de la mercancía denominada como normal.
- El número total de rutas de Ψ_0 será la suma de cada subconjunto, y no puede ser superior al número de pedidos existentes. Es decir, $f + i + n \leq z$.

- **Clasificación en tipos y compatibilidades**: En este caso se tienen únicamente los pedidos que no alcanzan para viajar solos en un vehículo, es decir, pueden viajar agrupados. Para satisfacer la restricción de compatibilidad y reducir el número de pedidos que entraran en la combinación se divide el conjunto en dos grupos (Frio + Isotermo y Normal). Los datos de entrada y resultados son:
  - Como entrada se tiene a *"*$E_1=E_0 \setminus \{R_f\}\{R_i\}\{R_n\}$*"*, es decir, el conjunto de rutas inicial menos los que se han quitado en el paso anterior, por tanto, el número de rutas o pedidos al inicio de esta parte es "*z-(f + i + n)".*
  - En este proceso se divide el conjunto E_1 en dos subconjuntos según la tipología de la carga. Es decir, la carga de cada subconjunto no puede entremezclares, por tanto, se tiene $E_2= \{E_1^{FI}\} \cup \{E_1^N\}$, donde: $\{E_1^{FI}\}= \{R_{fi}\}.\ fi \in [0, z-(f + i + n)]$ y $\{E_1^{N'}\}= \{R_{n'}\}.\ n' \in [0, z-(f + i + n)]$ tal que $(n' + fi) \leq z - (f + i + n)$.

- **Realizar Agrupación:** El objetivo es realizar la agrupación de clientes de cada subconjunto del paso anterior (Frio + Isotermo y Normal). En este apartado se aplican las técnicas metaheuristicas de planificación de rutas y llenado del camión, que se explican en los siguientes apartados. Como resultado de este paso se obtienen los itinerarios con los pedidos agrupados $\Psi_1 = \Psi_1^{FI} \cup \Psi_1^N$. Las entradas, salidas y comportamiento de este proceso encargado de poner en ejecución los métodos metaheurísticos son:
  - Este proceso se ejecutara dos veces, los subprocesos pueden realizarse en paralelo ya que uno es independiente del otro. Cada una de las ejecuciones se corresponde con uno de los resultados del apartado anterior, es decir, se realiza por un lado la agrupación del conjunto $\{E_1^{FI}\}$ y por otro lado la del conjunto $\{E_1^N\}$.
  - La salida de este proceso serán las rutas correspondientes a cada uno de los conjuntos. En este caso una ruta puede estar formada por más de un pedido, por

tanto, como resultado se obtiene $\Psi_1^{FI} = \{R_{fi'}\}$. $fi' \leq fi$ para la entrada $\{E_1^{FI}\}$ y $\Psi_1^{N'} = \{R_{n''}\}$.$n'' \leq n'$ correspondiente al input $\{E_1^{N}\}$.

- El conjunto de rutas resultante será la suma de ambas soluciones $fi' + n''$. Este número no podrá ser superior al número de rutas entrantes pero si menor ya que una ruta puede estar formada por más de un pedido. Entonces, $0 < fi' + n'' \leq fi + n' = z - (f + i + n)$.

Como resultado del algoritmo se obtendrán las rutas a realizar por los vehículos. La solución será por lo tanto, $\mathbf{S = \Psi_0 \cup \Psi_1}$. Este proceso es explicado a continuación.

4) Metaheurística para la planificación de rutas *(agrupación pedidos)*

En este apartado se plantea el algoritmo adaptado a partir de una investigación realizada por L. Barcos, V. Rodríguez, M.J. Álvarez y F. Robusté denominado "*Routing design for less-than-truckload motor carriers using Ant Colony Optimization"*. El objetivo es realizar la agrupación de los pedidos en vehículos con el objeto de disminuir el número de recursos a utilizar y el kilometraje total recorrido, es decir, se describe el funcionamiento de la caja *realizar agrupación* de la figura 1.

4.1) Estructura general del algoritmo

Este algoritmo está basado en una metaheurística denominada "Algoritmo de colonia de hormigas" descrito en el anexo V, se ejecutara dos veces en paralelo, con la intención de reducir el tiempo consumido. La entrada para cada uno se los procesos será uno de los subconjuntos clasificados en el proceso "Clasificación en tipos y compatibilidades" representado en la figura 1, es decir, $\{E_1^{FI}\}$ *o* $\{E_1^{N}\}$.

A través del esquema de la figura 2 se realiza una representación del flujo o comportamiento del algoritmo, seguido de una descripción general. A continuación, para clarificar un poco más el funcionamiento del mismo se incluye el pseudocódigo del algoritmo, mostrado en la figura 3 y explicaciones detalladas de cada una de las sentencias que lo forman.

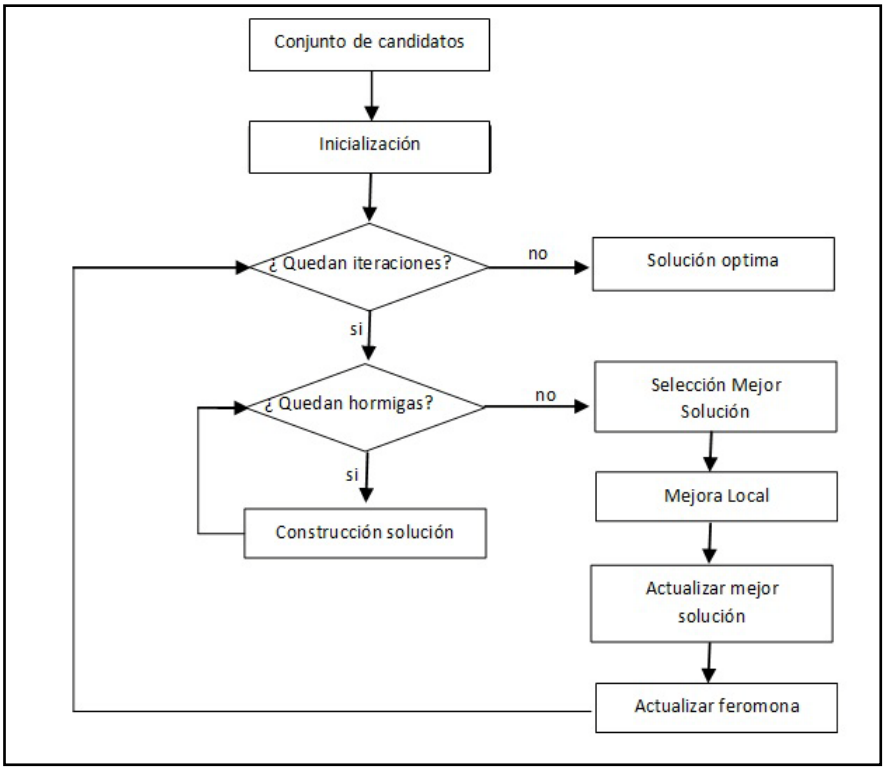

Figura 2. Estructura general del algoritmo de planificación de rutas.

El primer paso tiene como objetivo reducir el número de posibles combinaciones del subconjunto seleccionado. Para ello se definen cuatro grupos de candidatos *{G^1}, {G^2}, {G^3} o {G^4}*.En este TFM se ha considerado que un pedido está constituido por el par origen – destino, y se establecen una serie de relaciones con el resto de pedidos. La generación de estas relaciones se detalla al realizar la explicación del pseudocódigo mostrado en la figura 3.

Una vez finalizado el paso anterior, se ordenan los pedidos a agrupar en una lista de mayor a menor distancia entre el origen y el destino del pedido.

A continuación comienzan dos bucles. El primero de ellos es el que determina el refinamiento de la solución, es decir, cuantas veces se va a pulir la mejor solución encontrada hasta el momento. En el interior de este bucle se encuentra otro proceso iterativo en el que se genera la colonia de hormigas, es decir, se genera un número de soluciones igual al número de iteraciones del bucle. La colonia de hormigas es resultado de la ejecución de un algoritmo denominado "*construcción solución*" explicado a continuación y que en el pseudocódigo se ha etiquetado con el número (3.1.1). A continuación se realiza el refinamiento de la mejor solución encontrada hasta el momento. Para ello en primer lugar se selecciona la mejor solución de la colonia de hormigas, después se realiza una mejora local sobre la misma, y por último se compara si esta es mejor que la mas buena encontrada en las iteraciones anteriores, en caso afirmativo esta pasa ser la solución, en caso contrario se deja la que estaba. Por último

se actualiza el valor de la feromona. La mejora local y el mecanismo de actualización de la feromona son explicados a continuación.

El pseudocódigo que implementa el esquema de la figura 2 se incluye en la figura 3 y es utilizado como apoyo para explicar de forma concisa el funcionamiento del mismo.

Determinación de los conjuntos de candidatos	(1)		
Inicialización del proceso	(2)		
Para t = 1 **hasta** NIt **hacer**	(3)		
Para k = 1 **hasta** NH **hacer**		(3.1)	
Construcción de la solución Ψ (t, K)			(3.1.1)
FinPara			
Selección de la mejor solución de la iteración actual Ψ^{ib} (t)		(3.2)	
Mejora local de esta solución Ψ^{ib*} (t)		(3.3)	
Actualización de la mejor solución global Ψ^{gb} (t)		(3.4)	
Actualización del valor de la feromona		(3.5)	
FinPara			

Figura 3. Pseudocódigo del algoritmo general de planificación de rutas.

Cada uno de los puntos del pseudocódigo de la figura 3 es explicado a continuación.

Determinación de los conjuntos de candidatos (1)

Este paso tiene como objetivo reducir el número de posibles combinaciones del subconjunto seleccionado $\{E_1^{Fl}\}$ *o* $\{E_1^{N}\}$.Para ello se definen cuatro grupos de candidatos $\{G^1\}$, $\{G^2\}$, $\{G^{3,}\}$ *o* $\{G^4\}$. Es decir, para cada pedido del conjunto $\{E_1^{Fl}\}$ *o* $\{E_1^{N}\}$.se generan los subconjuntos $\{G^1\}$, $\{G^2\}$, $\{G^3\}$ *o* $\{G^4\}$ de candidatos según la relación que guarde con el resto de pedidos del conjunto $\{E_1^{Fl}\}$ *o* $\{E_1^{N}\}$.En este TFM se ha considerado que un pedido está constituido por el par origen – destino, y se establecen una serie de relaciones con el resto de pedidos descritas a continuación:

1. *Mismo origen y mismo destino* ***$\{G^1\}$***

Se define $\{G_i^1\}$, como el conjunto de pedidos pertenecientes a $\{E_1^{Fl}\}$ *o* $\{E_1^{N}\}$, que tienen el mismo origen y el mismo destino que el pedido $P_i(r_i, e_i)$. Es decir, sea $P_j(r_j, e_j)$ otro pedido perteneciente al mismo conjunto $\{E_1^{Fl}\}$ *o* $\{E_1^{N}\}$ que P_i, entonces:

Con la siguiente expresión se describe la relación mismo origen – mismo destino que se establece entre dos pedidos.

$$\{G_{ij}^1\} = \{(P_i \cup P_j) \leftrightarrow (r_i = r_j \,\&\, e_i = e_j) \text{ tq } (r_i, e_i) \in P_i \,\&\, (r_j, e_j) \in P_j\}$$

A modo ilustrativo el significado de la expresión anterior se muestra en la figura 4.

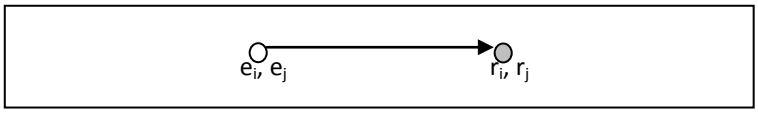

Figura 4. Relación $\{G^1\}$ entre el pedido i y el pedido j.

La unión de todos los pedidos P_j que guarden la relación $G_{ij}^{\ 1}$ con P_i constituyen el conjunto siguiente:

$$\{G_i^{\ 1}\} = \bigcup_{j=1}^{E_1^{\ Fl}} \{G_{ij}^{\ 1}\} \text{ o } \{G_i^{\ 1}\} = \bigcup_{j=1}^{E_1^{\ N}} \{G_{ij}^{\ 1}\} \text{ donde } i \neq j$$

Por último, la unión de todas las relaciones $G_i^{\ 1}$para todo i perteneciente a $\{E_1^{\ Fl}\}$ *o* $\{E_1^{\ N}\}$, constituyen el grupo:

$$\{G^1\} = \bigcup_{i=1}^{E_1^{\ Fl}} \{G_i^{\ 1}\} \text{ o } \{G^1\} = \bigcup_{i=1}^{E_1^{\ N}} \{G_i^{\ 1}\}$$

2. *Mismo origen y distinto destino* ***$\{G^2\}$***

Se define $\{G_i^{\ 2}\}$, como el conjunto de pedidos pertenecientes a $\{E_1^{\ Fl}\}$ *o* $\{E_1^{\ N}\}$, que tienen el mismo origen y diferente destino que el pedido $P_i(r_i, e_i)$. Es decir, sea $P_j(r_j, e_j)$ otro pedido perteneciente al mismo conjunto $\{E_1^{\ Fl}\}$ *o* $\{E_1^{\ N}\}$ que P_i, entonces:

Con la siguiente expresión se describe la relación mismo origen – distinto destino que se establece entre dos pedidos.

$$\{G_{ij}^{\ 2}\} = \{(P_i \cup P_j) \leftrightarrow (r_i = r_j \,\&\, e_i \neq e_j) \text{ tq } (r_i, e_i) \in P_i \,\&\, (r_j, e_j) \in P_j\}$$

A modo ilustrativo el significado de la expresión anterior se muestra en la figura 5.

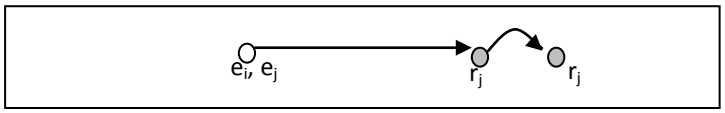

Figura 5. Relación $\{G^2\}$ entre el pedido i y el pedido j.

La unión de todos los pedidos P_j que guarden la relación $G_{ij}^{\ 2}$ con P_i constituyen el conjunto siguiente:

$$\{Gi2\} = \bigcup_{j=1}^{E_1^{\ Fl}} \{G_{ij}^{\ 2}\} \quad \text{o } \{Gi2\} = \bigcup_{j=1}^{E_1^{\ N}} \{G_{ij}^{\ 2}\} \quad \text{donde } i \neq j$$

Por último, la unión de todas las relaciones $G_i^{\ 2}$para todo i perteneciente a $\{E_1^{\ Fl}\}$ *o* $\{E_1^{\ N}\}$, constituyen el grupo:

$$\{G2\} = \bigcup_{i=1}^{E_1^{Fl}} \{G_i^2\} \quad \text{o} \quad \{G2\} = \bigcup_{i=1}^{E_1^{N}} \{G_i^2\}$$

3. *Distinto origen y mismo destino* ***$\{G^3\}$***

Se define $\{G_i^3\}$, como el conjunto de pedidos pertenecientes a $\{E_1^{Fl}\}$ *o* $\{E_1^{N}\}$, que tienen el mismo origen y diferente destino que el pedido $P_i(r_i, e_i)$. Es decir, sea $P_j(r_j, e_j)$ otro pedido perteneciente al mismo conjunto $\{E_1^{Fl}\}$ *o* $\{E_1^{N}\}$ que P_i, entonces:

Con la siguiente expresión se describe la relación distinto origen – mismo destino que se establece entre dos pedidos.

$$\{G_{ij}^3\} = \{(P_i \cup P_j) \leftrightarrow (r_i \neq r_j \,\&\, e_i = e_j)) \text{ tq } (r_i, e_i) \in P_i \,\&\, (r_j, e_j) \in P_j\}$$

A modo ilustrativo el significado de la expresión anterior se muestra en la figura 6.

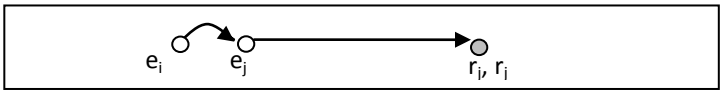

Figura 6. Relación $\{G^3\}$ entre el pedido i y el pedido j.

La unión de todos los pedidos P_j que guarden la relación G_{ij}^3 con P_i constituyen el conjunto siguiente:

$$\{G_i^3\} = \bigcup_{j=1}^{E_1^{Fl}} \{G_{ij}^3\} \text{ o } \{G_i^3\} = \bigcup_{j=1}^{E_1^{N}} \{G_{ij}^3\} \text{ donde } i \neq j$$

Por último, la unión de todas las relaciones G_i^3para todo i perteneciente a $\{E_1^{Fl}\}$ *o* $\{E_1^{N}\}$, constituyen el grupo:

$$\{G^3\} = \bigcup_{i=1}^{E_1^{Fl}} \{G_i^3\} \text{ o } \{G^3\} = \bigcup_{i=1}^{E_1^{N}} \{G_i^3\}$$

4. *Distinto origen y distinto destino* ***$\{G^4\}$***

Se define $\{G_i^4\}$, como el conjunto de pedidos pertenecientes a $\{E_1^{Fl}\}$ *o* $\{E_1^{N}\}$, que tienen el mismo origen y diferente destino que el pedido $P_i(r_i, e_i)$. Es decir, sea $P_j(r_j, e_j)$ otro pedido perteneciente al mismo conjunto $\{E_1^{Fl}\}$ *o* $\{E_1^{N}\}$ que P_i, entonces:

Con la siguiente expresión se describe la relación distinto origen – distinto destino que se establece entre dos pedidos.

$$\{G_{ij}^{4}\} = \{(P_i \cup P_j) \leftrightarrow (r_i \neq r_j \,\&\, e_i \neq e_j)\ tq\ (r_i, e_i) \in P_i \,\&\, (r_j, e_j) \in P_j\}$$

A modo ilustrativo el significado de la expresión anterior se muestra en la figura 7.

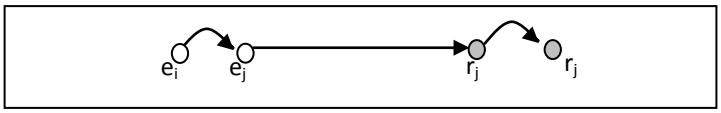

Figura 7. Relación $\{G^4\}$ entre el pedido i y el pedido j.

La unión de todos los pedidos P_j que guarden la relación G_{ij}^{4} con P_i constituyen el conjunto siguiente:

$$\{G_i^4\} = \bigcup_{j=1}^{E_1^{Fl}} \{G_{ij}^4\} \text{ o } \{G_i^4\} = \bigcup_{j=1}^{E_1^{N}} \{G_{ij}^4\} \text{ donde } i \neq j$$

Por último, la unión de todas las relaciones G_i^4para todo i perteneciente a *$\{E_1^{Fl}\}$ o $\{E_1^{N}\}$,* constituyen el grupo:

$$\{G^4\} = \bigcup_{i=1}^{E_1^{Fl}} \{G_i^4\} \text{ o } \{G^4\} = \bigcup_{i=1}^{E_1^{N}} \{G_i^4\}$$

En el caso *$\{G^2\}$, $\{G^3\}$ y $\{G^4\}$* estos conjuntos serán ordenados en orden creciente de distancias, que en el primer caso será la que existe entre los destinos, en el segundo entre los orígenes y en el tercero la suma de la distancia presente entre los orígenes y los destinos.

Inicialización del proceso (2)

Se ordena la lista de pedidos correspondientes a *$\{E_1^{Fl}\}$ o $\{E_1^{N}\}$* de mayor a menor distancia entre el origen del pedido y el destino del pedido.

Inicio bucle que controla el número de iteraciones (3)

Aquí se da comienzo a la búsqueda de la mejor solución para generar las rutas de un determinado día. Empieza con un bucle que termina cuando la variable k alcanza el valor del parámetro NIt. El valor de este parámetro es el que define el refinamiento de la solución, es decir, determina las veces que se va pulir el mejor resultado encontrado hasta el momento. Si este valor es muy pequeño, puede que la solución encontrada no sea buena, pero si el valor es muy elevado puede llegar a consumir demasiado tiempo.

Inicio bucle que controla el número de hormigas de la colonia (3.1)

Esta sentencia controla otro proceso iterativo a través del parámetro NH., cuando la variable t alcanza su valor, el bucle se detiene. El significado de NH se refiere al número de hormigas de la colonia. Cada una de las hormigas representa una solución del conjunto de posibles combinaciones calculada por el algoritmo "*construcción de la solución (3.1.1)*", es decir, las rutas a realizar en un día. Por lo tanto al fin de este bucle se tendrán NH hormigas o soluciones.

Construcción de la solución (3.1.1)

Se realiza la invocación del algoritmo del apartado 4.2 de este anexo. Como resultado de la ejecución de esta llamada se genera una solución, es decir, una planificación de rutas. Esta solución se corresponde con una de las NH hormigas del bucle.

Selección de la mejor solución de la iteración actual Ψib (t) (3.2)

En este paso se selecciona la mejor solución de las NH producidas por el bucle que invoca a construcción solución, según el valor de la función objetivo determinada en el apartado 4 del documento principal, es decir la que nos proporcione el menor coste.

Mejora local de esta solución Ψib* (t) (3.3)

Una vez seleccionada la mejor solución de la iteración t se realiza una mejora local sobre la misma, es decir, el algoritmo trata de optimizarla. Una mejora local para todas las rutas de la solución podría consumir excesivo tiempo de ejecución, por ello este proceso detecta los itinerarios que están prácticamente vacios e intenta enviar las cargas de cada uno a otras rutas alternativas. De esta forma, en caso de poder realizar algún agrupamiento se habrá conseguido reducir el coste al eliminar rutas y por tanto vehículos de la planificación.

Actualización de la mejor solución global Ψgb (t) (3.4)

Si la solución mejorada de la iteración t,Ψib* (t) es mas buena que la mejor encontrada hasta el momento Ψgb (t) entonces Ψgb (t) = Ψib* (t), sino se queda con la que tenía.

Actualización del valor de la feromona **(3.5)**

Después de que una colonia de hormigas de una iteración t haya generado la solución, se tiene que actualizar el valor de la feromona. Para ello se utilizan las formulas que se muestran a continuación:

$$\zeta_{ij}^{z}(t+1) = (1-\rho)\zeta_{ij}^{z}(t) + \zeta_{ij}^{best}(t), \forall(i,j), \forall z$$

$$\Delta\zeta_{ij}^{best}(t) = \begin{cases} \dfrac{1}{Coste(\psi^{b})} \\ \\ 0 \end{cases}$$

Donde $0 \leq \rho \leq 1$ y corresponde al ratio de evaporación de la feromona. Para aprovecharse de la mejor solución encontrada, solo se le permite añadir feromona a una hormiga tras cada iteración. Esta hormiga será la que provee de la mejor solución en la iteración t, es decir, Ψib (t) del punto (3.2). $\Delta\zeta_{ij}^{b}(t)$ es la cantidad de valor de feromona añadida por esta hormiga y Coste(Ψ^{b}) es el coste de Ψib (t).

Este es el algoritmo encargado de generar la solución, es decir la agrupación de pedidos en rutas. El esquema de funcionamiento se presenta en la figura 8, a continuación se explica de forma concisa el funcionamiento del mismo.

4.2) Construcción de la solución

Aquí se realiza la agrupación de pedidos de cada uno de los conjuntos, generando las rutas a seguir. El proceso seguido es el mostrado en la figura 8.

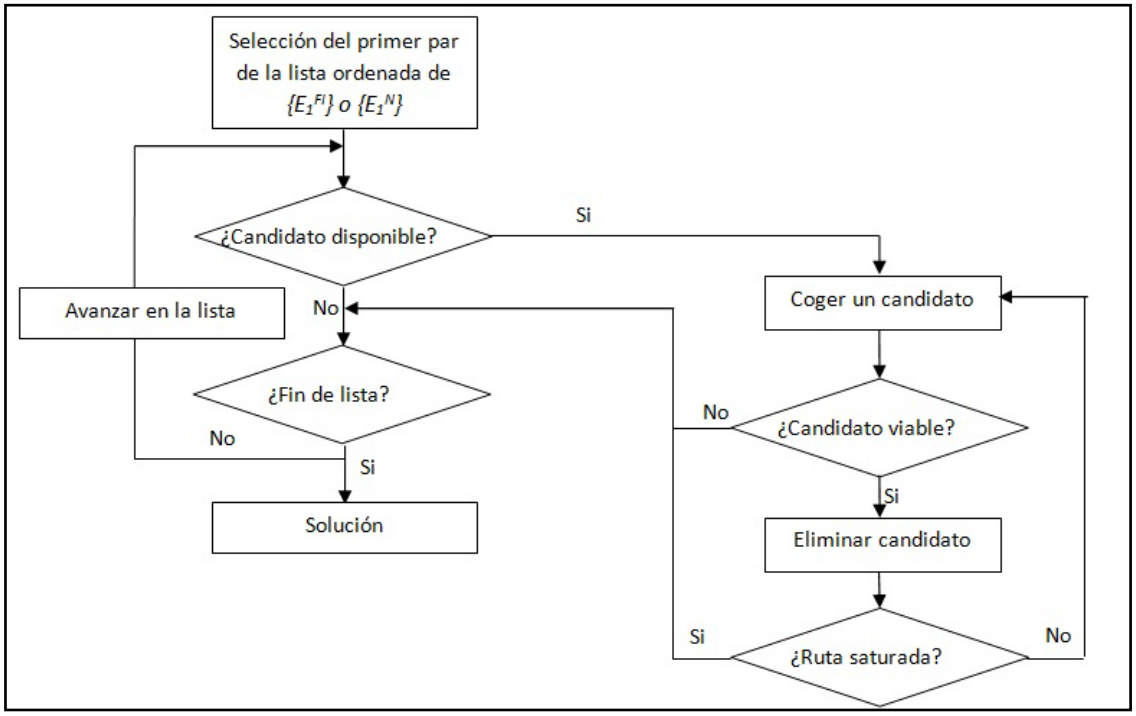

Figura 8. Construcción solución.

1. Se selecciona el primer pedido de lista ordenada, ya sea $\{E_1^{FI}\}$ o $\{E_1^{N}\}$.
2. Del conjunto de candidatos $\{G^1\}$, $\{G^2\}$, $\{G^3\}$ o $\{G^4\}$ correspondientes al pedido elegido, se selecciona el primero de cada lista en caso de existir. Si resulta más de uno la hormiga lo selecciona de forma probabilística según las siguientes formulas:

$$P_{ij}^{G1} = \frac{[\tau_{ij}^{G1}(t)]^{\alpha}[u_{ij}]^{\beta}}{\sum_{h=1}^{G1}[\tau_{ih}^{G1}(t)]^{\alpha}[u_{ih}]^{\beta} + \sum_{h=1}^{G2}[\tau_{ih}^{G2}(t)]^{\alpha}[u_{ih}]^{\beta} + \sum_{h=1}^{G3}[\tau_{ih}^{G3}(t)]^{\alpha}[u_{ih}]^{\beta} + \sum_{h=1}^{G4}[\tau_{ih}^{G4}(t)]^{\alpha}[u_{ih}]^{\beta}}$$

$$P_{ij}^{G2} = \frac{[\tau_{ij}^{G2}(t)]^{\alpha}[u_{ij}]^{\beta}}{\sum_{h=1}^{G1}[\tau_{ih}^{G1}(t)]^{\alpha}[u_{ih}]^{\beta} + \sum_{h=1}^{G2}[\tau_{ih}^{G2}(t)]^{\alpha}[u_{ih}]^{\beta} + \sum_{h=1}^{G3}[\tau_{ih}^{G3}(t)]^{\alpha}[u_{ih}]^{\beta} + \sum_{h=1}^{G4}[\tau_{ih}^{G4}(t)]^{\alpha}[u_{ih}]^{\beta}}$$

$$P_{ij}^{G3} = \frac{[\tau_{ij}^{G3}(t)]^{\alpha}[u_{ij}]^{\beta}}{\sum_{h=1}^{G1}[\tau_{ih}^{G1}(t)]^{\alpha}[u_{ih}]^{\beta} + \sum_{h=1}^{G2}[\tau_{ih}^{G2}(t)]^{\alpha}[u_{ih}]^{\beta} + \sum_{h=1}^{G3}[\tau_{ih}^{G3}(t)]^{\alpha}[u_{ih}]^{\beta} + \sum_{h=1}^{G4}[\tau_{ih}^{G4}(t)]^{\alpha}[u_{ih}]^{\beta}}$$

$$P_{ij}^{G4} = \frac{[\tau_{ij}^{G4}(t)]^{\alpha}[u_{ij}]^{\beta}}{\sum_{h=1}^{G1}[\tau_{ih}^{G1}(t)]^{\alpha}[u_{ih}]^{\beta} + \sum_{h=1}^{G2}[\tau_{ih}^{G2}(t)]^{\alpha}[u_{ih}]^{\beta} + \sum_{h=1}^{G3}[\tau_{ih}^{G3}(t)]^{\alpha}[u_{ih}]^{\beta} + \sum_{h=1}^{G4}[\tau_{ih}^{G4}(t)]^{\alpha}[u_{ih}]^{\beta}}$$

El valor de P_{ij}^{K} se refiere a la probabilidad de que una hormiga seleccione el pedido j ϵ k, donde k ϵ [G^1, G^2, G^3, G^4]. Dada esta regla probabilística la hormiga seleccionara aquella que posea el menor valor de $u_{ij.}$ Si se examina cada una de las formulas anteriores se tiene, el valor de ζ_{ij}^{t}, con t ϵ [1, 4], que se refiere al valor de la feromona y u_{ij}, que es el parámetro de información heurística. Estos parámetros deben ser inicializados antes de que el proceso iterativo comience. El rango de valores para ζ_{ij}^{K} será [$\zeta_{min,}$ ζ_{max}] y el que tomara al inicio será $\zeta_{max,}$ es decir, todos los ζ_{ij} del proceso serán inicializados con este valor, de forma que la primera vez que se ejecuta se cogen de forma aleatoria. El valor del parámetro heurístico u_{ij} se refiere a la suma de la distancia entre los orígenes de la ruta, los destinos de la ruta y la distancia entre el origen y el destino del último pedido insertado. Según los subconjuntos *{G^1}, {G^2}, {G^3} o {G^4}* las distancias que se pueden dar se muestran en la figura 9.

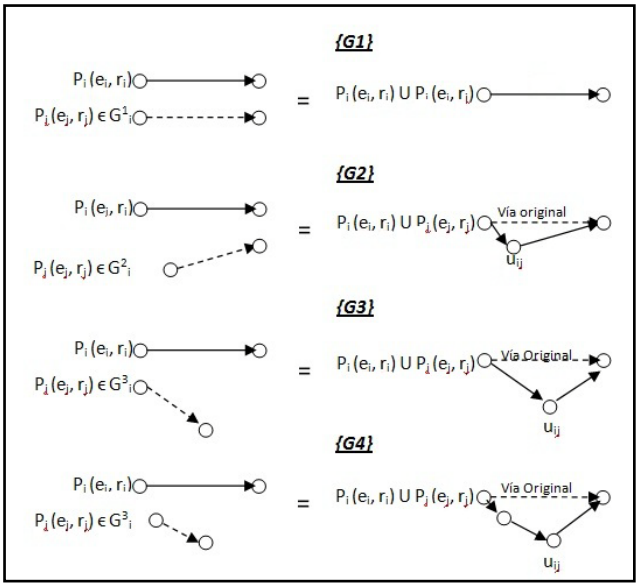

Figura 9 Distancia u_{ij} para cada candidato del pedido P_i.

Además se tienen los parámetros α y β que determinan la influencia en la determinación de $P_{ij,}$ donde α regula el valor de la feromona y β la del parámetro heurístico. Es decir, si α = 0, la hormiga seleccionara una estrategia *{G^1}, {G^2}, {G^3} o {G^4}* guiada exclusivamente por la información heurística, por otro lado si β= 0 entonces únicamente se considera el valor de la feromona, lo que puede ocasionar una rápida convergencia a una situación de estancamiento.

3. Se comprueba si el candidato seleccionado es viable, es decir, si la expresión (3) del valor de la función objetivo descrita en el apartado 4 del documento principal resulta viable, en caso afirmativo se sigue en el punto 4, en caso negativo se vuelve al punto 2.
4. Se elimina el candidato P_j de la lista inicial de ordenados y de todas las listas de candidatos existentes.
5. Se comprueba si la ruta R_k con los pedidos P_i y P_j está saturada, es decir, si la capacidad disponible del camión o el tiempo de ruta es demasiado largo y no permite insertar otro pedido. En caso afirmativo se regresa al punto 3 y en caso negativo se va al punto 6.

6. Se introduce la ruta en el conjunto de soluciones, se elimina el pedido P_i de la lista de ordenados y se comprueba si quedan más pedidos en esta lista. En caso afirmativo se vuelve al punto 1 con el siguiente pedido de la lista, en caso negativo se sigue en el punto

7. El proceso termina y se tiene la solución para la hormiga k.

Anexo VIII: Función objetivo y restricciones

1) Introducción

En este documento se explica la función objetivo y restricciones descritas en el apartado 4 del documento principal pero de una forma más sencilla de entender por el interesado. Para ello se explica cada restricción y cálculos oportunos mediante el pseudocódigo, que a su vez puede ser utilizado para la implementación de las mismas.

En primer lugar se describe la terminología utilizada para entender que significa cada componente de la función objetivo y las restricciones que la condicionan explicadas en este documento En segundo lugar, se expone la función objetivo y se explica su comportamiento. Por último, se detalla cómo se considera que una restricción, ya sea sobre la capacidad, la superficie, la disponibilidad o el tiempo califica a una solución como viable o inviable y el procedimiento para calcular el coste relativo a una operación individualmente.

2) Terminología y conceptos

Se describen los parámetros, variables y formulas utilizadas en la función objetivo y en la descripción de las restricciones.

S (k) = solución formada por k rutas.

R_i (a_j) = Ruta i formada por j pedidos.

R_k (a_j) = X_i, i ϵ [1,2j]. Operación i de la ruta k, donde **R_k (a_j) = $O_w D_z$,** w ϵ [1, 2j] y z ϵ [j, 1]. Es decir:

- X_i, puede ser una operación de recogida en el origen O_w o una operación de entrega en el destino D_z.
- Para asegurar la estructura LIFO se realizan las recogidas en orden creciente (del primer pedido hasta el último y las entregas en orden decreciente, desde el último pedido hasta el primero): $R_k(a_j)=X_1...X_{2j}= O_1 \dots O_j D_j \dots D1$.

F(S (k)) = coste de la solución formada por k rutas.

C (R_i (a_j)) = coste de una ruta formada por 2j operaciones o 2j X_k.

C' (X_i) = coste de realizar la operación X_i.

R = coste por unidad de distancia recorrida, (€/km).

P = coste fijo relativo a hacer una parada en un punto.

E = coste por unidad de tiempo que un vehículo permanece parado hasta que es servido (€/min).

LD = límite de tiempo de disco máximo que puede estar un vehículo en funcionamiento.

β_m = la operación X_i satisface las restricción m y puede ser servida.

α_1: comprueba que la localización de la operación X_i es distinta de la X_{i-1}.

α_2 = comprueba que la operación X_i ha tenido que esperar a ser servida.

v = Velocidad (km/h).

Operación (X_i) = {Entrega/Recogida}.

Peso (X_i) = Si **Operación (X_i) == Recogida** devuelve el peso de la mercancía a recoger, **p_i.**

PesoAcumulado (X_i) = $\sum_{j=1}^{i} Peso(x_j)$

CargaMaximaVehiculo = Q.

CapacidadDisponible (R_k, X_i) = CargaMaximaVehiculo - PesoAcumulado (X_i).

DimensionesVehiculo = {L, W}; **SuperficeVehiculo =** L x W

Ítems (X_i) = Si **Operación (X_i) == Recogida** devuelve el conjunto de ítems a recoger: (IT_1,… IT_m), donde m es el número de ítems a recoger.

Dimensión (Xi, IT_j) = Devuelve las dimensiones del pedido **IT_j** del cliente **X_i**: (l_{ij}, w_{ij}).

SuperficieNecesaria(Xi)= $\sum_{j=1}^{items(X_i)} (\text{Dimension}(X_i, IT_j).l) x (\text{Dimension}(X_{i,} IT_j).w) = \sum_{j=1}^{items(X_i)} l_{ij} x w_{ij}$

VT_i = ventana temporal correspondiente a la operación X_i.

VT_i= [Imin_X_i, Imax_X_i].

- **Imin_X_i** = cota inferior del intervalo temporal.
- **Imax_X_i** = cota superior del intervalo temporal.

Hora_Salida_0 (X_i) = Hora Llegada (X_{i-1}) + TiempoDesplazamiento ($X_{i-1,}$ X_i) o

Hora_Salida_0 (X_i) = Imin_X_i.

Hora_Salida (X_i) = Hora_Salida_0 (X_i) + α_1xP.

Hora_Llegada (X_i) = Hora_Salida (X_{i-1}) + TiempoDesplazamiento (X_{i-1}, X_i).

TiempoDesplazamiento (X_{i-1}, X_i) = Distancia (X_{i-1}, X_i)/v.

TiempoEspera (X_i) = α_2 x (Imin_X_i - Hora_Llegada (X_i)).

TiempoOperacion (X_i) = α_1xP.

TiempoDescanso=TD, tiempo de descanso a satisfacer por el vehículo.

TiempoDiscoAcumulado (Ri)=Tiempo de conducción acumulado por el vehículo.

Distancia (X_{i-1}, X_i) = distancia real entre la Localización (X_{i-1}) y la Localización (X_i).

Localización (X_i) = (Latitud (X_i), Longitud (X_i)).

3) Función objetivo

Se presenta la función de costes que es la que determina la calidad de la solución y las restricciones que valoran la viabilidad.

Coste de la solución.

$$F(S_{(k)}) = \sum_{i=1}^{k} C(R_i(a_j)) \qquad (1)$$

Es decir, una solución está constituida por un conjunto de k rutas, por tanto el valor de la función de costes será la suma del coste de cada una de las rutas R_i individualmente. A su vez cada ruta está constituida por j pedidos, representados en la formula anterior por a_j., donde j ≥ 1.

Coste de una ruta.

$$C(R_i(aj)) = \sum_{i=1}^{2j} \prod_{K=1}^{r} \beta_k \times C'(X_i) \qquad (2)$$

Un pedido a_j, está constituido por dos operaciones (una recogida en el origen y una entrega en el destino), por tanto el coste de una ruta implica la suma del coste de 2j operaciones. El coste de una operación se representa con *C' (X_i).*

β_k, corresponde a la restricción que se está considerando; el valor que puede tomar es cero o uno dependiendo de si la inserción de la operación X_i, en la ruta R_i se considera viable o inviable. Una operación se considera factible si satisface todas las restricciones, por ello si existe una cuyo valor resulta cero (no valida) el producto de todas ellas es cero y convierte a la solución en irrealizable independientemente de que se satisfagan el resto de restricciones.

Coste de una operación de la ruta:

$$C'(X_i) = \alpha_1 \times distancia(X_{i-1}, X_i) \times R + \alpha_1 \times P + \alpha_2 \times TiempoEspera(X_i) \times E \qquad (3)$$

El coste de una ruta será la suma de los costes de todas las operaciones a realizar en la ruta (entregas, recogidas) representadas por X_i. El primer termino de la suma corresponde al cálculo de la distancia entre el punto de la operación X_{i-1} y X_i, el segundo termino corresponde a un coste fijo implicado en toda localización. Ambos sumandos están multiplicados por un parámetro α_1, este parámetro tomará el valor cero o uno dependiendo de la relación entre la localización de las operaciones. El tercer sumando se refiere al cálculo del coste que implica realizar una espera, que se producirá si y sólo si el vehículo llega antes de la cota inferior de la

ventana temporal de la operación vigente X_i, concepto que se parametriza a través de α_2 tomando el valor cero o uno.

A continuación se explican las restricciones a las que está sujeta la función de costes y que están representadas con los parámetros α y *β.*

El coeficiente *β,* determina si la operación puede ser insertada en la ruta. Existen cuatro restricciones a satisfacer, por tanto se obtienen cuatro parámetros β_k, que son listados en primer lugar y después se explica el significado y como se ha determinado cada uno de ellos.

Cabe comentar que toda expresión que no se explica con detalle está desarrollada en el punto 2 de este anexo.

4) Restricciones

Se tienen cuatro restricciones que delimitan si una operación es viable. Estas restricciones se representan en la función objetivo con el coeficiente β_k. Dependiendo del valor de k, se tiene una restricción u otra.

4.1) Restricción de capacidad

Si es β_k=$\boldsymbol{\beta_1}$ *se refiere a la* ***restricción de capacidad.***

- $\beta_1 = 1$. La operación puede ser servida y se continúa con la exploración del resto de operaciones de los pedidos, *(los X_i).*
- $\beta_1 = 0$. La operación no puede ser servida y se descarta la ruta, es decir, el último X_i, no puede ser combinado con la ruta y se descarta la combinación.

Para obtener el valor se comprueba si la capacidad disponible del vehículo supera a los kilos del pedido del tipo recogida en el punto i, tal y como se muestra en la figura 1.

Comprobar valor de β_1:
Si *(CapacidadDisponible(R_k, X_{i-1}) > Peso(X_i))* **entonces** $\beta_1 = 1$
Si *(CapacidadDisponible(R_k, X_{i-1}) < Peso(X_i))* **entonces** $\beta_1 = 0$

Figura 1. Evaluación de la restricción de capacidad.

4.2) Restricción de superficie

Si es β_k=$\boldsymbol{\beta_2}$ *se refiere a la* ***restricción de superficie.***

- $\beta_2 = 1$. La operación puede ser servida y se continúa con la exploración del resto de operaciones de los pedidos, *(los X_i).*
- $\beta_2 = 0$. La operación no puede ser servida y se descarta la ruta, es decir, el último X_i, no puede ser combinado con la ruta y se descarta la combinación.

Para acreditar la superficie disponible, se realiza una aproximación ya que quedan huecos entre los paquetes. Es decir, se comprueba la situación en el mejor de los casos, que el suelo del vehículo sea ocupado en su totalidad. Para obtener el valor se realizan los cálculos de la figura 2.

$$\boldsymbol{SuperficieDisponible\ (X_i)} = SuperficeVehiculo - \sum_{j=1}^{i} SuperficieNecesaria(X_j)$$

Comprobar valor de β_2:

Si *(SuperficieDisponible(X_i) ≥ SuperficieNecesaria(X_i))* **entonces** $\beta_2 = 1$

Si *(SuperficieDisponible(X_i) < SuperficieNecesaria(X_i))* **entonces** $\beta_2 = 0$

Figura 2. Evaluación de la restricción de superficie.

4.3) Restricción de disponibilidad

Dependiendo del resultado de la ejecución del algoritmo de llenado de camiones descrito en el apartado 3.5 del documento principal, se determina el valor de la restricción de disponibilidad tal y como se muestra en la figura 3. Para representar el algoritmo en esta sección se denomina EncuentraColocación(X_i).

Comprobar valor de β_3:

Si *(EncuentraColocación (X_i))* **entonces** $\beta_3 = 1$

Si *(not (EncuentraColocación (X_i)))* **entonces** $\beta_3 = 0$

Figura 3. Evaluación de la restricción de disponibilidad.

4.4) Restricción temporal

Controlar que la operación es servida dentro de la ventana temporal. Al mismo tiempo se determina el valor del parámetro $\boldsymbol{\alpha_2}$, es decir, si ha llegado antes de la cota inferior de la ventana temporal. La expresión que representa el cálculo de los tiempos es expuesta a en la figura 4.

Comprobar valor de β_4:

Si *(HoraLlegada(X_i) ϵ VT_i)* **entonces**

β_4= 1

HoraSalida_0(X_i) = HoraLlegada(X_i) + TiempoDesplazamiento(X_{i-1},X_i)

α_2=0

Si *(HoraLlegada(X_i) <Imin_X_i)* **entonces**

β_4= 1

HoraSalida_0(X_i) = Imin_X_i

α_2=1

TiempoEspera(X_i)= Imin_X_i – HoraLlegada(X_i)

Si *(HoraLlegada(X_i) >Imax_X_i)* **entonces**

β_4= 0

Además se debe comprobar que no se ha excedido el tiempo de disco:

Si *(TiempoDiscoAcumulado(Ri)<LD)* **entonces** β_4= *1*

Si *(TiempoDiscoAcumulado(Ri)>LD)* **entonces** β_4 = *0*

Figura 4. Evaluación de la restricción temporal.

4.5) Cálculo de los tiempo de una operación individual (entrega/ recogida)

Para determinar si una operación es viable hay que tener pleno control en el tiempo consumido hasta el momento y el tiempo que conlleva esta operación para actualizarlo. En las figuras 5 y 6 se muestra como se calcula cada uno de estos tiempos y el valor de **α_1.**

Si *(Localización(X_{i-1}) ==Localización (X_i))* **entonces**

TiempoDesplazamiento(X_{i-1}, X_i) =0

TiempoOperacion(Xi) = 0

α_1=0

Si *(Localización(X_{i-1})!=Localización(X_i))* **entonces**

TiempoDesplazamiento(X_{i-1}, X_i) =distancia(X_{i-1},X_i)/v

TiempoOperacion(Xi) = P

α_1=1

Figura 5. Cálculo del tiempo de desplazamiento, de operación y viabilidad de la operación.

Se tiene que actualizar el valor de *TiempoDiscoAcumulado (Ri),* tal y como se muestra en la figura 6.

Si (TiempoDiscoAcumulado (Ri)>LD*)* **entonces**

(*) *TiempoDiscoAcumulado (Ri)=0.*

(**) *TiempoDesplazamiento (X_{i-1}, X_i)= TiempoDesplazamiento (X_{i-1}, X_i) + TD.*

Sino *(cualquier otro caso)*

TiempoDiscoAcumulado(Ri)=TiempoDiscoAcumulado(Ri)+TiempoDesplazamiento(X_{i-1},X_i)

Figura 6. Actualización del valor del tiempo de disco acumulado.

Es decir, si el tiempo de disco acumulado ha superado el limite el camión debe parar un tiempo TD. En este momento el contador del disco se reinicia y al tiempo de desplazamiento que conlleva la realización de la ruta hay que sumarle el tiempo de descanso TD.

En caso de considerar, la limitación del disco de conducción se procedería de la misma manera, pero se consideraría el limite de tiempo de conducción y el mínimo tiempo de descanso TC.

Anexo IX: Estructuras de datos

1) Introducción

En este documento se explican algunos de los datos del problema, la información entrante o inputs, los resultados u outputs y elementos intermedios que facilitan el cálculo de la solución. Además se incluye una descripción de la estructura de datos que los representa.

2) Información Entrante o Inputs

Los inputs son los datos que alimentan al algoritmo descrito en el apartado 3 del documento principal. A partir de esta información se calcula la solución.

2.1) Descripción de los inputs

El dato que alimenta al algoritmo es el conjunto de pedidos realizados en un día. Antes de describir un pedido es necesario explicar los siguientes conceptos:

Ventana temporal [a, b]: Intervalo de tiempo en el que realizar una operación (entrega, recogida), donde a es la cota inferior y b el límite superior.

Tipo de mercancía: Los artículos a transportar pertenecen a una determinada clase, que exige un tratamiento especial dependiendo del conjunto al que corresponda. La clasificación que se puede realizar es {F, I, N}, es decir, frio, isotermo y normal respectivamente.

Ítem: Paquete correspondiente a una carga, está caracterizado por la superficie (m^2) y el peso (kg).

Carga: Son el conjunto de ítems a transportar correspondientes a una orden. Todos los paquetes pertenecerán al mismo tipo de mercancía, la suma de los pesos de todos los ítems generan el peso de la carga y la suma de la superficie el área total de la misma.

Punto o nodo: Localización en una posición geográfica de una operación, ya sea entrega o recogida. Cada uno de ellos está caracterizado por una ventana temporal en la que realizar el servicio.

Pedido: El pedido es una orden realizada por un cliente y se caracteriza por estar formado por dos puntos, el origen donde se realiza una recogida y un destino donde realizar la entrega y poseerá una carga a transportar.

2.2) Estructura de los inputs

tpVentanaTemporal
hora_inicio: hh:mm hora_fin: hh:mm
hora_inicio? hora_fin?

tpCarga
Ítems: {item1,…itemn} número: n; peso: sum(peso(ítem(i))) i en [1, n] superficie: sum(superficie(ítem(i))) i en [1, n] tipo_mercancía:{F} o {I} o {M}
items? número? superficie? tipo_mercancía? peso?

tpItem
l: cm w: cm tipo_mercancía:{F} o {I} o {M} peso: kg
longitud? anchura? superficie? tipo_mercancía? peso?

tpPunto
posición_x: real; posición_y: real; VT: tpVentanaTemporal;
posición_x? posición_y? VT_inferior? VT_superior?

Pedido
identificador: entero; origen: tpPunto; destino: tpPunto; carga: carga;
identificador? origen? destino? carga?

3) Información saliente u Outputs

Los outputs son los resultados a devolver por el algoritmo.

3.1) Descripción de los outputs

El resultado del algoritmo es el conjunto de rutas a realizar en un día. Para ello es necesario definir en primer lugar el concepto de ruta.

Ruta: Se define como ruta(k) el conjunto de pedidos a realizar por el vehículo (k). La tipología de la mercancía de los pedidos correspondientes a ruta(k), caracteriza el tipo de vehículo(k).

Con el objetivo de reducir el espacio de memoria necesario para implementar este dato, una ruta estará constituida por una estructura dinámica. Tendrá un puntero al siguiente pedido a recorrer y otro puntero al anterior, de forma que se asegura la estructura Lifo. Es decir, si ruta(k) = P(1)P(2)P(3) el puntero en orden ascendente avanza hasta llegar al último pedido de forma que se comprueba si los orígenes de los pedidos analizados en el siguiente orden (r(1), r(2), r(3)) satisfacen las restricciones de la función objetivo. Al llegar al último pedido se recorre la lista en orden descendente hasta llegar al primer pedido comprobando que los destinos analizados en el siguiente orden (e(3), e(2), e(1)) satisfacen las restricciones.

Solución: Conjunto de rutas. Con el objetivo de ahorrar memoria se define como una estructura dinámica, en este caso como una lista.

3.2) Estructura de los outputs

tpRuta
i Identificador: entero; pedido:tpPedido; siguiente: puntero tpPedido; anterior: puntero tpPedido;
identificador? pedido? siguiente? anterior?

Solución
identificador: entero; ruta: tpRuta; siguiente: puntero tpRuta;
identificador? pedido? siguiente?

4) Información intermedia

Los datos más relevantes de los necesarios para realizar la búsqueda de la solución óptima son muchos, por ello, en este caso sólo se ha definido el que se considera más importante, que es la matriz de distancias entre todos los puntos del sistema.

4.1) Descripción de la matriz de distancias

Matriz de distancias: matriz NxN conteniendo las distancias reales entre cada par de nodos. Es asimétrica, ya que distancia de i a j no es igual a la distancia de j a i.

4.2) Estructura de la matriz de distancias

Matriz Distancias
matriz: NxN;
distancia (i,j)?

Referencias Bibliográficas:

[1] Estudio y descripción del problema:

Francisco Aparicio Izquierdo: *INGENIERÍA DEL TRANSPORTE*, Editoriales Dossat, 2008

Juan de Dios Ortúzar: *MODELOS DE TRANSPORTE*, Universidad de Cantabria, 2008

[2] Estado del arte:

Emmanouil E. Zachariadis, Christos D. Tarantilis, Christos T. Kiranoudis: *A GUIDED TABÚ SEARCH FOR THE VEHICLE ROUTING PROBLEM WITH TWO DIMENSIONAL LOADING CONSTRAINTS,* European Journal of Operational research, 2007

L. Barcos, V. Rodríguez, M. J. Álvarez, F. Robusté: *ROUTING DESIGN FOR LESS THAN TRUCKLOAD MOTOR CARRIERS USING ANT COLONY OPTIMIZATION,* Tansportation Research Part E available at SceinceDirect, 2009

David Escuin Finol: *DESARROLLO DE UN ALGORITMO BASADO EN TECNICAS HEURISTICAS PARA LA RESOLUCION DE PROBLEMAS DE GESTION DE TRANSPORTE,* Universidad de Zaragoza, 2010

Francisco Baptista Pereira, Jorge Tavares: *BIO INSPIRED ALGORITHMS FOR THE VEHICLE ROUTING PROBLEM,* Springer, 2009

Mitsuo Gen, Runwei Cheng: *GENETIC ALGORITHMS AND ENGINEERING OPTIMIZATION,* John Wiley and sons, 2000

[3] Anexo NP-Completo:

http://www.uv.es/jkliment/Documentos/TeoComp.pc.pdf
http://www.ucm.es/info/pslogica/automatas.pdf

[4] Anexo VRP:

http://redalyc.uaemex.mx/redalyc/pdf/1492/149212815002.pdf
http://www.liacs.nl/assets/Masterscripties/2010-01FrankTakes.pdf
http://neo.lcc.uma.es/radi-aeb/WebVRP/

[5] Anexo BPP:

http://catarina.udlap.mx/u_dl_a/tales/documentos/lmnf/bonilla_g_le/capitulo2.pdf
http://www.mecalux.es/navigation/event/Detailinterview.do?idinterview=28623096

[6] Anexo Metaheurísticas:

http://redalyc.uaemex.mx/redalyc/pdf/1492/149212815002.pdf
www.ayc.unavarra.es/orduna/LIA/Teoria/juegos.ppt
http://www.iiia.csic.es/~pedro/busqueda1-introduccion.pdf
http://redalyc.uaemex.mx/redalyc/pdf/1492/149212815002.pdf

[7] Anexo colonia de hormigas:

Francisco Baptista Pereira, Jorge Tavares: *BIO INSPIRED ALGORITHMS FOR THE VEHICLE ROUTING PROBLEM,* Springer, 2009.