Universidad de Valladolid
Facultad de Ciencias

TRABAJO DE FIN DE GRADO

Grado en Estadística

Problema del Mínimo Árbol con Capacidades: Modelización y resolución Heurística

---

Alumna: María Padrones Rengel

Tutores: Jesús Sáez Aguado
Ricardo Josa Fombellida

Agradecimientos

Agradecer expresamente a Jesús Saéz.

A mi familia, en especial a Isa, a los amigos que lo eran y a los que en esta carrera he encontrado.

Resumen

Es evidente que vivimos en un mundo conectado. Podemos encontrar sistemas de conexión a mayor o menor escala en, prácticamente, cualquier ámbito en el que se nos ocurra pensar: desde infraestructuras de carácter civil como carreteras, hasta en el terreno personal como es el caso de las redes sociales. Dada la utilidad y la importancia de unir elementos que, por determinada razón, interesa que estén relacionados, es necesario dedicar esfuerzo a encontrar la mejor forma de hacerlo.

En este trabajo de fin de carrera se aborda el problema del mínimo árbol expandido con capacidades conocido como CMST. A lo largo de estas páginas se aporta información sobre las características y la implementación de estas estructuras, y se exponen distintos métodos exactos y heurísticos (inluyendo métodos de mejor y metaheurísticas) de enfrentar el problema. Todos los modelos se han aplicado sobre varios conjuntos de datos para poder comparar y sacar conclusiones sobre el desempeño de cada uno.

Resumen

It is evident that we live in a connected world. Connecting systems can be found in such different fields like road infrastructures or social networks. Due to how important and useful connecting up several elements can be, it is necessary to design carefully a way to do it.

This paper is an approach to the capacitated minimum spanning tree problem. In the following pages some characteristics will be provided as well as a couple of ways to implement the solution trees. Also, exact and heuristic methods (including enhancements and metaheuristics) to solve the CMST problem are exposed. These methods have been applied to several different data files in order to compare their performance.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Vivimos en un mundo hiper relacionado en el que, prácticamente todo lo que nos rodea, funciona gracias a conexiones. Estas conexiones forman redes de cuyo uso nos beneficiamos a diario en aspectos tan cotidianos como pueden ser hacer la compra, conectarse a internet, rellenar el depósito en cualquier gasolinera, usar el transporte público, ambulancias, hospitales y un largo etcétera. Como se puede observar, el uso de esta técnica es bastante generalizado, por lo que conviene dedicar los recursos necesarios a elaborar un diseño de conexiones adecuado.

El término red define a una estructura que cuenta con un patrón característico de elementos y uniones entre ellos. Existen muchos tipos de redes que se pueden clasificar en base a diversos criterios como pueden ser: el área de uso (red informática, red social, red eléctrica...), su infraestructura (PAN, LAN, MAN, WAN, WLAN), la relación funcional de los elementos que la forman (redes cliente/servidor, peer-to-peer o p2p) o su topología, que, en este caso, es la clasificación que nos interesa.
Se conoce como topología a la rama de las matemáticas que estudia las propiedades de los objetos más básicas que las geométricas, aquellas que permanecen invariantes cuando estos son plegados, dilatados, contraídos o deformados. La topología se aplica en diferentes campos, entre ellos, la teoria de grafos, en la que se considera que la naturaleza geométrica de las uniones entre los vértices que forman un grafo no tiene importancia, solo se tiene en cuenta la manera en la que dichos vértices están conectados. De esta forma, la topología de una red indica la forma en que está diseñada, cómo está conectado el conjunto de nodos que la forma. A continuación se detallan distintos tipos de redes según su topología y algunas ventajas y desventajas de los diseños [2]:

Estrella Todos los elementos de la red están unidos a un único nodo (concentrador) que transmite la información a todos los puntos conectados con él.
Ventajas: El concentrador controla el tráfico de la red y evita que se produzcan colisiones. Si por alguna razón se interrumpe la conexión de alguno de los nodos con el concentrador, el resto de la red sigue funcionando.
Desventajas: El coste de conexión es alto. La información se transmite a todos los elementos de la red aunque sólo sea relevante para uno. Si el concentrador falla deja de funcinar toda la red.

Anillo En este tipo de red todos los nodos están unidos unos a otros mediante un cable común formando un anillo. La información circula en un único sentido. Se puede implementar un doble anillo de forma que cada aro que conecta los nodos transmite el flujo en un sentido.

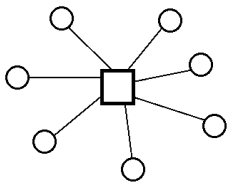

Figura 1.1: Red con topología en estrella

Ventajas: Es una arquitectura sólida pero con facilidad en la fluidez de la información. Se mantiene el rendimiento aunque aumente el número de puntos de la red. Acceso equitativo para todos los nodos.
Desventajas: Debido a cómo están establecidas las conexiones, si una deja de funcionar, se cae la red completa. Transmisión lenta debido a la longitud de los canales y al único sentido de tránsito. Problemas dificiles de idetificar y reparar. En redes de computadoras, problemas de privacidad ya que la información pasa por todos los nodos intermedios hasta que llega al receptor.

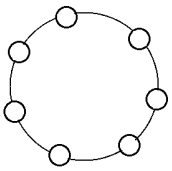

Figura 1.2: Red con topología en anillo

Bus Todas los nodos están conectados por un segmento de cable (bus). Cada nodo transmite información al bus esperando que no colisione con la información que hayan trasmitido otros nodos. Si ocurre alguna colisión, cada nodo espera un tiempo aleatorio para volver a transmitir.
Ventajas: Facilidad de implementación y de crecimiento. Arquitectura simple.
Desventajas: Es difícil detectar fallos, de haberlos, y reconfigurarla. Un problema en el bus afecta a toda la red. El desempeño disminuye cuando crece la red- puede haber un límite en los nodos que se puedaen conectar.

Jerárquica o arbórea En estas redes los nodos están conectados en forma de árbol. Se aprecia una estructura jerárquica en la que un nodo funciona como enlace o raíz a partir del que se ramifican el resto de conexiones. Se pueden ver como una serie de redes de estrellas interconectadas o como una variación de las redes tipo bus.
Ventajas: Cableado punto a punto entre dos nodos. El fallo en una conexión no implica la caída de toda la red. Facilidad de resolución de problemas lo que hace que sea un formato muy utilizado por desarrolladores.

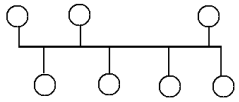

Figura 1.3: Red con topología en bus

Desventajas: Requiere grandes cantidades de cableado. Difícil configuración. Si un segmento falla, pierden también la conexión todos los nodos unidos a él por debajo en la jerarquía.

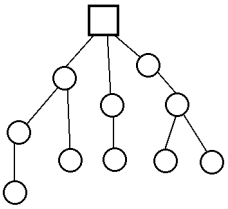

Figura 1.4: Red con topología jerárquica

Malla En este tipo de redes cada nodo está, de alguna forma directa o no, conectado con a todos los demás nodos de forma que la información se puede transmitir de un punto a otro por diversos caminos. Si la red está completamente conectada (todos los nodos unidos entre sí) no se pueden interrumpir las comunicaciones.
Ventajas: El tráfico se puede transmitir por distintos caminos de forma que es difícil que se interrumpan las conexiones- si un camino está ocupado, se usa otro. La desaparición de un nodo y por tanto, las conexines que dependían de él, no afecta a la conectividad de la red.
Desventajas: Esta red está pensada para implementarse de forma inalámbrica, si no tiene un coste muy alto.

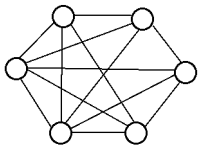

Figura 1.5: Red con topología en malla

Híbridas También se pueden crear redes que combinen tipos de los vistos como son los casos de: anillo en estrella, bus en estrella, estrella jerárquica, etc.

En realidad, la forma por sí misma no es lo más importante, a la hora de resolver cualquier problema de conexiones hay que tener en cuenta la situación de cada punto de la red respecto al resto, el orden en el que se van a unir...Una red no está definida solo por su topología o su infraestructura si no por todas sus características, por ejemplo: las redes en bus suelen implementarse en áreas no muy grandes y las de malla se suelen usar de forma WAN o WLAN, una red en bus o en anillo tienen una relación de igualdad peer-to-peer mientras que una red jerárquica o en estrella usan el modelo cliente/servidor. También se pueden implementar redes con varios concentradores (servidores) y hacer otras modificaciones de manera que se adapten a las necesidades que se tratan de cubrir.

Se puede decir, por lo tanto, que no existe un único diseño correcto y que el establecimiento de una red depende de muchos factores a tener en cuenta. Muchas veces estos factores entran en conflicto y se ha de fijar un objetivo principal (eficiencia, velocidad, mínimo coste, mínima distancia...) que alcanzar en detrimento de otros que también pueden ser importantes y/o válidos.

1.1. Objetivos y estructura

En este trabajo se aportan diferentes procedimientos, tanto exactos como heurísticos, para, dado un conjunto de nodos, encontrar la mejor manera de unirlos siguiendo una estructura de árbol y minimizando la distancia a recorrer para que todos los puntos estén conectados. Además entre los requisitos a cumplir se encuentra el que no se supere una capacidad fijada, lo que convierte el trabajo en la exposición y análisis de distintas técnicas de búsqueda de un árbol mínimo expandido con capacidades para cada conjunto de datos.

Los archivo de datos que se van a usar se han obtenido de la ORLib [1] y tienen las siguientes características:

- Hay archivos con problemas de 40 nodos, más el raíz 41, y archivos de 80 nodos, 81 con el base.
- Todos los archivos se van a resolver con tres capacidades límites distintas. En el caso de n=40 Q=3, 5, 10 y si n=80 la capacidad fijada puede ser Q=5, 10, 20.
- La posición que ocupa el nodo base o raíz cambia. En la mitad de los problema la raíz se sitúa en en un extremo del conjunto de nodos (end-point) y en la otra mitad se toma como raíz el nodo que ocupa una posición central (central-point).
- Se han seleccionado 3 archivos de cada tamaño para cada posición del nodo raíz lo que, con estas especificaciones, da lugar a un total de 36 problemas diferentes.

La estructura que va seguir el trabajo en los siguientes capítulos es:

- **Capítulo 2. Árboles. Complejidad del problema** Se aportan distintas nociones sobre árboles, varientes de estos y formas de implementarlos para trabajar con ellos. Breve encuadre de la Teoría de la Complejidad como parte de la Teoría de la Computación y clasificación del problema CMST dentro de ella.

[1] http://people.brunel.ac.uk/ mastjjb/jeb/orlib/capmstinfo.html

- **Capítulo 3. CMST. Formulaciones y modelos:** Se explica más detalladamente en qué consiste el problema del CMST y sus posibles variantes. Se exponen y formulan diferentes modelos exactos válidos para solucionar el problema que se está tratando.
- **Capítulo 4. Heurísticas base:** Se introduce el concepto de heurística. Se explican las técnicas de construcción (Prim modificado) y ahorros (Esau Williams y Esau Williams parametrizado).
- **Capítulo 5. Heurísticas de mejora:** En este capítulo se habla de distintas heurísticas de mejora, los distitos movimientos para alterar la solución inicial y se detallan los dos algoritmos de mejora empleados.
- **Capítulo 6. Metaheurísticas:** Se define el concepto de metaheurística y se exponen dos técnicas conocidas: Simulated Annealing y GRASP. Formas de aleatorizar el algoritmo que aporte la solución inicial.
- **Capítulo 7. Resultadoss:** Presentación de los resultados obtenidos aplicando las diferentes técnicas previamente explicadas sobre una serie de archivos de datos obtenidos de ORLib.
- **Capítulo 8. Conclusiones:** Comentarios sobre los distintos modos en los que se ha resuelto el problema y valoración comparativa de su desempeño.

Capítulo 2

Árboles. Complejidad del problema

En este caso, las características del problema que se trata de resolver nos llevan a plantear una solución con topología en árbol. Las configuraciones de este tipo están formadas por ramificaciones que convergen en una ráiz, asemejándose a la estructura de la que reciben el nombre. El concepto de árbol puede definirse desde distintos enfoques:

- *Tipo Abstracto de Datos (TAD).* En los campos relacionados con la informática y las ciencias de la computación, un árbol es una estructura jerárquica de datos. Se suele definir recursivamente como una colección de nodos que parten de un nodo raíz. Cada nodo es una estructura de datos en sí mismo y tiene una lista de referencias a los nodos hijos conectados con él. Para respetar la forma de árbol ha de ser acíclica, ninguna de las referencias debe estar duplicada (a cada nodo solo se puede acceder siguiendo una ruta única) y ninguna puede apuntar a la ráiz.

- *Grafo.* Matemáticamente, según la teoría de grafos un árbol se define como un grafo simple, conexo y acíclico que consiste en una estructura jerárquica basada en un nodo raíz al que están conectados un conjunto de subárboles. *G=(V,A)* donde *V=1,...,n+1* es el conjunto no vacío de nodos (incluyendo la base o raíz) que lo forman y *A* las aristas que los unen, de forma que un par de vértices están conectados por, exactamente, una secuencia de aristas.

Ambas perspectivas son útiles ya que, por ejemplo, es más intuitivo analizar un grafo, pero la representación como TAD facilita trabajar con ellos.

2.1. Terminología

En esta sección se van a incluir diferentes definiciones relacionadas con estas estructuras. Elementos constitutivos de un árbol:

- Nodos o vértices: Son los elementos de la red, puntos de conexión de aristas, que funcionan como registro de datos de interés. En el caso de los árboles, cada nodo está únicamente conectado según relaciones de jerarquía directa: con su ascendente inmediato o padre, y

con sus descendentes inmediatos o hijos, de forma que hay un único camino para llegar a cada nodo.

- Nodo raíz (o raíz): Es el nodo a partir del que se desarrolla el árbol.

- Nodo padre: Se conoce como nodo padre a aquel que conecta a un nodo particular con el nivel inmediatamente superior. Cada nodo tiene un único padre, salvo el raíz que únicamente puede tener nodos hijos.

- Nodo hijo: Se llaman hijos de un nodo a aquellos del nivel inmediatamente inferior conectados a este.Los nodos conectados a un mismo padre se llaman nodos hermanos.

- Nodo hoja, externo o terminal: Este término se refiere a los nodos finales del árbol en cuestión que no tienen conexiones con niveles inferiores, es decir, no tienen hijos. Los nodos que sí tienen descendencia se conocen como nodos rama o internos.

- Aristas: Ligaduras que unen los distintos vértices.

Otros conceptos que pueda resultar de interés:

- Nivel: El concepto de nivel en árboles es similar al de generación. El nivel de un nodo es el número de aristas que hay entre él y la raíz, teniendo en cuenta que el nodo raíz tiene nivel 0.

- Grado o factor de ramificación :

  -De un nodo: Indica el número de subárboles que se pueden forman con ese nodo como raíz, es decir, el número de hijos del nodo.

  -De un árbol:Es el grado máximo de los nodos del árbol, esto es, de los nodos del árbol el que tiene el mayor número de hijos.

- Descendientes: Los descendientes de un nodo son todos aquellos que están en niveles inferiores a los que se puede llegar partiendo de este.

- Antecesores: Son todos aquellos nodos de niveles superiores que forman la ruta para conectar el nodo en cuestión con la raíz.

- Subárbol: Subconjunto de nodos contenido en el árbol principal con raíz uno de los nodos de este.

- Camino: Ruta que conecta dos nodos. La longitud de camino de un árbol es la suma de las longitudes de los caminos a todos sus nodos.

- Altura:

  -De un nodo: Es el número de aristas del camino que une ese nodo y una hoja de sus descendientes más lejanos.La raíz tiene una altura igual al número de niveles del árbol. Los nodos hoja tienen altura cero.

  -De un árbol: Es la altura de su nodo raíz.

- Profundidad. Está relacionada con el concepto de altura.

  -De un nodo: Es el número de aristas que se recorren para llegar a la raíz desde el nodo. Coincide con el nivel en el que se encuentra ese nodo en el árbol. La raíz tiene profundidad cero.

  -De un árbol: Equivale al número de niveles que se pueden identificar en él, sin tener en cuenta el raíz.

2.2. Tipos/ variantes de árboles

Aunque todos los árboles tengan la misma estructura general básica que los identifica como tales, según sus características podemos distinguir:

- *Árbol vacío.* Se llama árbol vacío o nulo a aquel que no tiene ningún nodo.
- *Árbol dirigido.* Es un árbol en el que las aristas tienen un sentido, de forma que el flujo que se transporta parte hacia una dirección particular.
- *Árbol de expansión.* Es un árbol con un subconjunto de aristas que conectan todos los nodos.
- *Árbol ponderado o etiquetado.* Se conocen también como árboles con costos ya que cada arista tiene asociado un valor o etiqueta que representa su coste, la longitud, etc.
- *Árbol con capacidades.* Se dice de los árboles en los que cada nodo o vértice tiene asociado un peso, una demanda, etc.
- *Árbol ordenado.* Es un árbol en el que los descendentes de cada vértice siguen un tipo de orden.
- *Árbol homogéneo.* Todos los vertices de este tipo de árboles tienen el mismo grado.
- *Árbol k-ario.* Se llaman árboles k-arios a aquellos en lo que cada nodo tiene, como máximo, un número k de descendientes.
  Los más conocidos y usados son los árboles binarios, un caso particular en el que k=2.

  -*Árbol lleno o perfectamente equilibrado.* Si todos sus nodos internos tienen exactamente k hijos.

  -*Árbol completo.* Un árbol en el que todas las hojas tienen la misma profundidad.

  -*Árbol binario equilibrado.* Se dice esto de los árboles binarios en los que la altura del subárbol izquierdo y la altura del subárbol derecho son iguales o difieren en una unidad y, además, todos los subárboles son equilibrados.

  -*Árbol biario estricto.* Cada nodo puede tener cero o dos hijos, pero no se permite que haya árboles con descendencia y otros no.

  -*Montículo.* Árbol binario completo cuyos nodos almacenan un campo clave y cumplen la propiedad de montículo según la cual, la clave que almacena cada vértice es, bien menor que las claves de todos sus descendientes (montículo de mínimos), o mayor que las claves de sus hijos (montículo de máximos).

  -*Árbol binario de búsqueda (BB).* Árbol binario que almacena elementos con campo clave y los nodos cumplen cierta propiedadde ordenación: cada nodo tiene un elemento cuya clave es menor o igual que las claves de su subárbol derecho y mayor o igual que las claves de los nodos de su subárbol izquierdo. Se llama factor de equilibrio de cada nodo a la diferencia de altura entre su subárbol izquierdo y el derecho.
  Dentro de los árboles de búsqueda podemos distinguir también: *árboles AVL, árboles rojo-negro, splay trees o árboles biselados...*

Este trabajo está centrado en buscar los mínimos árboles expandidos con capacidades CMST:

- Cada subárbol es un árbol de expansión mínima (MST), es decir, es un árbol ponderado en el que las distancias de las aristas que lo forman suman menos o igual que las de los demás árboles expandidos que se puedan formar.

- Es un árbol con capacidades- la suma de los pesos de todos los nodos que forman cada subárbol no puede superar un límite fijado.

2.3. Implementación

A la hora de trabajar con árboles existen diferentes formas de guardar los elementos que los componen. No hay una única implementación posible ni buena ya que la elección de una sobre las otras depende sobre todo de qué operaciones se vayan a llevar a cabo con el árbol.

2.3.1. Vector de ascendentes

Se define un vector de dimensión el número de nodos, incluido el raíz, de tal forma que:

- Si hay n nodos más el raíz, se numeran los nodos de forma que cada vértice se representa por ese número *i=1,..,n+1*

- El nodo que representa el raíz es el último, *n+1*.

- En el vector se guarda, para cada índice, el índice de su nodo ascendente. Si *vector(i)=j* entonces *j* es el ascende de *i*.

- Los índices que tengan como ascendente *n+1* son las raíces de los subárboles principales. El nodo raíz tiene asignado su propio índice como ascendente *vector(n+1)=n+1*.

2.3.2. Listas con referencias

[1] Un árbol se almacena mediante nodos con referencias al nodo padre y una lista con referencias a los nodos hijos. Otra forma de implementarlo es que cada nodo tenga una referencia al nodo padre, a un nodo hijo y a un nodo hermano de manera que todos estén enlazados.

2.3.3. Matriz de adyacencia

El árbol se puede almacenar usando una matriz binaria de *(n+1) x (n+1)*. Si en la posición *(i,j)* aparece un 1 significa que hay una arista que une dichos nodos.

Por comodidad con los distintos métodos de resolver el CMST se ha elegido representar el árbol solución mediante una matriz de adyacencia. Esta matriz se complementa con un vector *gate* que indica, para cada índice, la raíz del subárbol al que pertenece. En el árbol que se ha usado de ejemplo en las figuras anteriores *gate(b)=b; gate(c)=gate(e)=gate(f)=gate(g)=c; gate(d)=d*.

[1] Si el árbol es ordenado el conjunto de subárboles que están unidos a la raíz se tratan como listas, si el árbol no es ordenado, como conjuntos.

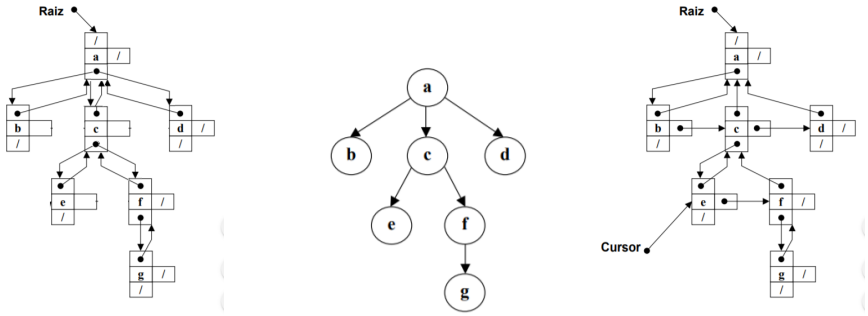

Figura 2.1: Representación del árbol mediante referencias padre/hijos (izq) y referencias padre/hijo/hermano (dcha)

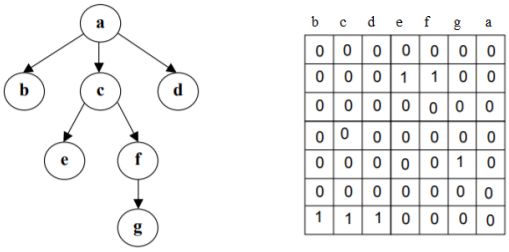

Figura 2.2: Representación del árbol mediante matriz de adyacencias

2.4. Operaciones sobre árboles

En esta sección se van a indicar las principales operaciones que se puden realizar sobre árboles. Señalar que, en realidad, cada tipo de árbol tiene asociadas una serie de operaciones, ya que, por sus características, no todos los árboles permiten que se realicen las mismas operaciones. Es importante también tener en cuenta que hay operaciones relacionadas que no se puden llevar a cabo de forma individual, pej. el borrado del nodo raíz implica la elevación de la raíz de alguno de los subárboles que dependían de ella y la reestructuración de este.

- Inserción de un nodo.
- Borrado de un nodo.

  Nodo raíz- Implica la elevación de algún nodo hijo como nuevo nodo raíz.

  Nodo intermedio- Implica una reestructuración del subárbol.

  Nodo hoja
- Elevación de un nodo.
- Descenso de un nodo.
- División.

- Fusión.
- Equilibrado.

2.5. Complejidad del problema. Formas de afrontarlo

En este trabajo se explora el problema del CMST, mínimo árbol expandido con capacidades. En los siguientes capítulos se detallaran diferentes métodos de afrontarlo y encontrar soluciones válidas. Estos métodos están sujetos a las características del problema, que es combinatoriamente difícil de resolver. A lo largo del trabajo se va hablar de su complejidad y de las limitaciones que esta supone, por lo que, a continuación, se presenta información para encuadrar este asunto.

El problema del mínimo árbol expandido CMST es, en última instancia, un problema de decisión. Se trata de determinar si cada uno de los elementos del problema (nodos) pertenecen o no a un conjunto determinado (subárboles solución) cumpliendo una serie de restricciones. Los problemas de decisión se pueden clasificar en base a distintos criterios como son los aportados por la Teoría de la Computabilidad y la Teoría de la Complejidad Computacional. Estas teorías se incluyen dentro de la Teoría de la Computación que, abarcando las áreas de autómatas, complejidad y computabilidad, estudia las capacidades y límites de los sistemas computacionales. La complejidad y la computabilidad son ramas muy relacionadas entre sí. La primera se centra en describir y analizar la complejidad de un algoritmo o inherente a un problema, mientras que la segunda clasifica los diferentes problemas en base a conceptos introducidos por la anterior, y estudia diferentes modelos formales de computación.

Según la *Teoría de la Computabilidad* los problemas de decisión se distinguen entre:

Decidibles. Son solubles por una Máquina de Turing de forma que esta se detiene sea cuál sea la entrada.

Parcialmente decidibles o reconocibles. Estos problemas tratados en una Máquina de Turing hacen que esta se detenga devolviendo el resultado de los parámetros si estos pertenecen al dominio de la definición, mientras que si dichos parámetros no se encuentran en el dominio, el comportamiento de la máquina es indefinido.

No decidibles. Una Máquina de Turing a la que se le aplique un problema de este tipo para de todas formas. Si los parámetros pertenezcan al dominio de la definición devuleve la solución, y en el caso en el que no, devuelve error.

Tanto los problemas parcialmente decidibles como los no decidicles se consideran indecidibles y se dice que no es posible elaborar un algoritmo que, ante cualquier caso, conduzca a una respuesta de decisión correcta.

La *Teoría de la Complejidad Computacional* mencionada anteriormente estudia el orden de complejidad de los algoritmos que resuelven un **problema decidible**. Con este fin se tiene en cuenta la cantidad de recursos computacionales, principalmente tiempo y espacio de memoria, que requiere resolver el problema en función del tamaño de la entrada. Para clasificar los problemas decidibles según esta teoría es necesario distinguir entre Máquinas de Turing Deterministas y No Deterministas [2]. Los conjuntos de problemas con la misma complejidad computacional se pueden agrupar siguiendo criterios espaciales o temporales, algunas de las clases son:

[2]Máquinas de Turing No Deterministas son aquellas en las que existe al menos un par <estado, símbolo>con más de una posible combinación de actuaciones.

- Clase L. Problemas solubles en espacio de orden logarítmico.
- Clase NL. Problemas solubles en espacio logarítmico por una Máquina de Turing No Determinista.
- Clase P (Polynomial time). En esta clase se incluyen los problemas que se pueden solucionar eficientemente en un tiempo de orden polinómico por una Máquina de Turing Determinista.
- Clase P-Completo. Se incluyen problemas de clase P de forma que todos los problemas de dicha clase se puedan reducir[3] a ellos en tiempo polinómico.
- Clase NP (Non-deterministic Polynomial time). A esta clase pertenecen los problemas que son eficientemente verificables por una Máquina de Turing No Determinista en un tiempo de orden polinómico. Se consideran inherentemente difíciles.
- Clase NP-Completo. Incluye problemas de la clase NP de forma que todos los problemas de dicha clase se puedan reducir a ellos en tiempo polinómico.
- Clase NP-Hard. Esta clase incluye problemas que son intrínsecamente más complejos que aquellos que se pueden resolver por una Máquina de Turing No Determinista en un tiempo polinómico. La optimización de un problema combinatorio de decisión pertenece a esta categoría, por lo que se incluye aquí el problema de encontrar el árbol CMST.

Máquinas de Turing Deterministas son un caso particular de las No Deterministas.En este tipo de máquinas, para cada par <estado, símbolo>posible existe a lo sumo una posibilidad de ejecución.

[3]Reducir un problema es convertirlo en otro de forma que la solución a este pueda ser utilizada para resolver el problema original.

Capítulo 3

Capacitated Minimum Spanning Trees (CMST). Formulaciones y modelos

El problema que se trata de solucionar pasa por encontrar la mejor manera de unir n terminales situadas en distintos puntos, de forma que estén conectadas con un nodo central o base, que será punto de partida o de llegada de los operaciones.
Un método para hacer esta labor de forma eficiente consiste en unir el flujo de varios terminales en una misma línea (red tipo bus), de forma que solo un terminal está directamente unido con la base y a partir de él se distribuya el tráfico hacia los demás. Esta manera de proceder da lugar a una estructura de árbol expandido cuya representación corresponde a la de un grafo *G=(V,A)* donde *V=1,...,n+1* es el conjunto de terminales o nodos (incluyendo la base o raíz) y A las aristas que los unen. Cada nodo conectado directamente con la base, *gate*, es la raíz de un subárbol (o inicio del bus).[1]
En orden a equilibrar el transporte por el grafo, los subárboles tienen asociada una capacidad de flujo máxima Q que pueden manejar (Q es un número entero fijado). Estas especificicaciones dan nombre al problema CMST (Capacitated Minimal Spanning Tree) o mínimo árbol expandido con capacidades.

En este trabajo se va a considerar el caso en que cada subárbol tiene el mismo límite sobre la capacidad. Este límite influye de manera directa en la resolución del problema dando lugar a las siguientes posibles situaciones:

- *Q= 1* La solución en este caso es trivial, ya que la única válida es una red en forma de estrella con todos los nodos como gates unidos al base. A no ser que específicamente se esté buscando este modelo, hay formas más eficientes de resolverlo.

- *Q= 2* Si esta es la capacidad fijada, el problema se puede plantear com uno de coincidencias perfectas (perfect matching)[2].

[1] Se considera que sea cuál sea la posición de la base en la red, toda la información relativa a ese nodo se va almacenar después de la de los terminales, en el índice n+1.

[2] La coincidencia perfecta es una forma de tamaño mínimo de cubrir todos los nodos de un grafo- se da si cada vértice está conectado a exactamente una arista. En grafos con nodos impares se habla de coincidencia casi perfecta ya que un nodo queda disparejo.

- *2<Q<n* Esta acotación es la más habitual y la que corresponde a todos los casos que se van a tratar en este trabajo. Da lugar a un problema de complejidad NP-completo. Como se ha visto antes, esto implica que encontrar un método exacto y eficiente de resolver el problema es complicado, de ahí que ese introduzcan los capítulos de heurísticas.

- $Q \geq n$ En este caso existen restricciones de capacidad redundantes lo que convierte el problema en uno de mínimo árbol expandido sin capacidades (MST). Estos problemas se resuleven de forma óptima con el algoritmo de Prim.

Otras consideraciones a tener en cuenta son los costes. Se va a trabajar únicamente con costes fijos c_{ij} asociados al uso de la unión *(i,j)* $\in A$. En este caso los costes son las distancias entre los distintos puntos. Relativo al uso de las distancias como coste, señalar que este problema se suele tratar como un grafo dirigido, de forma que hay lugar a dos interpretaciones: bien la raíz funciona como emisor de flujo que tiene que llegar a los nodos, o bien son los nodos los que envían flujo hacia la raíz. Dado que los costes son simétricos, ya que *dist(a,b)= dist(b,a)*, la solución óptima alcanzada es la misma independientemente del sentido del flujo.

Aunque no se va a enfocar de este modo en el trabajo presente, en ocasiones, sobre todo cuando este problema se aplica a establecer rutas de distribución física, se incluyen también costes variables que suelen estar asociados al mantenimiento de los caminos o de las terminales.

3.1. Demandas Unitarias

El caso más sencillo de este problema es en el que las demandas o pesos de cada nodo son iguales y unitarios, de manera que se trata de asegurar que todos los nodos estén incluidos en algún subárbol cuyo número de elementos no supere la capacidad Q fijada.
A continuación se presentan diferentes formulaciones válidas para resolver el problema CMST de forma exacta:

3.1.1. Formulación de Flujo

Las siguientes formulaciones están basadas en modelo de flujo en redes. Se implementan usando dos tipos de variables:

- X_{ij} Variables binarias. Indican si la arista que une los nodos i y j se incluye en la solución, en cuyo caso $X_{ij} = 1$, o si no se usa dicha unión y entonces $X_{ij} = 0$.

- Y_{ij} Variables de flujo. Variables enteras que representan la cantidad de flujo que se transporta del nodo i al j. Al tratarse de forma dirigida, el flujo puede estar producido por el nodo raíz y distribuirse hacia los nodos terminales, o al revés. En este caso de demandas unitarias, cada nodo aporta o recibe una unidad del flujo que se transporta en el subárbol al que pertenece.

También aparecen la matriz c_{ij} de distancias entre nodos y el vector $b_i, i = (1..., n + 1)$ que almacena las demandas de las terminales y de la base.
En todas las formulaciones planteadas se va a almacenar toda la información relativa al nodo base en la posición n+1.

GAVISH

Esta formulacion presentada por Gavish [2] se suele tomar como punto de partida a la hora de resolver el problema del CMST. En este caso el vector de demandas cuando el flujo parte de la base hacia las terminales toma los valores $b_{base} = 0$ y $b_i = 1$ *i=1,...,n* para el resto de nodos. El modelo planteado es el siguiente:

minimizar

$$\sum_{i=1}^{n+1} \sum_{j=1}^{n} c_{ij} X_{ij} \tag{3.1}$$

sujeto a

$$\sum_{i=1}^{n+1} X_{ij} = 1 \qquad j = 1, ..., n \tag{3.2}$$

$$\sum_{i=1}^{n+1} Y_{ij} - \sum_{i=1}^{n} Y_{ji} = 1 \qquad j = 1, ..., n \tag{3.3}$$

$$X_{ij} \leq Y_{ij} \leq (Q - b_i) \cdot X_{ij} \qquad i = 1, ..., n+1; \quad j = 1, ..., n \tag{3.4}$$

$$Y_{ij} \geq 0 \qquad i = 1, ..., n+1 \tag{3.5}$$

$$X_{ij} \in 0, 1 \qquad i = 1, ..., n+1; \quad j = 1, ..., n \tag{3.6}$$

La función objetivo a minimizar es el coste de establecer la red que una los terminales con la base. Las condiciones que se formulan para conseguirlo se reflejan en las restricciones.
La primera (3.1.1) garantiza que cada nodo distinto a la base tenga una única arista de entrada (la base no debería tener ninguna, solo arcos de salida- esto varía si cambia el sentido del flujo) haciendo de esta manera que se eviten los ciclos y se mantenga la estructura de árbol en la solución.
Las ecuaciones (3.3) se conocen como ecuaciones de balance, aseguran que se distribuya el flujo por el sistema y que todos los nodos estén incluidos en él.
Las restricciones (3.4) obligan a que solo se utilicen arcos si van a unir nodos relacionados y que el subárbol al que ambos pertenecen no supere la capacidad máxima fijada.
Las restricciones (3.5) y (3.24) son relativas a la naturaleza de las variables de decisión- el flujo entre dos nodos no puede se negativo y X_{ij} solo toma valores binarios indicando la incluisón de los arcos en la solución.

Como se ha comentado anteriormente, en general, los métodos exactos alcanzan raramente el óptimo a la hora de resolver el CMST. Debido a la cantidad de recursos computacionales que consumen normalmente se establece un límite -bien de tiempo o de uso de memoria- que hace que la solución, si es factible, acabe en algún óptimo local. En estos casos, los esfuerzos se enfocan en encontrar cotas que nos den una idea de lo alejado que está el valor óptimo que se busca del óptimo local encontrado. Para ello, se recurre a la versión relajada del problema. En esta se obvia el hecho de que las variables de decisión tengan que tomar valores enteros, lo que da menos rigidez a la hora de resolver el problema y alcanzar una solución. El resultado obtenido de la relajación linear funciona como cota inferior del óptimo de tal forma que:

valor de la relajación lineal $\leq$ *óptimo total* $\leq$ *óptimo local.*

Planteada esta desigualdad es trivial decir que, si el valor de la relajación obtenido y el óptimo local coinciden, entonces se ha alcanzado el valor óptimo absoluto.

En cuanto al límite de los recursos, en este trabajo se ha establecido respecto al tiempo. Se ofrecen soluciones con tiempo de ejecución de 20 y de 50 segundos. Sobra decir que cuantos más recursos se destinen a la resolución, mejores resultados se obtendrán, de manera que es posible que con menor tiempo se alcance únicamente una solución factible pero al ampliarlo se acabe encontrando el óptimo absoluto.[3]
Esta limitación también se aplica a la versión relajada- se va a calcular una primera realajación lineal antes de iniciar el proceso de búsqueda del óptimo y otra al final, pasado el tiempo fijado. Al igual que en la versión entera del problema, se obtienen mejores resultados cuanto más recursos se dedican de forma que se cumple:

$$\textit{relajación lineal 1} \leq \textit{relajación lineal 20s} \leq \textit{relajación lineal 50s.}$$

Una vez aclarado esto y planteada la formulación base, pasamos a hablar de distintas variantes de esta [4] que dan lugar a formulaciones equivalentes [4] según la manera de relacionar los dos tipos de variables de decisión X_{ij}, Y_{ij}. A estas formulaciones se llega sustituyendo las restricciones (3.4) por alguna de las siguientes opciones mientras se mantienen el resto de la formulación original:

$$Y_{ij} \leq QX_{ij} \qquad i = 1, ..., n+1; \quad j = 1, ..., n \tag{3.4a}$$

La relajación lineal que se obtiene a partir de este conjunto de restricciones es la más débil de las que se presentan.

Entre los dos siguientes conjuntos de restricciones (3.4b) y (3.4c) que se ofrecen como alternativas a (3.4) no existe relación de dominancia. Esto implica que en algunas situaciones la relajación lineal asociada a una da un mejor resultado final y, en ocasiones, ocurre al revés.

$$X_{ij} \leq Y_{ij} \leq Q \cdot X_{ij} \qquad i = 1, ..., n+1 \quad j = 1, ..., n \tag{3.4b}$$

$$0 \leq Y_{ij} \leq (Q - b_i) \cdot X_{ij} \qquad i = 1, .., n+1 \quad j = 1, ..., n \tag{3.4c}$$

Todas estas alternativas dan lugar a formulaciones cuya relajación lineal es más débil que la obtenida a partir de la original, pero son má sencillas de resolver.
La rigidez de la relajación lineal está relacionda con la amplitud de las cotas ofrecidas, de forma que, en el caso de hayarse un óptimo local, cuanto más débil sea la versión relajada, más amplia será la cota y más pobre el resultado.

[3] Los resultados de los distintos métodos empleados se han obtenido usando Xpress. Es un solver de optimización que resuelve problemas de programación entera, lineal, mixta, etc. Para la resolución de problemas enteros dedica un gran esfuerzo computacional a la búsqueda de mejores relajaciones. Esto se realiza usando planos de corte mediante un proceso automático. Los límites en el tiempo de ejecución detienen esta operación transcurridos los segundos fijados.

[4] Se habla de formulaciones equivalentes cuando la solución óptima obtenida con ambas es la misma, si bien el conjunto de soluciones factibles no lo es, y tampoco las relajaciones lineales derivadas de ellas.

FORMULACIÓN ALTERNATIVA

Una versión alternativa a la descrita es la siguiente, un caso especial de flujo en redes con costes fijos muy estudiado en el ámbito de la programación entera [3]. Aquí los valores de las demandas cambian respecto a la formulación anterior de forma que $b_{base} = n$ y $b_i = -1$ *; i=1,...,n.* Con esta particulariad el problema queda formulado así:

minimizar

$$\sum_{i=1}^{n+1}\sum_{j=1}^{n} c_{ij}X_{ij} \tag{3.7}$$

sujeto a

$$\sum_{i=1}^{n+1} X_{ij} = 1 \qquad j = 1, ..., n \tag{3.8}$$

$$\sum_{i=1}^{n+1} Y_{ij} - \sum_{i=1}^{n} Y_{ji} = b_i \qquad j = 1, ..., n \tag{3.9}$$

$$Y_{ij} \leq Q \cdot X_{ij} \qquad i = 1, ..., n+1; \quad j = 1, ..., n \tag{3.10}$$

$$Y_{ij} \geq 0 \qquad i = 1, ..., n+1 \tag{3.11}$$

$$X_{ij} \in \{0, 1\} \qquad i = 1, ..., n+1; \quad j = 1, ..., n \tag{3.12}$$

Como se puede observar, es muy parecida a la formulación anterior, salvo por el cambio de interpretación respecto a los nodos que aquí funcionan como *pozos* y la base como *fuente* única emisora de flujo.

3.1.2. Formulación con 2n restricciones

Como se viene señalando a lo largo del trabajo, los métodos exactos para resolver el CMST suelen estar limitados computacionalmente por la gran necesidad de recursos que requieren. Este uso de recursos deriva de la cantidad de ecuaciones a tener en cuenta para resolverlos. Las formulaciones de flujo tienen asociado un coste polinomial a la hora de encontrar la solución óptima, tanto en el problema entero como en el relajado.

Existen otras formulaciones equivalentes para el CMST que reducen el número de variables de decisión a una sola- entre ellas se encuentran las llamadas restricciones *Multistar* y las *Generalizadas de Ruptura de Subtours* [4] que son más complejas de implementar ya que requieren la programación de conjuntos de nodos y el estudio de su cardinalidad y las relaciones entre ellos. Para comparar los resultados obtenidos con las formulaciones de flujo con estas es necesario proyectar el conjunto de soluciones factibles de las primeras en el subespacio de las variables binarias, que son las únicas que aparecen en estas nuevas formulaciones. Se ha demostrado que cualquier conjunto de soluciones factibles X_{ij}, Y_{ij} para la relajación lineal de la formulación de flujo cumple las restricciones *Multistar* y es factible para su versión relajada. Además el coste de encontrar una solución óptima es igual para ambas formulaciones. En cuanto a la formulación que incluye las restricciones *Generalizadas de Ruptura de Subtours* el problema está en que, en general, este coste de encontrar una solución óptima es mayor que el de las anteriores ya que el numero de restricciones es exponencial. Esto hace que sea computacionalmente poco práctico de

resolver, incluso para problemas con un número pequeño de nodos, aunque las cotas que ofrece son, normalmente, más ajustadas que las que se obtienen con *Multistar*.

Se ha planteado otra reformulación del problema [5] usando una única variable con tres índices y carácter binario: Z_{ijk} donde *i=1,...,n+1; j=1,...,n; k=1,...,*$Q - b_i$, de forma que $Z_{ijk} = 1$ indica que por el arco que une los nodos i y j, se transporta una cantidad de flujo igual a k. Esta interpretación de las variables hace que no sean necesarias las restricciones de capacidad ya que su cumplimiento está implícito en el rango de estos nuevos índices. Las variables vistas hasta ahora expresadas en términos de la nueva son:

$$X_{ij} = \sum_{k=1}^{Q-b_i} Z_{ijk} \qquad i = 1, .., n+1; \quad j = 1, ..., n \tag{3.13}$$

$$Y_{ij} = \sum_{k=1}^{Q-b_i} k \cdot Z_{ijk} \qquad i = 1, .., n+1; \quad j = 1, ..., n \tag{3.14}$$

Adaptando el resto de ecuaciones de la formulación de flujo a estas variables se obtiene la formulación con 2n restricciones:

minimizar

$$\sum_{i=1}^{n+1} \sum_{j=1}^{n} \sum_{k=1}^{Q-b_i} c_{ij} Z_{ijk} \tag{3.15}$$

sujeto a

$$\sum_{i=1}^{n+1} \sum_{k=1}^{Q-b_i} Z_{ijk} j = 1, ..., n \tag{3.16}$$

$$\sum_{i=1}^{n+1} \sum_{k=1}^{Q-b_i} k \cdot Z_{ijk} - \sum_{i=1}^{n+1} \sum_{k=1}^{Q-1} k \cdot Z_{ijk} = 1 \qquad j = 1, ..., n \tag{3.17}$$

$$Z_{ijk} \in \{0, 1\} \qquad i = 1, ..., n+1; \quad j = 1, ..., n; \quad k = 1, ..., Q - b_i \tag{3.18}$$

Esta formulación tiene los mismos valores del vector de demandas que la formulación de "Gavish ver 3.1.1"$b_i = 1$ *i=1,...,n;* $b_{base} = 0$. También señalar que, para cada solución factible de la formulación de Gavish existe una solución factible para esta formulacion con el mismo coste, y viceversa. Por lo tanto, el valor objetivo óptimo debe coincidir en las dos, lo que la califica como forma válida de plantear el CMST. El coste de encontrar la relajación lineal es el mismo que para la planteada por Gavish, es decir, las cotas lineales tienen la misma rigidez. Sin embargo, esta nueva formulación es más compacta ya que tiene un número de restricciones de orden O(n), mientras que la de Gavish es de O(n^2).

3.1.3. Formulación Multiproducto

Otra manera de interpretar este problema es viéndolo como transporte de productos. Para esta formulación se va a asociar un producto ficticio a cada nodo distinto del base, de forma que, en el caso de que el sentido sea el de absteceer los terminales, la raíz tiene una unidad de

oferta de cada producto k=1,...,n y cada nodo una unidad de demanda del producto asociado a su índice. El planteamiento de las demandas cambia de forma que ahora ya no están asociadas únicamente a los nodos, si no que cada nodo tiene asociadas demandas de cada producto:

$$b_{ik} = \begin{cases} -1 & \text{si } i = k \quad \text{i=1,...,n; k=1,...,n} \\ 0 & \text{si } i \neq k \quad \text{i=1,...,n; k=1,...,n y si } i = k = base \\ 1 & \text{si } i = base = n+1 \quad \text{y } k = 1, ..., n \end{cases} \tag{3.19}$$

De esta forma, se garantiza que cada terminal únicamente demanda del producto cuyo índice coincide con el suyo, y cero de los demás, lo que hace que la matriz de demandas sea identidad nxn con una última columna (n+1) de -1 asociados a las ofertas de productos de la raíz y una última fila (n+1) de 0 que reflejan las demandas de la base.
En esta formulación entran en juego de nuevo dos variables:

- X_{ij} i=1,...,n+1, j=1,...,n; Variables binarias- Indican si el arco entre los nodo i y j se usa y, por lo tanto, hay una ruta directa entre estos terminales del mismo subárbol en la solución.
- Y_{ijk} i=1,...,n+1, j=1,...,n; k=1,...,n Variables de flujo- Su valor es siempre mayor o igual que cero y traza la ruta que sigue cada producto hasta llegar a su destino. $Y_{ijk} = 1$ si el producto k se transporta del nodo i al j.

Dicho esto, el problema queda formulado así:

minimizar

$$\sum_{i=1}^{n+1} \sum_{j=1}^{n} c_{ij} X_{ij} \tag{3.20}$$

sujeto a

$$\sum_{i=1}^{n+1} X_{ij} = 1 \qquad j = 1, ..., n \tag{3.21}$$

$$\sum_{i=1}^{n} Y_{jik} - \sum_{i=1}^{n+1} Y_{ijk} = b_{j,k} \qquad j = 1, ..., n \quad k = 1, ..., n \tag{3.22}$$

$$\sum_{k=1}^{n} Y_{ijk} \leq Q \cdot X_{ij} \qquad i = 1, ..., n+1; \quad j = 1, ..., n \tag{3.23}$$

$$X_{ij} \in 0, 1 \qquad i = 1, ..., n+1; \quad j = 1, ..., n \tag{3.24}$$

La formulación planteada es la que se conoce como versión agregada o débil. Si a este conjunto de restricciones se le añade la siguiente:

$$Y_{ijk} \leq X_{ij} \qquad i = 1, ..., n+1; \quad j = 1, ..., n; \tag{3.25}$$

se habla de formulación desagregada o fuerte. Este conjunto de ecuaciones aporta cotas más ajustadas gracias a su relajación lineal que, como su propio nombre indica, es más fuerte. pero más complicada de resolver.

3.2. Demandas No Unitarias

El problema del mínimo árbol expandido con capacidades se puede complicar si, en vez de tener demandas unitarias e iguales para cada nodo terminal, se cambia esta característica. Se pueden dar dos situaciones diferentes:

1. Demandas no unitarias pero iguales. En este caso, dado que todos los nodos tienen la misma cantidad de demanda asociada, p.ej. $b_i = (3, 3, ..., 3)$ *i=1,...,n*, se puede asumir que esta demanda es unitaria y ajustar en función el valor de la capacidad máxima Q para cada subárbol.

2. Demandas no unitarias y específicas para cada nodo. Este caso es más complejo ya que cada terminal tiene una demanda distinta p.ej. $b_i = (2, 3, 1, 4, ...)$ *i=1,...,n.* Esto afecta a la capacidad total Q y da lugar a una serie de especificaciones que hacen que el problema sea considerado de complejidad NP-Hard.

Capítulo 4

Heurísticas base

Se ha señalado a lo largo de los capítulos anteriores que el problema del mínimo árbol expandido con capacidades es un problema combinatoriamente complejo y difícil de resolver de manera óptima con métodos exactos. Esto se debe a que, con el número de nodos a conectar, aumentan las variables y las ecuaciones que se han de cumplir. Por eso, hallar una solución entera que cumpla todas las características es complicado y suele consumir muchos recursos, tanto de memoria del sistema como de tiempo. Teniendo esto en cuenta, se suelen aceptar óptimos locales como soluciones válidas, bien sea limitando el uso de recursos que se utiliza en los métodos exactos, fijando unas cotas entre las que se encuentra el óptimo total, o bien recurriendo a técnicas heurísticas.

La R.A.E. define heurística como "Técnica de la investigación y el descubrimiento". El término proviene del griego y significa hallar, inventar. El pensamiento heurístico es considerado algo característico de los seres humanos. Está relacionado con lo que se conoce como pensamiento lateral dado que, la resolución de problemas usando métodos de este tipo se basa, principalmente, en la experiencia para desarrollar algoritmos útiles en cada caso. La metodología heurística se lleva aplicando en diversos campos como las matemáticas, la ingeniería, la psicología e incluso situaciones cotidianas desde tiempos inmemoriales, aunque es a partir del siglo XX cuando se populariza y se comienza a estudiar más exhaustivamente.

Como todo método científico, la heurística tiene un conjunto de principios, reglas y estrategias en los que se apoya. Dentro de las técnicas heurísticas distinguimos dos propósitos: base y mejora.
En este apartado se hablará de las heurísticas base que son aquellas que tratan de dar con una solución factible teniendo en cuenta las premisas del problema. Estas heurísticas pueden ser más o menos rudimentarias, lo que suele dar lugar a un espacio de mejora del que se hablará en el próximo capítulo.

Para dar con una solución factible se pueden usar diferentes procedimientos que, según sus características, se pueden dividir en [7]:

4.1. Procedimientos de construcción. Algoritmo de Prim

Los llamados métodos de construcción parten de una solución incompleta, normalmente un árbol vacío, que se va ampliando a lo largo de las iteraciones del algoritmo hasta dar con una solución factible. Según cómo se vaya creando la solución se pueden distinguir dos métodos:

Mejor arco En cada iteración se usa un criterio greedy [1] para seleccionar el arco más conveniente según la función objetivo (en este caso el de menor coste asociado). Si las condiciones requeridas se cumplen, los dos nodos de los exremos quedan entonces unidos. Esta operación se repite hasta que se alcanza una solución factible. Este procedimiento se puede realizar de forma *secuencial* (se construyen subárboles de uno en uno, hasta que no se alcanza la capacidad límite no se empieza otro) o *en paralelo* (se van formando distintos subárboles simultáneamente).
Ejemplos de algoritmos de esta clase son *Kruskal Modificado* y *Prim Modificado.*

Mejor nodo En este caso se elige un nodo o subárbol concreto mediante criterios greedy y se une al componente más cercano mediante el mejor arco. Los procedimientos de este tipo también se conocen como algoritmos clustering.
Los algoritmos *Martin* y el *método de aproximación de Vogel* son ejemplos de esta forma de proceder.

En este trabajo se va a implementar el algoritmo de Prim modificado en su forma secuencial. Al tratarse de un árbol CMST hay que tener en cuenta las consabidas matriz de costes (distancias entre los puntos), demandas de los nodos (unitarias) y capacidad límite de los subárboles. Las aristas que se usarán en el árbol solución se almacenarán en una matriz binaria de forma que $link_{ij} = 1$ si el nodo i y el nodo j están unidos, y valdrá cero en caso contrario. Como en el caso de los métodos exactos, la función objetivo que se trata de minimizar es el resultado de sumar las distancias entre los nodos unidos por aristas $\sum_{i=1}^{n+1} \sum_{j=1}^{n} c_{ij} \cdot link_{ij}$.
El pseudocódigo de este algoritmo es el siguiente:

Prim secuencial

while haya nodos no incluido en la solucion **do**
  1- Elegir gate del subarbol
    for *i=1,..,n | nodo i* no incluido ya en la solución
      Seleccionar gate: *nodo i* que haga min $c_{base,i}$
    end-for
    Actualizar la solución:
      Marcar nodo seleccionado como gate del subárbol actual
      $link_{base,gate} = 1$
  2- Construir el subárbol con gate elegida en 1)
    while nº nodos en el subárbol $< Q$ **o** no haya más nodos que incluir en la solución **do**
      for *i* $\in$ *subárbol actual; j nodo no incluido en la solución* **do**
        Buscar los nodos *i, j* que hagan min c_{ij}
      end-for
      Actualizar la solución:
        Incluir nodo j en el subárbol
        $link_{i,j} = 1$
    end-while
end-while

[1] Greedy es la palabra inglesa para codicioso, voraz. Los algoritmos de este tipo se consideran ciegos ya que seleccionan en cada paso la mejor de las opciones posibles sin tener en cuenta que estas puedan llevar a situaciones menos favorables en los siguientes movimientos.

Pseudocódigo del algoritmo Prim modificado. Implementación secuencial.

Para desarrollar este algoritmo se han creado dos tipos de vectores binarios unidemensionales de longitud igual al nº de nodos. Un vector general lleva la cuenta de qué nodos han sido añadidos a la solución de forma que, si para un índice toma el valor 1, este nodo está ya conectado. Además, cada vez que se crea un subárbol se utiliza otro vector binario que almacena qué nodos se incluyen en el mismo- de esta forma si un nodo está marcado como visitado en el vector de un subárbol, también lo está en el vector general, pero no lo estará en los vectores asociados a los subárboles a los que no pertenece.

Este algoritmo se conoce como *Prim modificado* ya que en el algoritmo original de Prim se parte de un grafo conexo no dirigido y no se tienen en cuenta restricciones de capacidad a la hora de crear los subárboles de la solución, particularidad que sí encontramos al plantear el problema del CMST.

4.2. Procedimientos basados en ahorros. Algoritmo de Esau Williams

Este tipo de procedimientos parten de una solución factible no óptima (generalmente grafo en forma de estrella) y, siguiendo una serie de pasos, van encontrando soluciones gradualmente mejores al seleccionar en cada iteración del algoritmo la opción más conveniente de entre las posibles. El proceso iterativo termina cuando no quedan movimientos que supongan un ahorro. Por cómo estan concebidos, pueden partir de cualquier otra solución factible y considerarse así procedimientos de mejora.
Ejemplos de estos algoritmos son el de *Esau Williams*, la heurística de ahorros de *Whitney* o el *algoritmo de ahorros en paralelo, PSA* por sus siglas en inglés.

El algoritmo de Esau Williams es una de las herísticas más conocidas para afrontar el problema del CMST. Este algoritmo se puede interpretar desde dos perspectivas distintas.
Una es desde la óptica de **ahorros** basada en el algoritmo de ahorros *Clarke-Wright* desarrollado en 1964 [6] para problemas de rutas de vehículos. Se comienza con un grafo en estrella en el que todos los nodos están unidos directamente con la base dando lugar a tantos subárboles como nodos $link_{base,i} = 1; \; i = 1, .., n$ y $gate_i = i; \; i = 1, .., n$. Partiendo de aquí, se buscan dos nodos pertenecientes a distintos subárboles y, si la suma de las demandas de los nodos de ambos los subárboles no supera Q, entonces ambos componentes se unen en un único subárbol. Para ello se elimina el arco de mayor coste de los que unen los subárboles a la base y se remplaza por la arista *(i,j)* que los una y ofrezca la mejor mejora en el coste total. Se buscan los *nodos i, j* cuya unión haga máximo el ahorro s_{ij}:

$$s_{ij} = c_{base,gate_i} - c_{ij} \tag{4.1}$$

Otra forma de ver el algoritmo de Esau Williams es como una variación de un algoritmo de unificación. En este caso se empieza con un árbol vacío con tantos componentes como nodos $link_{i,j} = 0 \; \forall i, j$ y $gate_i = i; \; i = 1, .., n$ y se estudia si, para cada nodo, es menos costoso conectarlo a otro nodo que directamente a la base. Así se va buscando unir arcos factibles que conecten cada vez la pareja de *nodos i, j* que tenga asociado el mínimo **tradeoff**:

$$t_{ij} = c_{ij} - c_{base,gate_i} \tag{4.2}$$

Independientemente de si se elige usar la versión de ahorros o la de tradeoffs, hay que tener en cuenta que estas matrices no son simétricas $s_{ij} \neq s_{ji}$ y $t_{ij} \neq t_{ji}$ ya que, aunque la matriz de distancias en este caso sí es simétrica, los costes de conexión con la base $c_{base,gate_i} \neq c_{base,gate_j}$.
En cada iteración del algoritmo se eligen bien el valor máximo (ahorros) o mínimo (tradeoffs) absoluto de la matriz, por lo que ya se está determinando qué subárbol va a mantener su gate y cuál va a cambiar su enlace con la base para unirse a él.
En cuanto a la selección de dicho valor mínimo o máximo, puede darse el caso de que varias parejas de nodos tomen el mismo. Estas coincidecias se pueden resolver de diversas maneras: quedándonos con el primer valor que encontremos, con el último, o, como se ha implementado en este caso, seleccionando una pareja de nodos aleatoriamente de entre todas cuyos valores de s_{ij} o t_{ij} coincidan con el óptimo. Esta forma de proceder incluye cierto aleatoriedad en el algoritmo de Esau Williams que hace que la solución no sea necesariamente igual en cada ejecución del programa.
Señalar que este algoritmo forma las distintas ramas favoreciendo la unión de los nodos más alejados de la base. Además, una vez que se unen dos nodos no se pueden desconectar. Cuando se unen dos componentes siempre es por el gate de uno de ellos y, de haberlos, arrastra consigo los nodos unidos a él previamente.

Se ha decido usar la versión del algoritmo en la que se buscan los mínimos *tradeoffs* cuyo pseudocódigo se describe a continuación. Este mismo pseudocódigo, con pequeñas variaciones, es fácilmente adaptable al caso de ahorros.

Esau Williams

Valores iniciales:
$t_{i,j} = 0$ *i,j=1,..,n*
$t_{i,i} = \infty$ *i=1,..,n*
$gate_i = i$ *i,j=1,..,n*

while haya *tradeoffs* negativos **do**
1- Calcular Tradeoffs
Calcular la matriz de tradeoffs
for *i, i=1,..,n* **do**
Buscar, para cada *nodo i*, el tradeoff min
end-for
Encontrar el mínimo tradeoff total de entre los anteriores
(si hay coincidencias se elige uno al azar)
Se estudia unir los nodos:
nodo1=indice i de t_{ij} *en el que se ha encontrado el tradeoff mínimo*
nodo2=indice j de t_{ij} *correspondiente al menor tradeoff relacionado con nodo1*
2- Comprobar requisitos
if $t_{nodo1,nodo2} \geq 0$ **then**
no hay tradeoffs negativos
for *i, i=1,..,n/* $gate_i = i$ **do**
$link_{base,i} = 1$
end-for
FIN
else Comprobar restricciones:
if $capacidad_{subarbolnodo1} + capacidad_{subarbolnodo2} \leq Q$ **and** no se forman ciclos **then**
Actualizar la solución:
$link_{nodo1,nodo2} = 1$
Actualizar subárboles-unir componentes:
for *i, i=1,..,n /gate(i)=gate(nodo2)* **do**
$gate_i = gate_{nodo1}$
end-for
end-if
Actualizar Tradeoffs
$t_{nodo1,nodo2} = \infty$
$t_{nodo2,nodo1} = \infty$
end-if
end-while

Pseudocódigo del algoritmo de Esau Williams

En la implementación de este algoritmo también se usa un vector binario que indica, para cada nodo distinto a la base, si este se ha incluido en la solución en cuyo caso toma valor 1, o no. Este vector es útil para comprobar la presencia de ciclos al formar la solución.

Este algoritmo se suele usar como referencia para comparar otras heurísticas tanto en cuanto a los resultados obtenidos como al uso de recursos para llegar a ellos. La complejidad computacionl en este algoritmo es de orden $O(n^2 \cdot logn)$ siendo la actualización de los tradeoffs y los

cálculos que implica en cada iteración lo que más tiempo requiere.

Mejoras del algoritmo. Parametrizaciones.

El algoritmo de Esau Williams es una técnica heurística bastante competente pero, aunque siempre ofrece una solución factible, es un algoritmo miope y codicioso, características que hacen mella en su desempeño.
Se han planteado distintas formas de salvar estos inconvenientes, desde procedimientos de mejora que se verán en el próximo capítulo, hasta ligeras modificaciones en el algoritmo. Estas modificaciones pasan por usar una versión parametrizada del criterio de selección de componentes a unir[10], bien sean *ahorros* o *tradeoffs*. Dado que el algoritmo original se ha desarrollado usando tradeoffs, vamos a continuar adaptando esta métrica.

Tal cómo está planteado el algoritmo, se tiende a formar bastantes más subárboles de los que habría en la solución óptima, siendo estos de pesos muy parecidos y con una capacidad ligeramente superior a la mitad de la fijada (lo que impide posibles uniones entre ellos). Teniendo en cuenta esto, interesaría encontrar una manera de unir subárboles más controlada. De acuerdo a esta idea Jothi y Raghavachari (2003) [9] proponen la siguiente reformulación del cálculo de los tradeoffs:

$$t_{ij} = (c_{ij} - c_{base,gate_i}) \cdot (\sum_{k \in V_i} b_k)^{\delta} \tag{4.3}$$

Esta fórmula [11] tiene en cuenta las demandas del subárbol en el que se encuentra el *nodo i*, cuya inclusión está justificada por la naturaleza del problema CMST en sí mismo. El CMST es, básicamente, la combinación de dos problemas diferentes: el problema del mínimo árbol expandido (MSTP) y el de Bin-Packing (BPP). Sabiendo esto es fácil pensar que combinar las técnicas de resolución de ambos problemas puede dar buenos resultados.
El algoritmo de Esau Williams trata de equilibrar el coste de formar un subárbol (coste MST) con el que supone unir los nodos a la base. Mientras que el primero de estos costes es independiente del tamaño del subárbol, el segundo es proporcional al número de nodos de este. Sin embargo, en la ecuación original no aparece ninguna información relativa a esta observación, por eso en esta modificación se pretende mejorar el equilibrio entre estos costes y se incluyen las demandas del subárbol que mantiene su anclaje a la base. De esta forma, se da prioridad a que crezcan los árboles de mayor peso (en este caso de demadas unitarias, aquellos con mayor número de nodos) y se evita la ramificación. Esta idea de reducir el número de subárboles y que, por lo tanto, aumente su capacidad es propia de los problemas de Bin-Packing.

El parámetro δ es una constante fijada que puede tomar valores reales entre 0 y 1. Ajustar el valor del parámetro es complicado ya que, según la distribución de los nodos y el peso que vayan tomando los subárboles, se puede desembocar en situaciones que no sean las buscadas.
Dado no hay una manera específica de encontrar un valor de δ que sea válido para todos los problemas, se van a probar diferentes opciones δ=0.25, 0.5, 0.75, 1 y comparar los resultados. Observar que si $\delta = 0$ la fórmula es la misma que la del cálculo de tradeoffs original por lo que no se va a evaluar en esta sección.

Como se podrá comprobar en el posterior apartado de resultados, la gracia de esta modificación es que, los resultados obtenidos con ella van a ser, en general, mejores y, en el peor de los casos, igual de buenos que los obtenidos con la versión original del algoritmo de Esau Williams compensando que el coste computacional sea ligeramente superior.

Capítulo 5

Heurísticas de mejora: Búsqueda local

Las heurísticas de mejora son, como su nombre indica, algoritmos pensados para perfeccionar una solución, ya de por sí factible, a un problema. Este tipo de procedimientos se pueden clasificar en:

- Búsqueda local. Los algoritmos de este tipo parten de una solución factible y buscan posibles modificaciones (incluir o excluir arcos de la solución reconectando los nodos implicados en diferentes posiciones dentro del árbol) que den lugar a una mejora en la solución. Este procedimiento se repite mientras se encuentren mejoras que cumplan las condiciones del problema y se puedan llevar a cabo.

- Algoritmos de Segundo Orden (SOA). En este caso podríamos hablar más específicamente de SOGA (Second Orden Greedy Algorithms). Estos algoritmos usan una dinámica de maestro-esclavo [1] en la que, partiendo de una solución inicial factible generada por un algoritmo de primer orden (FOGA) y restringiendo la solución para que se incluyan o excluyan determinados arcos, se intenta generar mejores soluciones a lo largo de las iteraciones. Como algoritmo de primer nivel se suele usar Esau Williams o alguna modificación de este. Existes diferentes desarrollos de procedimientos de este tipo, dos ejemplos conocidos son los algoritmos *inhibit* y *join*.

Vamos a centrarnos en los procedimientos de búsqueda local. Existen diferentes formas de abordar estos métodos de mejora, desde el número de nodos que están implicados en ella y la posición que ocupan en el árbol, pasando por la forma en la que se hace el cambio y cómo se buscan los elementos que van a participar en él, cómo se adapta la solución a los cambios incorporados, etc.

Antes de seguir explorando estas opciones, hay que tener en cuenta la diferencia entre un cambio factible y un cambio rentable o beneficioso. Un *cambio factible* es, como su nombre indica, un cambio que puede hacerse, es decir, si se lleva a cabo no se violan las características del problema: se cumplen los límites de capacidad en cada subárbol, se mantiene la estructura de árbol sin la presencia de ciclos en la solución. Un *cambio benefecioso* tiene que suponer una

[1] La dinámica de maestro-esclavo es propia de algoritmos paralelos en los que un maestro genera y reparte tareas entre los esclavos que las ejecutan y devuelven sus resultados.

ventaja respecto a la solución de la que se parte, es decir, el coste del árbol resultante tiene que ser menor del que se toma de partida $c(\boldsymbol{T}) > c(\boldsymbol{T'})$ siendo **T** el árbol inicial factible y **T'** el nuevo árbol tras aplicar el algoritmo de mejora. Los cambios que se buscan han de cumplir ambas condiciones y ser, por lo tanto, factibles y beneficiosos.

5.1. Estructuras de búsqueda local

Se pueden distinguir principalmente dos formatos de búsqueda de una mejor solución:

Estructuras basadas en nodos
En el trabajo de Alberg et al. [12] se propone un procedimiento que mejore una solución factible mediante cambios en la asignación de nodos a subárboles. Existen dos tipos de movimientos entre subárboles:

- *Shift* es la palabra para cambio, desplazamiento. Los movimientos de este tipo seleccionan un nodo y simplemente modifican su posición en la solución incluyéndolo en un subárbol diferente.
  El coste de la operación *shift* para que el nodo i pase del subárbol T[i] al subárbol T[j] es

$$c(\mathbf{S}[i]\backslash i) - c(\mathbf{S}[i]) + c(i \cup \mathbf{S}[j]) - c(\mathbf{S}[j]) \tag{5.1}$$

  Siendo **T** el árbol solución, $\mathbf{T}[i]$ indica el subárbol en el que se encuentra el nodo i, $S[i]$ el conjunto de nodos incluidos en el subárbol $\mathbf{T}[j]$ y los costes se refieren a las distancias.

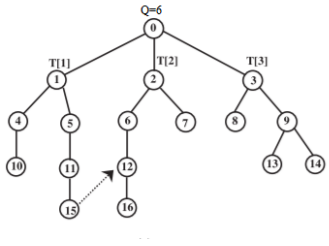

Figura 5.1: Ejemplo de movimiento tipo shift

- *Exchange*. Los movimientos de tipo exchange, o intercambios, implican dos modificaciones. En este caso dos nodos de subárboles ditintos se cambian sus posiciones en la solución, tomando cada uno el lugar que ocupaba el otro.
  El coste de un movimiento *exchange* entre los nodos i y j, tal que $\mathbf{T}[i] \neq \mathbf{T}[j]$, es

$$c(j \cup \mathbf{S}[i]\backslash i) - c(\mathbf{S}[i]) + c(i \cup \mathbf{S}[j]\backslash j) - c(\mathbf{S}[j]) \tag{5.2}$$

Los nodos implicados en estos movimientos pueden ser tanto nodos hojas como nodos intermedios por lo que pueden ser necesarios ajustes en los subárboles implicados para que las conexiones se realicen de la mejor manera posible (buscar subárboles MST).

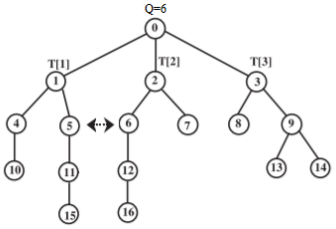

Figura 5.2: Ejemplo de movimiento tipo exchange

El procedimiento, como se ha comentado anteriormente, permite solo cambios factibles, lo que pasa por comprobar que se cumplan las restricciones de capacidad. En el caso de movimientos *shift* está comprobación es indispensable antes de aprobar el cambio ya que es una acción aditiva, la demanda del nodo que se cambia de posición pasa de sumar en el subárbol de origen a hacerlo en el de destino. En cambio, en los movimientos *exchange*, en este caso de demandas iguales y unitarias, dicha comprobación no es necesaria ya que el peso de los subárboles implicados en el movimiento no varía.
En cuanto al coste del algoritmo, se podría acotar superiormente como de orden $O(n^2 \cdot Q^2)$. Se deduce teniendo en cuenta el número de movimientos *exchange* posibles n^2, superior al número de cambios *shift* posibles $n \cdot L$ donde L es el número de subárboles conectados a la raíz; y sabiendo que la posible reestructuración de los subárboles pasa por encontrar un árbol MST con demandas unitarias que se resuelve en tiempo $O(Q^2)$.

Estructuras basadas en árboles
Sharaiha et al. [13] presentan otra forma de llevar a cabo la mejora de la solución. Plantean un sistema basado en "cortar y pegar" en el que se corta un subárbol o una parte de él, y se pega por la raíz del subárbol cortado a la base o a otro subárbol mediante el arco de menor coste, manteniendo su forma de origen. Este método también se puede ver como un doble exchange.
El coste de la opercaión de "cortar y pega" es simplemente $c_{kj} - c_{ij}$ ya que se solo se requiere eliminar el arco que une el nodo i, base del subárbol que se corta, a su antecedente original j y reconectarlo mediante el nodo k a su nuevo árbol de destino.
La complejidad de este método es igual que la del anterior, $O(n^2 \cdot Q^2)$, ya que hay n^2 posibles operaciones de "cortar y pega" y calcular el coste de cada operción es de orden $O(Q^2)$.

Ahuja et al. [14] han propuesto una versión compuesta que incluye elementos de la búsqueda local basada tanto en nodos como en árboles. Le añaden una heurística para encontrar los cambios más adecuados que reduce a complejidad total del algoritmo y lo sitúa como referecia en cuanto a métodos de búsqueda local.

5.2. Multi-exchanges

Estos procesos [15] descritos se pueden aplicar de forma que se estudie un único movimiento de mejora en cada iteración hasta que no se encuentren cambios favorables, o bien, a mayor escala, lo que se conoce como multi-exchanges. Los intercambios múltiples son movimienos enlazados en los que participan más de dos subárboles. Se conocen dos tipos de movimiento:

Movimientos tipo ciclo

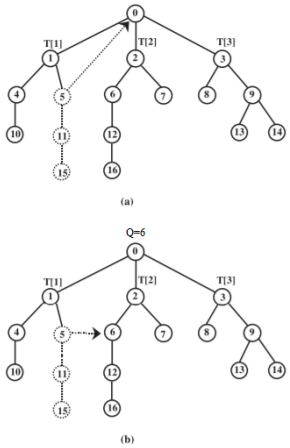

Figura 5.3: Ejemplos de "corta y pega" en estructuras basadas en árboles

En los intercambios cíclicos intervienen todos los subárboles con raíz en la base. Se basan en seleccionar un nodo de cada subárbol $i_1, i_2, ..., i_L$ de forma que $\mathbf{T}[i_j] \neq \mathbf{T}[i_k]$ si $j \neq k$ y moverlos de forma que el nodo i_1 pase a estar en el subárbol $\mathbf{T}[i_2]$, i_2 en el subárbol $\mathbf{T}[i_3]$ y así sucesivamente hasta que i_L se mueva al subárbol $\mathbf{T}[i_1]$.

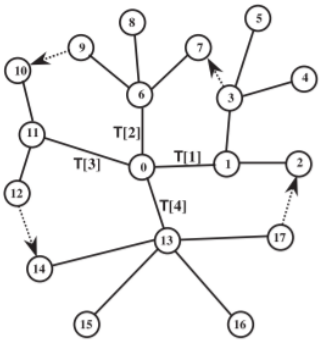

Figura 5.4: Ejemplo de exchange cíclico

Estos cambios han de ser factibles para poder llevarse a cabo, es decir, el límite de la capacidad se tiene que respetar en cada subárbol. Además, para que los intercambios sean convenientes y por tanto se lleven a cabo, el nuevo árbol solución **T'** ha de tener un coste asociado menor que el original, $c(\boldsymbol{T}) > c(\boldsymbol{T'})$.

Al ejecutar este procedimiento los nuevos subárboles pueden necesitar un reajuste a fin de que sigan siendo MST. Al llevar a cabo esta operación puede darse el caso de que un subárbol se divida aumentando así el número de subárboles unidos a la base. Sin embargo, el número de subárboles no puede disminuir en este tipo de proceso.

Movimientos tipo camino
Estos intercambios son parecidos a los descritos anteriormente, en ellos también intervienen todos los subárboles unidos al base, pero en este caso no se completa el ciclo de intercambios. Se selecciona un nodo de cada subárbol $i_1, i_2, ..., i_L - 1$ de forma que $\mathbf{T}[i_j] \neq \mathbf{T}[i_k]$ si $j \neq k$ y moverlos de forma que el nodo i_1 pase a estar en el subárbol $\mathbf{T}[i_2]$, i_2 en el subárbol $\mathbf{T}[i_3]$ y así sucesivamente pero, en este caso, no hay intercambios entre el subárbol $\mathbf{T}[i_L]$ y el $\mathbf{T}[i_1]$. Es decir, el subárbol $\mathbf{T}[i_1]$ no recibe nodos nuevos y $\mathbf{T}[i_L]$ no pierde ninguno de los nodos de la solución original.

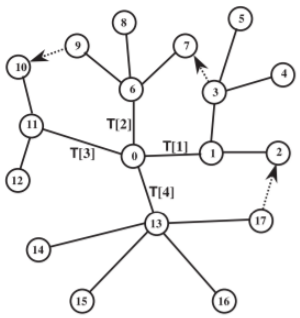

Figura 5.5: Ejemplo de path exchange

De igual forma que en el caso cíclico, los nuevos subárboles tras los intercambios pueden necesitar reajustes para ser árboles mínimos expandidos MST. Este proceso puede llevar a la formación de nuevos subárboles unidos al base, pero a diferencia de en los intercambios cíclicos también se puede disminuir el número de estos respecto a la solución original **T**. Esto se debe a la naturaleza del intercambio que, en cada solución alternativa elimina un nodo del primer subárbol que tiene en cuenta $\mathbf{T}[i_1]$ sin reponerlo, por lo que este puede llegar a desaparecer.

Como en todos los procedimientos mencionados se tienen que respetar la restricción de capacidad fijada para que sea un movimiento factible. En este caso hay que tener especial cuidado con el último subárbol del camino $\mathbf{T}[i_L]$ ya que en el resto el peso disminuye o se mantiene (demandas unitarias) pero en este subárbol siempre aumenta. Para que se considere un cambio favorable y sea aceptado, el coste de las uniones en el nuevo árbol **T'** ha de ser menor que el de partida **T**, $c(\boldsymbol{T}) > c(\boldsymbol{T'})$.

5.3. Implementación de mejoras

En este trabajo se han implementado dos procedimientos de mejora más sencillos que los anteriores descritos. Ambos realizan cambios únicamente sobre dos subárboles cada vez y se aplican solo sobre los nodos finales del árbol solución original **T** obtenido mediante una de las heurísticas descritas en el capítulo anterior: Prim modificado, Esau-Williams y Esau-Williams parametrizado. Señalar que, aunque se apliquen sobre los resultados obtenidos a partir de las heurísticas nombradas, los procedimientos de mejora se pueden implementar a partir de cualquier otra solución factible que tenga margen de mejora, por lo que se podrían usar también sobre los árboles resultado de métodos exactos.

Shift

Se ha desarrollado un algoritmo de mejora basado en la idea shift de transladar un nodo i de un subárbol a otro.
La idea es, partiendo de los nodos hoja originales, evaluar las distancias que los separan de los nodos pertenecientes a los subárboles distintos al que le contiene. En el caso de que haya un nodo j de otro subárbol $\mathbf{T}[j] \neq \mathbf{T}[i]$ cuya distancia al nodo de estudio sea menor a la que le une a su ascendente actual, moverle de $\mathbf{T}[i]$ a $\mathbf{T}[j]$ como descendente de dicho nodo j.
Sabiendo esto, una serie de aclaraciones a tener en cuenta:

- Se parte evaluando los nodos hojas de $\mathbf{T}$. Si tras un movimiento un nodo que en el árbol original era intermedio pasa a ser final, se estudia también.
- Si tras un movimiento un nodo que era hoja en $\mathbf{T}$ pasa a ser intermedio al haberse conectado con otro anteriormente, no se estudia.
- Cuando se han explorado los posibles cambios de un nodo hoja, se haya llevado a cabo el movimiento o no, no se vuelve a examinar.
- El procedimiento termina cuando no quedan movimientos factibles.
- Se considera que el cambio de que un nodo i conectado a j pase a estarlo al nodo k es favorable si $c_{j,i} > c_{k,i}$.
- Antes de dar por bueno cualquier cambio es necesario comprobar que se respeta el límite de la capacidad en el subárbol al que se incorpora el nuevo nodo.

El pseudocódigo correspondiente al procedimiento descrito es el siguiente:

Shift nodos finales

Valores iniciales:
link$_{i,j}$ i=1,..,n+1; j=1,..,n+1 matriz de aristas correspondiente a T árbol solución
nodoshoja vector con los nodos hoja de *link*

for i nodoshoja(i) que no haya sido estudiado
Buscar nodo j más cercano a i tal que $\boldsymbol{T}[i] \neq \boldsymbol{T}[j]$
Comprobar que se cumpla la restricción capacidad:
if capacidad($\mathbf{T}[j]$)+$b_i \leq Q$
Comprobar que el cambio sea beneficioso
if $c_{j,i} - c_{k,i} > 0$
Actualizar la solución
$link_{ascendente(i),i} = 0$
$link_{j,i} = 1$
$gate_i = gate_j$
end-if
end-if
Marcar nodo i como estudiado
Actualizar la lista de nodos finales *nodoshoja*
end-for

$c(\boldsymbol{T'})$=$\sum_{i=1}^{n+1} \sum_{j=1}^{n+1} c_{ij} \cdot link_{ij}$

Exchange
Se ha implementado también un procedimiento de mejora basado en el movimiento *exchange*. Este algoritmo es diferente al anterior enntre otras cosas en que en el movimiento intervienen dos nodos finales i, j pertetencientes a diferentes subárboles unidos al base T[i]≠T[j] cuyas posiciones se intercambian si el cambio es factible y beneficiso. Aspectos a tener en cuenta:

- En este procedimiento la estructura del árbol original **T** no cambia, no se añaden ni eliminan nodos a ningún subárbol, simplemente pueden variar los elementos que ocupan las posiciones finales.
- En relación con el punto anterior, señalar también que, en el caso que se está estudiando en el que todos los nodos tienen demandas iguales, no es necesario controlar las capacidades de los subárboles ya que no cambia con los movimientos. Por lo tanto, dado que se parte de una solució factible **T** que cumple las restricciones propias del problema, independientemente de qué intercambios se realicen se obtendrá una solucion **T'** válida.
- Este procedimiento es especialmente útil cuando la mayoría de los subárboles de **T** alcanzan el límite de la capacidad.
- Aunque todos los intercambios sean factibles, para que sea beneficioso intercambiar las posiciones de un nodo i conectado a k y de un nodo j conectado a l se tiene que cumplir: $c_{k,i} + c_{l,j} > c_{l,i} + c_{k,j}$ siendo T[i]≠T[j].

Con intención de aclarar el procedimiento se incluye el pseudocódigo del mismo:

Exchange nodos finales

Valores iniciales:
link$_{i,j}$ i=1,..,n+1; j=1,..,n+1 matriz de aristas correspondiente a T árbol solución
nodoshoja vector con los nodos hoja de *link*

for i=1,..,nº de nodos hoja -1
  for j=i,..,nº de nodos hoja | T[i]≠T[j]
    Comprobar que el intercambio de los nodos i y j sea beneficioso:
      if $(c_{k,i} + c_{l,j}) - (c_{l,i} + c_{k,j}) > 0$
        Actualizar la solución
          ai:=ascendente(i); aj:=ascendente(j)
          Aristas:
          $link_{ai,i} = 0$
          $link_{aj,j} = 0$
          $link_{ai,j} = 1$
          $link_{aj,i} = 1$
          Subárboles:
          $gi := gate_i$
          $gate_i = gate_j$
          $gate_j = gi$
      end-if
    end-if
end-for

$c(\boldsymbol{T'})$=$\sum_{i=1}^{n+1} \sum_{j=1}^{n+1} c_{ij} \cdot link_{ij}$

Capítulo 6

Metaheurísticas

El nombre que se le da a estas técnicas se obtiene de añadir el prefijo griego "meta" que indica más allá o nivel superior. Las metaheurísticas son por lo tanto métodos heurísticos que, a base de fuerza computacional, exploran el espacio de búsqueda buscando entre los candidatos la mejor opción como solución al problema. Para recorrer el espacio de soluciones se desarrolla un algoritmo iterativo de forma que en cada repeteción del mismo se obtiene una solución factible para el problema. Estas técnicas no garantizan que cada solución obtenida sea mejor a la de la iteración anterior, si no que las evalúan y se quedan con la que mejor valor de la función objetivo presente de todas las encontradas a lo largo del proceso.
Existen diversas técnicas metaheurísticas, en el problema concreto del CMST las más utilizadas entre otras son: *búsqueda tabú, simulated annealing y GRASP* [16].

6.1. Simulated Annealing

Esta técnica está inspirada en el mundo de la metalurgia y la termodinámica teniendo en cuenta el proceso de solidifcación del hierro fundido que, al enfriarse muy despacio, tiende a formar una estructura de energía mínima. Este proceso es imitado por la estrategia de búsqueda local que se va a presentar.

Este procediemiento es de memoria a corto plazo, lo que significa que únicamente almacena una solución, que será la solución final ofrecida, y que se va actualizando a lo larg de las iteraciones según se encuentran otras mejores.
En cada iteración del SA se parte de una solución factible **T** para el problema obtenida mediante una heurística previa aleatorizada (Esau Williams, Esau Williams modificado, etc). De forma aleatoria se elige una solución factible del entorno de la inicial **T'** (es decir, una a la que se pueda llegar aplicando alguna técnica de intercambios sobre la dada). Entonces se comparan los costes de ambas soluciones. Si $c(\boldsymbol{T'}) \leq c(\boldsymbol{T})$ se actualiza **T'** como mejor solución. En cambio, si $c(\boldsymbol{T'}) > c(\boldsymbol{T})$, se acepta **T'** como solución con una probabildad p.
Esta función de probabilidad de acuerdo a la analogía con la metalurgia suele decrecer con el tiempo y el deterioro de la función objetivo, es decir, al principio se aceptarán muchas soluciones alejadas del óptimo, pero la probabilidad de cambios a soluciones peores irá disminuyendo. La función de probabilidad ha de ser de tipo exponencial, típicamente se usa la distribución de Boltzamann [1] que depende de un parámetro inicial que responde a la "temperatura" en esa

[1]La distribución de Boltzamann, también conocida como distribución de Gibbs, es utilizada para indicar la

iteración y que ha de especificarse. Si la solución **T'** no se ha aceptado como mejor solución, se mantiene el valor que se tenía guardado de ella para la siguiente iteración.
El criterio de parada del algoritmo suele ser cuando se realizan un número fijado de iteraciones.

6.2. GRASP

Esta metaheurística es de las técnicas más utilizadas en problemas de optimización combinatoria. Es un proceso de búsqueda multiarranque y adaptativo[17] en el que cada iteración tiene dos fases diferenciadas:

1. **Fase de construcción**. En esta fase inicial se busca dar con una solución factible del problema, no necesariamente óptima. Para encontrar esta solución típicamente se usan algoritmos greedy aleatorizados, como puede ser Esau Wiliamso alguna modificación de este.

2. **Fase de mejora (búsqueda local)**. En esta fase se busca mejorar la solución que se obtiene en el paso anterior encontrando, al menos, un óptimo local. Aquí se puede implementar cualquier algoritmo heurístico de mejora.

El pseudocódigo de las instrucciones básicas de esta técnica es el siguiente:

Metaheurística GRASP

Valores iniciales:
    nº de iteraciones
    solGrasp=∞
    $linkGrasp_{i,j} = 0 i = 1,..,N; j = 1,..,N$

for *i=1,..,nº de iteraciones*
    1-Construcción
        T obtenido mediante heurística greedy aleatorizada
    2- Mejora
        T'=HeurísticaMejora (**T**)
    ¿Actualizar la solución?
        if *c(**T'**)<solGrasp* **then**
            solGrasp=*c(**T'**)*
            linkGrasp=***T'***
        end-if
end-for

$c(\boldsymbol{T})=\sum_{i=1}^{n+1}\sum_{j=1}^{n+1} c_{ij}\cdot link_{ij}$

probabiliad de que un sistema se encuentre en cierto estado. Se utiliza en el campo de la metalurgia antes mencionado $p = exp(-\frac{1}{Temp(n)}\Delta F_n)$ donde $\triangle F_n$=F(x)-F(x_n) es el incremento de la función objetivo en la iteración n. En nuestro caso F(x) es c(**T**) el coste del subárbol inicial y F(x_n) es c(**T'**) obtenida en la n-ésima iteración del algoritmo SA.

Este método, al igual que el *simulated annealing* visto en la sección anterior, es de mememoria a corto plazo e, igual que este, almacena a lo largo de las iteraciones la mejor solución encontrada. El hecho de que sea multiarranque o multistart tiene como objetivo no estancarse en óptimos locales y ampliar el espacio de búsqueda. Para que el multistart sea posible es necesario que el algoritmo que se use en la fase de construcción esté aleatorizado.

El acto de aleatorizar un algoritmo greedy pasa por crear una lista de candidatos RCL (Restricted Candidate List) en la que entran unos determinados valores seleccionados según alguno de los criterios detallados a continuación. De los elementos de esta lista se elige uno aleatoriamente para pasar a formar parte de la solución. En muchas ocasiones en esta lista se incluyen elementos que, aún cercanos a ella, no son la mejor opción según el criterio definido por el algoritmo greedy que se use. Esta forma de elaborar la solución es la que permite que se den variaciones en la solución de la fase de construcción. Esto es lo que da sentido al multiarranque, ya que de otra forma, la única fuente de variedad en las soluciones sería la forma de resolver la presencia de coincidencias a la hora de elegir candidatos.

6.2.1. Formas de aleatorizar algoritmos heurísticos

Se van detallar dos maneras comunes y sencillas de incorporar la aleatoriedad necesaria a las heurísticas codiciosas de la fase de costrucción.

Cardinalidad

Se fija una cantidad de elementos k que van a formar la lista RCL y se introducen en ella los que cumplan cierto criterio. En nuesto caso:

RCL basado en cardinalidad

Valores iniciales:
- k :integer
- $link_{i,j} = 0$ i=1,..,N; j=1,..,N

Calcular función de costes miope [2](*matriz de tradeoffs en EW)*
while sea posible establecer uniones *(haya tradeoffs neegativos EW)* **do**
- Crear lista de candidatos RCL
  - RCL=k elementos con menor coste miope
- Seleccionar uno al azar de entre los candidatos
  - *s=RCL((random*k)+1)*
- Actualizaciones
  - Incluir s en la solución
  - Recalcular matriz miope (tradeoffs EW)

end-while

Valor

Otra forma de crear la lista de candidatos RCL es, no seleccionando un número exacto de can-

didatos, si no incluyendo en ella todos los que estén en un rango de valores que nos interese.
Para calcular ese rango vamos a usar c_*, el valor más bajo de la función miope, c^* el valor más alto de la misma función y un parámetro real $0 \leq \alpha \leq 1$ cuyo valor se va a fijar según nos interese.
Teniendo esto en cuenta, el psedocódigo de esta forma de crear la RCL qeda de la siguiente manera:

RCL basado en valor

Valores iniciales:
    α :real
    $link_{i,j} = 0$ i=1,..,N; j=1,..,N

Calcular función de costes miope (*matriz de tradeoffs en EW)*
while sea posible establecer uniones *haya tradeofs neegativos EW)* **do**
    Crear lista de candidatos RCL
        c_*=min valor de los costes miopes
        c^*=max valor de los costes miopes
        RCL=elementos cuyo coste miope $\leq c_* + \alpha \cdot (c^* - c_*)$
    Seleccionar uno al azar de entre los candidatos
        *s=RCL((random*k)+1)*
    Actualizaciones
        Incluir s en la solución
        Recalcular matriz de costes miope (tradeoffs EW)
end-while

El árbol solución **T** que se obtiene usando cuanlquiera de estas versiones aleatorizadas es una solución siempre factible, aunque no siempre coincide con un óptimo local.

Capítulo 7

Resultados

7.1. Método Exacto

7.1.1. Flujo en redes

GAVISH 40 NODOS

Q=3									
	Archivo	Restr original		Restr a		Restr b		Restr c	
Dist. total 20s	TE4001	1190	Óptimo	1163 <=opt<= 1192		1190	Óptimo	1190	Óptimo
Dist. total 50s		1190	Óptimo	1167 <=opt<= 1190		1190	Óptimo	1190	Óptimo
Relajación lineal		1144.09375		1059.333333		1059.333333		1144.09375	
Nº variables		3321		3160		3321		3321	
Nº restricciones		3360		1803		3360		1720	
Dist. total 20s	TE4002	1103	Óptimo	1093 <=opt<= 1107		1103	Óptimo	1103	Óptimo
Dist. total 50s		1103	Óptimo	1103	Óptimo	1103	Óptimo	1103	Óptimo
Relajación lineal		1075.166667		997.3333333		997.9444444		1075.166667	
Nº variables		3321		3160		3321		3321	
Nº restricciones		3360		1804		3360		1720	
Dist. total 20s	TE4003	1115	Óptimo	1105 <=opt<= 1115		1115	Óptimo	1115	Óptimo
Dist. total 50s		1115	Óptimo	1115	Óptimo	1115	Óptimo	1115	Óptimo
Relajación lineal		1091.145833		1017		1017		1089.895833	
Nº variables		3321		3160		3321		3321	
Nº restricciones		3360		1814		3360		1720	
Dist. total 20s	TC4001	742	Óptimo	742	Óptimo	742	Óptimo	742	Óptimo
Dist. total 50s		742	Óptimo	742	Óptimo	742	Óptimo	742	Óptimo
Relajación lineal		712.5833333		673.6666667		673.6666667		712.5833333	
Nº variables		3321		3321		3321		3321	
Nº restricciones		3360		1720		3360		1720	
Dist. total 20s	TE4002	717	Óptimo	717	Óptimo	717	Óptimo	717	Óptimo
Dist. total 50s		717	Óptimo	717	Óptimo	717	Óptimo	717	Óptimo
Relajación lineal		689.4166667		663.3333333		664.2962963		687.4166667	
Nº variables		3321		3321		3321		3321	
Nº restricciones		3360		1720		3360		1720	
Dist. total 20s	TE4003	716	Óptimo	716	Óptimo	716	Óptimo	716	Óptimo
Dist. total 50s		716	Óptimo	716	Óptimo	716	Óptimo	716	Óptimo
Relajación lineal		684		646		647.7777778		682	
Nº variables		3321		3321		3321		3321	
Nº restricciones		3360		1720		3360		1720	

Q=5									
	Archivo	Restr original		Restr a		Restr b		Restr c	
Dist. total 20s	TE4001	830	Óptimo	807 <=opt<= 835		830	Óptimo	823 <=opt<= 835	
Dist. total 50s		830	Óptimo	811 <=opt<= 835		830	Óptimo	830	Óptimo
Relajación lineal		779.6875		737.6		737.92		779.25	
Nº variables		3321		3160		3321		3160	
Nº restricciones		3360		1796		3360		1756	
Dist. total 20s	TE4002	792	Óptimo	767 <=opt<= 814		792	Óptimo	787 <=opt<= 793	
Dist. total 50s		792	Óptimo	768 <=opt<= 810		792	Óptimo	792	Óptimo
Relajación lineal		744.9		710.8		711.56		744.1	
Nº variables		3321		3160		3321		3160	
Nº restricciones		3360		1780		3360		1747	
Dist. total 20s	TE4003	797	Óptimo	765 <=opt<= 834		797	Óptimo	790 <=opt<= 798	
Dist. total 50s		797	Óptimo	766 <=opt<= 834		797	Óptimo	795 <=opt<= 797	
Relajación lineal		747.05		707		707.16		746.95	
Nº variables		3321		3160		3321		3160	
Nº restricciones		3360		1810		3360		1735	

Q=5									
	Archivo	Restr original		Restr a		Restr b		Restr c	
Dist. total 20s	TC4001	586	Óptimo	572 <=opt<= 586		586	Óptimo	586	Óptimo
Dist. total 50s		586	Óptimo	579 <=opt<= 586		586	Óptimo	586	Óptimo
Relajación lineal		548.1		533.4		533.4		548.1	
Nº variables		3321		3160		3321		3321	
Nº restricciones		3360		1725		3360		1720	
Dist. total 20s	TC4002	578	Óptimo	578	Óptimo	578	Óptimo	578	Óptimo
Dist. total 50s		578	Óptimo	578	Óptimo	578	Óptimo	578	Óptimo
Relajación lineal		548.146875		539.4		540.272		547.325	
Nº variables		3321		3321		3321		3321	
Nº restricciones		3360		1720		3360		1720	
Dist. total 20s	TC4003	577	Óptimo	577	Óptimo	577	Óptimo	577	Óptimo
Dist. total 50s		577	Óptimo	577	Óptimo	577	Óptimo	577	Óptimo
Relajación lineal		545.025		538.4		539.12		543.9	
Nº variables		3321		3321		3321		3321	
Nº restricciones		3360		1720		3360		1720	

Q=10									
	Archivo	Restr original		Restr a		Restr b		Restr c	
Dist. total 20s	TE4001	588 <=opt<= 596		575 <=opt<= 604		578 <=opt<= 604		577 <=opt<= 611	
Dist. total 50s		595 <=opt<= 596		577 <=opt<= 604		583 <=opt<= 598		579 <=opt<= 602	
Relajación lineal		558.7160494		550		550.242		558.666667	
Nº variables		1287		3160		235		3160	
Nº restricciones		1425		1718		2498		1723	
Dist. total 20s	TE4002	573	Óptimo	573	Óptimo	573	Óptimo	573	Óptimo
Dist. total 50s		573	Óptimo	573	Óptimo	573	Óptimo	573	Óptimo
Relajación lineal		546.4074074		538		538		546.4074074	
Nº variables		3321		3321		3321		3321	
Nº restricciones		3360		1720		3360		1720	
Dist. total 20s	TE4003	555 <=opt<= 572		548 <=opt<= 578		555 <=opt<= 578		554 <=opt<= 572	
Dist. total 50s		560 <=opt= 570		550 <=opt<= 575		559 <=opt<= 568		558 <=opt<= 568	
Relajación lineal		532.4444444		520.9		520.9		532.4444444	
Nº variables		1403		3160		2032		3160	
Nº restricciones		1540		1719		2188		1714	
Dist. total 20s	TC4001	498	Óptimo	498	Óptimo	498	Óptimo	498	Óptimo
Dist. total 50s		498	Óptimo	498	Óptimo	498	Óptimo	498	Óptimo
Relajación lineal		469.1777778		467		467		469.0444444	
Nº variables		3321		3321		3321		3321	
Nº restricciones		3360		1720		3360		1720	
Dist. total 20s	TC4002	490	Óptimo	490	Óptimo	490	Óptimo	490	Óptimo
Dist. total 50s		490	Óptimo	490	Óptimo	490	Óptimo	490	Óptimo
Relajación lineal		474.1888889		471.8		472.4		473.0444444	
Nº variables		3321		3321		3321		3321	
Nº restricciones		3360		1720		3360		1720	
Dist. total 20s	TC4003	500	Óptimo	500	Óptimo	500	Óptimo	500	Óptimo
Dist. total 50s		500	Óptimo	500	Óptimo	500	Óptimo	500	Óptimo
Relajación lineal		478.7851852		478		478.28		478.4888889	
Nº variables		3321		3321		3321		3321	
Nº restricciones		3360		1720		3360		1720	

Tabla 7.3: Resultados CMST exacto con demandas unitarias- Formulación de Flujo en redes (Gavish 1985). Base->Nodos. N=41

GAVISH 80 NODOS

Q=5					
	Archivo	Restr original	Restr a	Restr b	Restr c
Dist. total 20s	te80-1	2508 <=opt<= 2591	2443 <=opt<= 2762	2508 <=opt<= 2591	2509 <=opt<= 2634
Dist. total 50s		2514 <=opt<= 2591	2444 <=opt<= 2754	2514 <=opt<= 2591	2510 <=opt<= 2626
Relajación lineal		2412.183447	2266.8	2267.392381	2412.168213
Nº variables		9138	12720	9138	12720
Nº restricciones		9484	6838	9484	6721
Dist. total 20s	te80-2	2501 <=opt<= 2708	2428 <=opt<= 2990	2501 <=opt<= 2708	2502 <=opt<= 2747
Dist. total 50s		2502 <=opt<= 2708	2430 <=opt<= 2974	2502 <=opt<= 2708	2502 <=opt<= 2701
Relajación lineal		2401.735254	2264.6	2265.4	2400.483691
Nº variables		12800	12720	12800	12720
Nº restricciones		13179	6811	13179	6730
Dist. total 20s	te80-3	2567 <=opt<= 2816	2492 <=opt<= 2754	2567 <=opt<= 2816	2567 <=opt<= 2659
Dist. total 50s		2569 <=opt<= 2816	2494 <=opt<= 2748	2569 <=opt<= 2816	2569 <=opt<= 2659
Relajación lineal		2449.65013	2299.2	2302.6	2446.311719
Nº variables		12800	12800	12800	12720
Nº restricciones		13209	13209	13209	6759
Dist. total 20s	tc80-1	1090 <=opt<= 1100	1074 <=opt<= 1168	1090 <=opt<= 1100	1084 <=opt<= 1113
Dist. total 50s		1099 Óptimo	1075 <=opt<= 1114	1099 Óptimo	1085 <=opt<= 1099
Relajación lineal		1060.275	1026.2	1026.2	1060.275
Nº variables		13041	12720	13041	12720
Nº restricciones		13120	6675	13120	6640
Dist. total 20s	tc80-2	1090 <=opt<= 1100	1078 <=opt<= 1164	1090 <=opt<= 1100	1085 <=opt<= 1110
Dist. total 50s		1097 <=opt<= 1100	1078 <=opt<= 1113	1097 <=opt<= 1100	1086 <=opt<= 1104
Relajación lineal		1059.8375	1034.4	1034.4	1059.5875
Nº variables		1998	12720	1998	12720
Nº restricciones		2259	6646	2259	6616
Dist. total 20s	tc80-3	1063 <=opt<= 1073	1046 <=opt<= 1111	1062 <=opt<= 1073	1054 <=opt<= 1104
Dist. total 50s		1068 <=opt<= 1073	1047 <=opt<= 1090	1069 <=opt<= 1073	1055 <=opt<= 1085
Relajación lineal		1024.525	996.6	996.6	1024.275
Nº variables		2620	12720	2620	12720
Nº restricciones		2913	6676	2913	6632

Q=10					
	Archivo	Restr original	Restr a	Restr b	Restr c
Dist. total 20s	te80-1	1582 <=opt<= 1733	1553 <=opt<= 1821	1581 <=opt<= 1733	1582 <=opt<= 1962
Dist. total 50s		1585 <=opt<= 1733	1554 <=opt<= 1753	1585 <=opt<= 1733	1584 <=opt<= 1849
Relajación lineal		1513.538884	1472	1472.13	1513.314815
Nº variables		12800	12720	12800	12720
Nº restricciones		13138	6750	13138	6719
Dist. total 20s	te80-2	1554 <=opt<= 182v	1527 <=opt<= 1856	1554 <=opt<= 1822	1557 <=opt<= 1865
Dist. total 50s		1556 <=opt<= 1794	1530 <=opt<= 1831	1556 <=opt<= 1794	1558 <=opt<= 1751
Relajación lineal		1487.783661	1444.9	1444.9	1487.783661
Nº variables		12800	12720	12800	12720
Nº restricciones		13145	6695	13145	6655
Dist. total 20s	te80-3	1615 <=opt<= 1857	1590 <=opt<= 1808	1615 <=opt<= 1857	1618 <=opt<= 1836
Dist. total 50s		1617 <=opt<= 1857	1592 <=opt<= 1808	1617 <=opt<= 1857	1619 <=opt<= 1798
Relajación lineal		1540.888364	1500	1501.66679	1538.967078
Nº variables		12800	12720	12800	12720
Nº restricciones		13149	6696	13149	6666
Dist. total 20s	tc80-1	877 <=opt<= 890	874 <=opt<= 901	877 <=opt<= 890	876 <=opt<= 898
Dist. total 50s		882 <=opt<= 889	875 <=opt<= 898	882 <=opt<= 889	877 <=opt<= 892
Relajación lineal		867.7777778	863.8	863.8	867.7777778
Nº variables		1914	12720	1914	12720
Nº restricciones		2162	6638	2162	6604
Dist. total 20s	tc80-2	877 Óptimo	876 <=opt<= 884	877 Óptimo	877 Óptimo
Dist. total 50s		877 Óptimo	877 Óptimo	877 Óptimo	877 Óptimo
Relajación lineal		869	864.2	864.2	869
Nº variables		13041	12720	13041	13041

Q=10							
	Archivo	Restr original		Restr a	Restr b		Restr c
Nº restricciones		13120		6579	13120		6640
Dist. total 20s	tc80-3	878	Óptimo	869 <=opt<= 894	876 <=opt<= 878		870 <=opt<= 882
Dist. total 50s		878	Óptimo	871 <=opt<= 880	878	Óptimo	872 <=opt<= 878
Relajación lineal		854.8888889		852.4	852.4		854.8888889
Nº variables		13041		12720	13041		12720
Nº restricciones		13120		6555	13120		6597

Q=20							
	Archivo	Restr original		Restr a	Restr b		Restr c
Dist. total 20s	te80-1	1238 <=opt<= 1291		1226 <=opt<= 1294	1238 <=opt<= 1291		1227 <=opt<= 1281
Dist. total 50s		1243 <=opt<= 1275		1227 <=opt<= 1291	1243 <=opt<= 1275		1230 <=opt<= 1277
Relajación lineal		1182.039873		1172	1172.765789		1181.199785
Nº variables		5996		12720	5996		12720
Nº restricciones		6262		6668	6262		6658
Dist, total 20s	te80-2	1199 <=opt<= 1224		1186 <=opt<= 1317	1199 <=opt<= 1224		1192 <=opt<= 1270
Dist. total 50s		1203 <=opt<= 1224		1188 <=opt<= 1234	1203 <=opt<= 1224		1193 <=opt<= 1236
Relajación lineal		1136.456092		1125.2	1125.403684		1136.3241
Nº variables		2448		12720	2448		12720
Nº restricciones		2721		6665	2721		6627
Dist. total 20s	te80-3	1245 <=opt<= 1267		1238 <=opt<= 1296	1245 <=opt<= 1267		1243 <=opt<= 1267
Dist. total 50s		1250 <=opt<= 1267		1239 <=opt<= 1279	1251 <=opt<= 1267		1244 <=opt<= 1267
Relajación lineal		1184.594182		1174.3	1174.9805		1183.909034
Nº variables		2492		12720	2492		12720
Nº restricciones		2765		6612	2765		6598
Dist. total 20s	tc80-1	834	Óptimo	834	Óptimo	834	Óptimo	834	Óptimo
Dist. total 50s		834	Óptimo	834	Óptimo	834	Óptimo	834	Óptimo
Relajación lineal		823.0526316		822.7	822.7		823.0526316	
Nº variables		13041		13041	13041		13041	
Nº restricciones		13120		6640	13120		6640	
Dist. total 20s	tc80-2	820	Óptimo	820	Óptimo	820	Óptimo	820	Óptimo
Dist. total 50s		820	Óptimo	820	Óptimo	820	Óptimo	820	Óptimo
Relajación lineal		813.2631579		813	813		813.2631579	
Nº variables		13041		13041	13041		13041	
Nº restricciones		13120		6640	13120		6640	
Dist. total 20s	tc80-3	828	Óptimo	828	Óptimo	828	Óptimo	828	Óptimo
Dist. total 50s		828	Óptimo	828	Óptimo	828	Óptimo	828	Óptimo
Relajación lineal		819.0526316		817	819		819.0526316	
Nº variables		13041		13041	13041		13041	
Nº restricciones		13120		6640	13120		6640	

Tabla 7.6: Resultados CMST exacto con demandas unitarias- Formulación de Flujo en redes (Gavish 1985). Base->Nodos. N=81

FORMULACIÓN ALTERNATIVA 40 NODOS

	Archivo	Q=3		Q=5	Q=5	
Dist. total 20s	TE4001	1168 <=opt<= 1196		806 <=opt<= 853	576 <=opt<= 602	
Dist. total 50s		1172 <=opt<= 1194		808 <=opt= 840	579 <=opt<= 602	
Relajación lineal		1059.333333		737.6	550	
Nº variables		3240		3240	3240	
Nº restricciones		1873		1836	1762	
Dist. total 20s	TE4002	1103	Óptimo	767 <=opt<= 822	573	Óptimo
Dist. total 50s		1103	Óptimo	768 <=opt<= 801	573	Óptimo
Relajación lineal		997.3333333		710.8	538	
Nº variables		3362		3362	3362	

	Archivo	Q=3		Q=5		Q=5	
Nº restricciones		1762		1762		1762	
Dist. total 20s	TE4003	1105 <=opt<= 1126		766 <=opt<= 838		548 <=opt<= 582	
Dist. total 50s		1112 <=opt<= 1123		767 <=opt<= 822		551 <=opt<= 570	
Relajación lineal		1017		707		520.9	
Nº variables		3240		3240		3240	
Nº restricciones		1856		1846		1757	
Dist. total 20s	TC4001	742	Óptimo	571 <=opt<= 592		498	Óptimo
Dist. total 50s		742	Óptimo	577 <=opt<= 591		498	Óptimo
Relajación lineal		673.6666667		533.4		467	
Nº variables		3362		3240		3362	
Nº restricciones		1762		1769		1762	
Dist. total 20s	TC4002	717	Óptimo	578	Óptim	490	Óptimo
Dist. total 50s		717	Óptimo	578	Óptimo	490	Óptimo
Relajación lineal		663.333333		539.4		471.8	
Nº variables		3362		3362		3362	
Nº restricciones		1762		1762		1762	
Dist. total 20s	TC4003	716	Óptimo	577	Óptimo	500	Óptimo
Dist. total 50s		716	Óptimo	577	Óptimo	500	Óptimo
Relajación lineal		646		538.4		478	
Nº variables		3362		3362		3362	
Nº restricciones		1762		1762		1762	

Tabla 7.7: Resultados CMST exacto con demandas unitarias- Formulación de Flujo en redes. Base->Nodos. N=41

FORMULACIÓN ALTERNATIVA 80 NODOS

	Archivo	Q=3	Q=5	Q=5
Dist. total 20s	te80-1	2442 <=opt<= 2675	1560 <=opt<= 1796	1240 <=opt<= 133
Dist. total 50s		2443 <=opt<= 2675	1561 <=opt<= 1796	1241 <=opt<= 1320
Relajación lineal		2266.8	1472	1172
Nº variables		12880	12880	12880
Nº restricciones		6900	6784	6694
Dist. total 20s	te80-2	2432 <=opt<= 2938	1528 <=opt<= 1867	1193 <=opt<= 1258
Dist. total 50s		2433 <=opt<= 2872	1532 <=opt<= 1848	1195 <=opt<= 1226
Relajación lineal		2264.6	1444.9	1125.2
Nº variables		12880	12880	12880
Nº restricciones		6924	6811	6730
Dist. total 20s	te80-3	2492 <=opt<= 2755	1592 <=opt<= 1875	1240 <=opt<= 1273
Dist. total 50s		2493 <=opt<= 2709	1594 <=opt<= 1803	1241 <=opt<= 1268
Relajación lineal		2299.2	1500	1174.3
Nº variables		12880	12880	12880
Nº restricciones		6938	6783	6708
Dist. total 20s	tc80-1	1074 <=opt<= 1187	874 <=opt<= 898	834 Óptimo
Dist. total 50s		1076 <=opt<= 1115	875 <=opt<= 890	834 Óptimo
Relajación lineal		1026.2	863.8	822.7
Nº variables		12880	12880	13122
Nº restricciones		6740	6645	6722
Dist. total 20s	tc80-2	1079 <=opt<= 1126	876 <=opt<= 885	820 Óptimo
Dist. total 50s		1080 <=opt<= 1110	877 Óptimo	820 Óptimo
Relajación lineal		1034.4	864.2	823
Nº variables		12880	13122	13122
Nº restricciones		6757	6722	6722
Dist. total 20s	tc80-3	1047 <=opt<= 1139	869 <=opt<= 886	828 Óptimo
Dist. total 50s		1047 <=opt<= 1099	871 <=opt<= 880	828 Óptimo
Relajación lineal		996.6	852.4	819
Nº variables		12880	12880	13122
Nº restricciones		6741	6644	6722

	Archivo	Q=3	Q=5	Q=5

Tabla 7.8: Resultados CMST exacto con demandas unitarias- Formulación de Flujo en redes. Base->Nodos. N=81

7.1.2. 2n restricciones

40 NODOS

	Archivo	Q=3		Q=5		Q=5	
Dist. total 20s	TE4001	1190	Óptimo	830	Óptimo	589 <=opt<= 598	
Dist. total 50s		1190	Óptimo	830	Óptimo	596	Óptimo
Relajación lineal		1059.333333		779.6875		558.7160494	
Nº variables		3403		6765		15170	
Nº restricciones		80		80		80	
Dist. total 20s	TE4002	1103	Óptimo	792	Óptimo	573	Óptimo
Dist. total 50s		1103	Óptimo	792	Óptimo	573	Óptimo
Relajación lineal		1075.166667		744.9		546.4074074	
Nº variables		3403		6765		15170	
Nº restricciones		80		80		80	
Dist. total 20s	TE4003	1115	Óptimo	797	Óptimo	557 <=opt<= 583	
Dist. total 50s		1115	Óptimo	797	Óptimo	559 <=opt<= 573	
Relajación lineal		1091.145833		747.05		532.4444	
Nº variables		3403		6765		7295	
Nº restricciones		80		80		231	
Dist. total 20s	TC4001	742	Óptimo	586	Óptimo	498	Óptimo
Dist. total 50s		742	Óptimo	586	Óptimo	498	Óptimo
Relajación lineal		712.5833333		548.1		469.17778	
Nº variables		3403		6765		15170	
Nº restricciones		80		80		80	
Dist. total 20s	TC4002	717	Óptimo	578	Óptim	490	Óptimo
Dist. total 50s		717	Óptimo	578	Óptimo	490	Óptimo
Relajación lineal		689.4166667		548.146875		474.18889	
Nº variables		3403		6765		15170	
Nº restricciones		80		80		80	
Dist. total 20s	TC4003	716	Óptimo	577	Óptimo	500	Óptimo
Dist. total 50s		716	Óptimo	577	Óptimo	500	Óptimo
Relajación lineal		684		545.025		478.7851852	
Nº variables		3403		6765		15170	
Nº restricciones		80		80		80	

Tabla 7.9: Resultados CMST exacto con demandas unitarias- Formulación 2n restricciones. Base->Nodos. N=41

80 NODOS

	Archivo	Q=3		Q=5	Q=5
Dist. total 20s	te80-1	2544	Óptimo	1635 <=opt<= 2648	1214 <=opt<= 2222
Dist. total 50s		1190	Óptimo	1635 <=opt<= 2648	1214 <=opt<= 2222
Relajación lineal		2412.183447		1513.538884	1182.039873
Nº variables		26325		57680	121680
Nº restricciones		160		543	511
Dist. total 20s	te80-2	2538 <=opt<= 2627		1602 <=opt<= 7023	1183 <=opt<= 1492
Dist. total 50s		2539 <=opt<= 2580		1605 <=opt<= 7023	1183 <=opt<= 1492
Relajación lineal		2401.735254		1487.783661	1136.456092
Nº variables		18267		57680	121680
Nº restricciones		376		551	457

	Archivo	Q=3	Q=5	Q=5
Dist. total 20s	te80-3	2599 <=opt<= 2790	1665 <=opt<= 1713	1244 <= opt<= 1471
Dist. total 50s		2599 <=opt<= 2676	1665 <=opt<= 1713	1244 <=opt<= 1471
Relajación lineal		2449.65013	1540.888364	1184.594182
Nº variables		25680	57680	121680
Nº restricciones		398	583	121680
Dist. total 20s	tc80-1	1099 Óptimo	875 <=opt<= 4086	829 <=opt<= 4177
Dist. total 50s		1099 Óptimo	876 <=opt<= 4086	829 <=opt<= 4177
Relajación lineal		1060.275	867.77778	823.0526316
Nº variables		26325	57680	121680
Nº restricciones		160	334	275
Dist. total 20s	tc80-2	1093 <=opt<= 1191	875 <=opt<= 3998	817 <=opt<= 1295
Dist. total 50s		1095 <=opt<= 1100	876 <=opt<= 3998	817 <=opt<= 1295
Relajación lineal		1059.8375	869	813.2631579
Nº variables		25680	57680	121680
Nº restricciones		303	320	311
Dist. total 20s	tc80-3	1073 Óptimo	870 <=opt<= 1475	829 <=opt<= 1163
Dist. total 50s		1073 Óptimo	871 <=opt<= 1192	829 <=opt<= 1163
Relajación lineal		1024.525	854.88889	819.0526316
Nº variables		26325	57680	121680
Nº restricciones		160	372	286

Tabla 7.10: Resultados CMST exacto con demandas unitarias- Formulación 2n restricciones. Base->Nodos. N=81

7.1.3. Multiproducto

40 NODOS

	Archivo	Q=3	Q=5	Q=5
Dist. total 20s	TE4001	1070 <=opt<= 1283	751 <=opt<= 949	557 <=opt<= 683
Dist. total 50s		1084 <=opt<= 1257	755 <=opt<= 879	562 <=opt<= 683
Relajación lineal		1059.3333	737.6	550
Nº variables		67241	67241	67241
Nº restricciones		3329	3495	3413
Dist. total 20s	TE4002	1016 <=opt<= 1264	720 <=opt<= 951	556 <=opt<= 626
Dist. total 50s		1018 <=opt<= 1264	723 <=opt<= 836	557 <=opt<= 626
Relajación lineal		997.33333	710.8	538
Nº variables		67241	67241	67241
Nº restricciones		3461	3616	3348
Dist. total 20s	TE4003	1017 <=opt<= 1252	714 <=opt<= 963	521 <=opt<= 947
Dist. total 50s		1039 <=opt<= 1205	719 <=opt<= 963	534 <=opt<= 620
Relajación lineal		1017	707	520.9
Nº variables		67241	67241	67241
Nº restricciones		3326	3320	3344
Dist. total 20s	TC4001	693 <=opt<= 759	550 <=opt<= 606	495 <=opt<= 498
Dist. total 50s		695 <=opt<= 759	552 <=opt<= 606	498 Óptimo
Relajación lineal		673.66667	533.4	467
Nº variables		67241	67241	68921
Nº restricciones		3318	3320	3361
Dist. total 20s	TC4002	681 <=opt <= 734	547 <=opt<= 608	481 <=opt<= 490
Dist. total 50s		682 <=opt<= 734	551 <=opt<= 608	484 <=opt<= 490
Relajación lineal		663.33333	539.4	471.8
Nº variables		67241	67241	48675
Nº restricciones		3539	3550	3030
Dist. total 20s	TC4003	668 <=opt<= 746	555 <=opt<= 604	498 <=opt<= 509
Dist. total 50s		671 <=opt<= 728	559 <=opt<= 577	500 Óptimo
Relajación lineal		646	538.4	478
Nº variables		67241	67241	68921

	Archivo	Q=3	Q=5	Q=5
Nº restricciones		3522	3337	3361

Tabla 7.11: Resultados CMST exacto con demandas unitarias- Formulación Multiproducto. Base->Nodos. N=41

80 NODOS

	Archivo	Q=3	Q=5	Q=5
Dist. total 20s	te80-1	2187 <=opt<= 12934	1472 <=opt<= 13164	1172 <=opt<= 13281
Dist. total 50s		2189 <=opt= 10699	1475 <=opt<= 2691	1177 <=opt<= 1988
Relajación lineal		2186.4	1472	1172
Nº variables		524881	524881	524881
Nº restricciones		13244	13120	13115
Dist. total 20s	te80-2	2272 <=opt<= 7134	1456 <=opt<= 4024	1126 <=opt<= 12715
Dist. total 50s		2272 <=opt<= 4899	1456 <=opt<= 3265	1146 <=opt<= 6136
Relajación lineal		2264.6	1444.9	1125.2
Nº variables		524881	524881	524881
Nº restricciones		13284	13128	13163
Dist. total 20s	te80-3	2300 <=opt<= 3595	1500 <=opt<= 12766	1175 <=opt<= 13003
Dist. total 50s		2316 <=opt<= 3399	1523 <=opt<= 4105	1211 <=opt<= 3597
Relajación lineal		2299.2	1500	1174.3
Nº variables		524881	524881	524881
Nº restricciones		13242	13149	13147
Dist. total 20s	tc80-1	1027 <=opt<= 2060	864 <=opt<= 4596	823 <=opt<= 4796
Dist. total 50s		1027 <=opt<= 1674	864 <=opt<= 2986	823 <=opt<= 1027
Relajación lineal		1026.2	863.8	822.7
Nº variables		524881	524881	524881
Nº restricciones		13106	13047	12999
Dist. total 20s	tc80-2	1035 <=opt<= 2676	865 <=opt<= 5003	813 <=opt<= 4881
Dist. total 50s		1035 <=opt<= 1592	865 <=opt<= 954	814 <=opt<= 982
Relajación lineal		1034.4	864.2	813
Nº variables		524881	524881	524881
Nº restricciones		13111	13080	13024
Dist. total 20s	tc80-3	997 <=opt<= 4179	861 <=opt<= 1063	819 <=opt<= 2108
Dist. total 50s		997 <=opt<= 1271	861 <=opt<= 1020	819 <=opt<= 858
Relajación lineal		996.6	852.4	819
Nº variables		524881	524881	524881
Nº restricciones		13106	13237	13005

Tabla 7.12: Resultados CMST exacto con demandas unitarias- Formulación Multiproducto. Base->Nodos. N=81

7.2. Heurísticas y mejoras

7.2.1. Prim secuencial

40 NODOS

Archivo	Capacidad	Prim	Tiempo	Shift	Tiempo	Exchange	Tiempo
TE4001	Q=3	1291	0.002	1291	0.009	1278	0.006
	Q=5	900	0.002	900	0.007	900	0.003
	Q=10	658	0.004	658	0.005	658	0.004
TE4002	Q=3	1230	0.003	1230	0.001	1196	0.005
	Q=5	940	0.003	940	0.001	940	0.004
	Q=10	615	0.004	615	0.004	615	0.003
TE4003	Q=3	1221	0.001	1221	0.006	1201	0.005

Archivo	Capacidad	Prim	Tiempo	Shift	Tiempo	Exchange	Tiempo
	Q=5	858	0.003	858	0.003	858	0.004
	Q=10	608	0.004	608	0.002	608	0.003
TC4001	Q=3	857	0.003	857	0.011	834	0.011
	Q=5	656	0.003	656	0.007	645	0.007
	Q=10	545	0.006	545	0.007	545	0.008
TC4002	Q=3	841	0.002	841	0.004	833	0.009
	Q=5	714	0.003	714	0.006	707	0.006
	Q=10	600	0.004	600	0.005	581	0.005
TC4003	Q=3	759	0.002	759	0.01	758	0.006
	Q=5	672	0.003	672	0.003	624	0.005
	Q=10	586	0.004	586	0.005	586	0.004

Tabla 7.13: Resultados Heurística Prim y algoritmos de mejora con demandas unitarias. Base->Nodos. N=41.

80 NODOS

Archivo	Capacidad	Prim	Tiempo	Shift	Tiempo	Exchange	Tiempo
te80-1	Q=5	2740	0.009	2740	0.008	2740	0.028
	Q=10	1792	0.013	1792	0.011	1792	0.014
	Q=20	1434	0.024	1434	0.021	1424	0.013
te80-2	Q=5	2638	0.01	2638	0.018	2638	0.021
	Q=10	1743	0.014	1743	0.041	1743	0.015
	Q=20	1353	0.024	1353	0.025	1353	0.014
te80-3	Q=5	2747	0.01	2747	0.019	2747	0.023
	Q=10	1976	0.014	1976	0.042	1976	0.023
	Q=20	1486	0.024	1486	0.015	1486	0.017
tc80-1	Q=5	1290	0.01	1290	0.009	1290	0.029
	Q=10	1076	0.018	1076	0.032	1076	0.027
	Q=20	954	0.026	954	0.094	954	0.017
tc80-2	Q=5	1329	0.01	1329	0.025	1307	0.027
	Q=10	1036	0.013	1036	0.018	1063	0.022
	Q=20	941	0.024	941	0.033	941	0.02
tc80-3	Q=5	1303	0.01	1303	0.023	1288	0.046
	Q=10	1132	0.018	1132	0.012	1132	0.025
	Q=20	920	0.025	920	0.022	920	0.025

Tabla 7.14: Resultados Heurísticas Prim secuencial y algoritmos de mejora con demandas unitarias. Base->Nodos. N=81.

7.2.2. Esau Williams

40 NODOS

Archivo	Capacidad	EW	Tiempo	Shift	Tiempo	Exchange	Tiempo	GRASP S	Tiempo	GRASP E	Tiempo
TE4001	Q=3	1198	0.79	1198	0.003	1198	0.005	1271	15.651	1266	15.28
	Q=5	877	0.717	877	0.003	877	0.003	922	29.235	922	30.268
	Q=10	640	0.41	640	0.003	640	0.002	680	23.599	680	23.605
TE4002	Q=3	1141	0.868	1141	0.003	1141	0.006	1175	32.484	1175	32.499
	Q=5	834	0.593	834	0.008	834	0.004	872	28.509	872	28.513
	Q=10	618	0.485	618	0.004	618	0.003	642	22.931	642	23.012
TE4003	Q=3	1148	0.856	1148	0.014	1148	0.013	1192	35.765	1192	35.574
	Q=5	841	0.804	841	0.005	841	0.041	882	33.272	882	33.728
	Q=10	616	0.504	616	0.008	616	0.003	618	27.869	618	26.696
TC4001	Q=3	768	0.296	768	0.003	768	0.005	779	15.953	779	15.545
	Q=5	611	0.162	611	0.005	611	0.005	621	12.854	621	12.602
	Q=10	520	0.103	520	0.006	520	0.004	540	8.825	540	8.586

Archivo	Capacidad	EW	Tiempo	Shift	Tiempo	Exchange	Tiempo	GRASP S	Tiempo	GRASP E	Tiempo
TC4002	Q=3	748	0.069	748	0.002	748	0.001	743	8.972	743	8.733
	Q=5	608	0.052	608	0.001	608	0.001	611	10.414	609	10.149
	Q=10	522	0.148	522	0.004	522	0.004	520	7.356	520	7.566
TC4003	Q=3	729	0.325	729	0.005	729	0.006	735	15.266	735	15.667
	Q=5	597	0.179	597	0.004	597	0.005	602	10.642	602	10.516
	Q=10	529	0.107	529	0.003	529	0.004	528	5.569	528	5.036

Tabla 7.15: Resultados Heurísticas Esau Williams y algoritmos de mejora.
Base->Nodos. N=41. GRASP 50 iteraciones.

80 NODOS

Archivo	Capacidad	EW	Tiempo	Shift	Tiempo	Exchange	Tiempo	GRASP S	Tiempo	GRASP E	Tiempo
te80-1	Q=5	2627	12.938	2627	0.03	2627	0.018	2803	431.32	2787	431.885
	Q=10	1832	8.448	1832	0.018	1832	0.013	1948	386.864	1948	386.936
	Q=20	1411	3.495	1411	0.018	1411	0.015	1512	277.675	1512	277.712
te80-2	Q=5	2638	12.127	2638	0.02	2638	0.018	2808	430.265	2802	430.797
	Q=10	1762	10.04	1762	0.036	1762	0.012	1911	312.032	1911	312.259
	Q=20	1317	4.163	1317	0.046	1317	0.012	1453	254.864	1453	254.461
te80-3	Q=5	2734	13.424	2734	0.011	2734	0.022	2854	451.78	2854	451.348
	Q=10	1830	11.143	1830	0.016	1830	0.019	1883	341.265	1883	341.347
	Q=20	1357	6.277	1357	0.012	1357	0.014	1453	254.651	1453	254.684
tc80-1	Q=5	1171	3.041	1171	0.008	1171	0.02	1221	102.32	1221	102.24
	Q=10	939	0.965	939	0.051	939	0.018	977	61.357	977	61.765
	Q=20	860	0.703	860	0.016	860	0.016	884	77.91	884	77.92
tc80-2	Q=5	1178	3.286	1178	0.006	1178	0.019	1226	83.492	1226	83.351
	Q=10	921	1.773	921	0.009	921	0.022	957	68.642	957	68.859
	Q=20	846	0.867	846	0.008	846	0.018	895	75.235	895	75.614
tc80-3	Q=5	1134	2.986	1134	0.026	1134	0.019	1175	82.321	1174	89.196
	Q=10	941	2.306	941	0.034	941	0.024	940	92.624	940	92.78
	Q=20	856	0.564	856	0.019	856	0.02	872	57.632	875	57.115

Tabla 7.16: Resultados Heurísticas Esau Williams y algoritmos de mejora.
Base->Nodos. N=81. GRASP 50 iteraciones.

7.2.3. Esau Williams modificado

40 NODOS

Archivo	Capacidad	δ	EWM	Tiempo	Shift	Tiempo	Exchange	Tiempo	GRASP S	Tiempo	GRASP E	Tiempo
TE4001	Q=3	0.25	1198	0.858	1198	0.004	1198	0.005	1222	36.56	1222	36.64
		0.5	1201	0.827	1201	0.003	1201	0.005	1219	40.817	1219	40.835
		0.75	1202	0.817	1202	0.003	1202	0.004	1208	40.254	1208	40.198
		1	1201	0.82	1201	0.005	1201	0.005	1214	34.310	1213	34.316
	Q=5	0.25	876	0.72	876	0.005	876	0.004	886	33.86	886	33.951
		0.5	845	0.716	845	0.005	845	0.003	884	33.754	884	33.763
		0.75	871	0.753	871	0.005	871	0.004	881	33.39	881	33.384
		1	859	0.749	859	0.005	859	0.002	922	32.64	922	32.647
	Q=10	0.25	646	0.47	646	0.004	646	0.002	675	27.86	675	27.868
		0.5	657	0.706	657	0.005	657	0.004	655	30.465	655	30.473
		0.75	661	0.563	661	0.004	661	0.003	683	30.22	683	30.223
		1	643	0.688	643	0.006	643	0.003	702	28.814	702	28.8
TE4002	Q=3	0.25	1130	0.8	1130	0.006	1130	0.005	1142	38.84	1142	38.86
		0.5	1135	0.871	1135	0.005	1135	0.006	1140	40.376	1140	40.365
		0.75	1155	0.795	1155	0.007	1155	0.006	1127	38.486	1127	38.49
		1	1140	0.826	1140	0.007	1140	0.005	1114	35.139	1114	35.141
	Q=5	0.25	821	0.694	821	0.009	821	0.002	840	32.73	840	32.739

Archivo	Capacidad	δ	EWM	Tiempo	Shift	Tiempo	Exchange	Tiempo	GRASP S	Tiempo	GRASP E	Tiempo
		0.5	825	0.715	825	0.009	825	0.005	841	33.88	841	33.87
		0.75	836	0.677	836	0.01	834	0.004	829	30.715	829	30.71
		1	851	0.664	851	0.008	851	0.004	854	29.08	854	29.05
	Q=10	0.25	611	0.412	611	0.01	611	0.003	647	25.706	647	25.7
		0.5	612	0.533	612	0.008	612	0.003	636	28.86	636	28.87
		0.75	616	0.597	616	0.009	616	0.004	657	29.237	657	29.24
		1	632	0.621	632	0.009	630	0.003	669	29.42	669	29.4
TE4003	Q=3	0.25	1135	0.967	1135	0.008	1135	0.006	1179	42.513	1179	42.517
		0.5	1135	0.995	1135	0.003	1135	0.006	1165	42.39	1165	42.384
		0.75	1152	0.942	1152	0.003	1152	0.01	1143	42.74	1143	42.738
		1	1151	0.908	1151	0.014	1151	0.007	1140	38.198	1140	38.2
	Q=5	0.25	809	0.779	809	0.011	809	0.004	852	33.53	852	33.528
		0.5	809	0.678	809	0.002	809	0.003	843	35.06	843	35.071
		0.75	839	0.792	839	0.002	839	0.004	845	34.98	845	34.986
		1	813	0.851	813	0.003	813	0.003	841	34.87	841	34.865
	Q=10	0.25	612	0.48	612	0.003	612	0.003	633	29.73	633	29.734
		0.5	617	0.426	617	0.003	617	0.004	611	30.098	611	30.1
		0.75	614	0.689	614	0.002	614	0.003	643	32.77	643	32.79
		1	612	0.706	612	0.003	612	0.003	656	3.07	656	3.073
TC4001	Q=3	0.25	760	0.33	760	0.007	760	0.005	790	16.65	790	16.664
		0.5	759	0.315	759	0.006	759	0.005	777	15.537	777	15.55
		0.75	757	0.361	757	0.004	757	0.005	767	14.9	767	14.92
		1	757	0.324	757	0.006	757	0.005	775	14.582	775	14.58
	Q=5	0.25	611	0.004	611	0.005	611	0.004	626	12.65	626	12.648
		0.5	603	0.241	603	0.007	603	0.003	631	11.07	631	11.01
		0.75	597	0.233	597	0.003	597	0.004	626	11.3	626	11.293
		1	600	0.249	596	0.007	596	0.004	789	14.41	789	14.4
	Q=10	0.25	520	0.182	516	0.007	516	0.004	538	10.59	538	10.6
		0.5	524	0.217	524	0.003	524	0.004	552	10.26	552	10.27
		0.75	516	0.227	516	0.003	516	0.003	554	10.22	554	10.22
		1	524	0.251	524	0.004	524	0.004	576	11.01	576	11.012
TC4002	Q=3	0.25	728	0.294	728	0.003	728	0.005	755	15.309	755	15.31
		0.5	745	0.31	745	0.003	745	0.005	738	14.66	738	14.67
		0.75	731	0.267	731	0.003	731	0.004	739	13.24	739	13.25
		1	733	0.294	733	0.004	733	0.005	739	12.8	739	12.81
	Q=5	0.25	612	0.254	612	0.01	612	0.004	607	13.477	607	13.48
		0.5	596	0.272	596	0.004	596	0.004	618	12.518	618	12.516
		0.75	596	0.281	596	0.004	596	0.004	621	12.086	621	12.09
		1	598	0.282	598	0.005	598	0.006	618	11.64	618	11.642
	Q=10	0.25	515	0.224	515	0.006	515	0.007	525	10.921	525	10.92
		0.5	521	0.215	521	0.004	521	0.005	535	10.167	535	10.16
		0.75	509	0.258	509	0.004	509	0.004	571	11.014	571	11.02
		1	521	0.278	521	0.004	521	0.005	544	10.95	544	10.958
TC4003	Q=3	0.25	739	0.286	739	0.003	739	0.006	741	17.017	741	17.0183
		0.5	724	0.261	724	0.003	724	0.005	735	16.117	735	16.12
		0.75	738	0.367	738	0.005	738	0.007	741	15.85	741	15.823
		1	728	0.282	728	0.002	728	0.007	733	14.077	733	14.081
	Q=5	0.25	590	0.213	590	0.002	590	0.004	595	11.082	595	11.08
		0.5	608	0.209	608	0.003	608	0.004	603	10.99	603	10.99
		0.75	605	0.236	605	0.003	605	0.005	609	10.096	609	10.099
		1	590	0.257	590	0.004	590	0.004	629	10.086	629	10.085
	Q=10	0.25	522	0.171	522	0.008	522	0.004	544	9.94	544	9.926
		0.5	528	0.202	528	0.003	528	0.004	542	9.72	542	9.72
		0.75	528	0.227	528	0.003	528	0.004	553	9.75	553	9.754
		1	537	0.259	537	0.003	537	0.005	576	9.94	576	9.943

Tabla 7.17: Resultados Heurísticas Esau Williams modificado y algoritmos de mejora. Base->Nodos. Base->Nodos. N=41. GRASP 50 iteraciones. labellongEWM40

80 NODOS

Archivo	Capacidad	δ	EW	Tiempo	Shift	Tiempo	Exchange	Tiempo	GRASP S	Tiempo	GRASP E	Tiempo
te80-1	Q=5	0.25	2635	11.389	2635	0.009	2631	0.012	2745	565.59	2745	565.94
		0.5	2619	10.749	2619	0.012	2619	0.015	2677	476.468	2677	476.471
		0.75	2616	11.541	2616	0.0015	2616	0.009	2702	445.578	2702	445.565
		1	2620	11.968	2620	0.014	2620	0.022	272	506.578	2722	506.58
	Q=10	0.25	1769	8.146	1769	0.014	1769	0.015	1894	433.96	1894	433.9
		0.5	1819	9.284	1819	0.014	1819	0.012	1893	435.29	1893	435.288
		0.75	1783	10.346	1783	0.015	1783	0.016	1855	477.855	1855	477.85
		1	1775	10.594	1775	0.015	1775	0.016	1894	489	1894	489.03
	Q=20	0.25	1436	6.261	1436	0.015	1436	0.014	1483	302.16	1483	302.164
		0.5	1484	8.903	1453	0.039	1453	0.013	1513	360.501	1513	360.5
		0.75	1475	9.022	1475	0.016	1475	0.015	1582	444.827	1582	444.824
		1	1476	11.059	1453	0.016	1453	0.013	1628	476.343	1628	476.34
te80-2	Q=5	0.25	2648	10.262	2648	0.022	2648	0.02	2700	580.338	2700	580.334
		0.5	2649	9.45	2649	0.015	2649	0.025	2656	602.1	2676	602
		0.75	2642	11.959	2642	0.028	2642	0.022	2632	600.74	2636	600.7
		1	2638	11.461	2638	0.038	2638	0.019	2711	539.74	2714	539.743
	Q=10	0.25	1737	9.436	1737	0.046	1737	0.017	1773	423.23	1773	423
		0.5	1739	9.923	1739	0.046	1739	0.016	1813	467.14	1813	467.1
		0.75	1749	8.296	1749	0.02	1749	0.014	1819	507.62	1824	507.615
		1	1769	11.246	1769	0.044	1769	0.015	1882	476.642	1882	476.56
	Q=20	0.25	1350	6.162	1350	0.045	1350	0.014	1396	310.97	1406	310.64
		0.5	1340	8.69	1340	0.052	1340	0.015	1423	412.9	1447	412.03
		0.75	1340	9.432	1340	0.035	1340	0.011	1480	466.9	1480	466.84
		1	1348	11.13	1348	0.052	1348	0.017	1600	487.82	1596	487.03
te80-3	Q=5	0.25	2735	13.317	2735	0.012	2735	0.014	2773	565.75	2773	565.7
		0.5	2741	12.644	2741	0.012	2741	0.023	2720	599.97	2720	599.963
		0.75	2737	12.736	2737	0.012	2737	0.023	2725	602.675	2725	602.68
		1	2735	12.637	2735	0.012	2735	0.019	2742	593.61	2742	593.615
	Q=10	0.25	1827	11.809	1827	0.016	1827	0.015	1840	481.2	1840	481.23
		0.5	1824	11.043	1824	0.011	1824	0.016	1834	517.63	1834	517.632
		0.75	1819	11.75	1819	0.012	1819	0.018	1875	553.23	1875	553.231
		1	1819	11.766	1819	0.012	1819	0.015	1925	569.96	1925	569.948
	Q=20	0.25	1393	4.227	1393	0.006	1393	0.007	1418	399.67	1418	399.668
		0.5	1414	9.183	1412	0.016	1412	0.015	1423	513.492	1423	513.49
		0.75	1380	10.929	1380	0.012	1380	0.014	1450	522.903	1456	522.9
		1	1369	12.121	1369	0.016	1369	0.014	1524	480.19	1524	480.188
tc80-1	Q=5	0.25	1153	3.051	1153	0.016	1153	0.022	1201	150.387	1201	150.388
		0.5	1168	3.496	1168	0.019	1168	0.024	1190	135.53	1190	135.538
		0.75	1159	3.645	1159	0.015	1159	0.024	1198	167.29	1198	167.295
		1	1164	3.238	1164	0.007	1164	0.013	1209	155.57	1209	155.581
	Q=10	0.25	914	1.908	914	0.016	914	0.017	979	122.65	979	122.651
		0.5	930	2.953	930	0.015	930	0.017	989	124.6	989	124.6
		0.75	918	3.135	918	0.014	918	0.018	990	143.8	990	143.801
		1	933	3.022	933	0.009	933	0.012	1030	156.54	1030	156.54
	Q=20	0.25	866	2.067	866	0.019	866	0.017	898	110.73	898	110.718
		0.5	858	2.484	858	0.012	858	0.013	932	122.65	932	122.65
		0.75	870	3.012	870	0.015	870	0.016	961	148.35	961	148.351
		1	870	2.962	870	0.031	870	0.015	1104	150.19	1120	150.2
tc80-2	Q=5	0.25	1140	3.839	1140	0.03	1140	0.02	1193	179.98	1193	179.981
		0.5	1159	3.579	1159	0.029	1159	0.03	1185	177.09	1185	177.09
		0.75	1155	3.675	1155	0.008	1155	0.024	1184	155.17	1184	155.17
		1	1135	3.748	1135	0.008	1135	0.023	1192	180.331	1192	180.335
	Q=10	0.25	925	2.399	925	0.008	925	0.02	962	125.57	962	125.48
		0.5	905	2.641	905	0.015	905	0.018	943	133.14	943	133.142
		0.75	915	3.215	915	0.03	915	0.016	1010	152.39	1010	152.4
		1	917	3.345	917	0.008	917	0.019	1008	139.22	1008	139.231
	Q=20	0.25	834	1.834	834	0.009	834	0.019	897	112.31	897	112.3
		0.5	844	2.73	840	0.029	840	0.018	902	139.48	902	139.473

Archivo	Capacidad	δ	EW	Tiempo	Shift	Tiempo	Exchange	Tiempo	GRASP S	Tiempo	GRASP E	Tiempo
		0.75	858	3.171	858	0.016	858	0.02	918	156.44	918	156.44
		1	834	3.637	834	0.008	834	0.022	1000	156.88	1000	156.898
tc80-3	Q=5	0.25	1115	3.073	1115	0.019	1115	0.021	1164	174.429	1164	174.429
		0.5	1116	3.825	1116	0.014	1116	0.035	1161	159.54	1161	159.563
		0.75	1110	3.382	1110	0.018	1110	0.019	1208	151.83	1208	151.8
		1	1110	3.505	1110	0.043	1110	0.028	1155	159.35	1155	159.353
	Q=10	0.25	915	1.898	915	0.033	915	0.014	967	118.077	967	118.0
		0.5	932	2.628	932	0.018	932	0.018	990	120.028	990	120
		0.75	941	2.835	937	0.02	937	0.018	1005	115.24	1005	115.243
		1	940	3.274	936	0.016	936	0.015	996	130.77	996	130.77
	Q=20	0.25	858	1.856	858	0.018	858	0.018	917	101.8	917	101.8
		0.5	852	2.619	852	0.025	852	0.015	892	111.78	892	111.771
		0.75	860	3.008	860	0.02	860	0.017	959	127.07	959	127.068
		1	862	3.575	862	0.028	862	0.017	979	109.14	979	109.138

Tabla 7.18: Resultados Heurísticas Esau Williams y algoritmos de mejora.
Base->Nodos. N=81. GRASP 50 iteraciones.

Capítulo 8

Conclusiones

A la vista de los resultados, se puede comprobar que los métodos exactos dan mejores soluciones, en muchos casos alcanzando el óptimo global, objetivo que no se logra usando técnicas heurísticas. Además, las cotas que se ofrecen en los problemas factibles no óptimos son bastante ajustadas, en ocasiones incluso dejando fuera valores totales obtenidos con Prim o Esau Williams. Destacar también que muchos problemas que no alcanzan el óptimo cuando se establece el límite de tiempo en 20 segundos, sí lo hacen cuando este se aumenta a 50 segundos.

En cuanto al modelo de 2n restricciones, señalar que, si se encuentra el óptimo absoluto, funciona bastante bien ya que se reducen el número de restricciones respecto a otros modelos exactos planteados y, por lo tanto, no consume tantos recursos. En cambio, si el problema es factible pero no encuentra el óptimo, el número de restricciones aumenta y usa bastantes recursos tanto computacionales como temporales.

En cuanto a las técnicas heurísticas, en general son más rápidas a la hora de obtener resultados, pero en ningún caso se ha encontrado el árbol solución óptimo.
Como era previsible, el algoritmo de Prim modificado es el que peores resultados ha dado de las heurísticas de base, por debajo de las dos versiones planteadas de Esau-Williams.
El algoritmo de Esau-Williams original demuestra funcionar bastante bien, aunque prácticamente en todos los casos, hay, al menos, un valor de δ que hace que el algoritmo de Esau-Williams modificado aporte una solución mejor. En cuanto a qué valor debe tomar δ no se saca una conclusión en claro. Si bien es cierto que cuando toma el valor 0.25 la solución mejora en la mayoría de los casos, no siempre se obtiene la mayor mejora con este valor si no con otros de δ.

Respecto a las heurísticas de mejora, señalar que no han sido la mejor elección ya que al efectuarse sobre los nodos finales no permiten muchos cambios significantes en la solución. Además, muchas veces las propias restricciones del árbol impiden movimientos (p.ej. si se tiene 40 nodos y Q es 5 o 10, en muchas ocasiones se van a formar subárboles al límite de la capacidad fijada lo que no va dejar espacio a que se produzcan movimientos de tipo *shift*) que implican la inclusión de nodos en subárboles.

Para terminar, las metaheurísticas dan bastante malos resultados, generalmente incluso peores que los obtenidos únicamente con las heurísticas de base. Esto no es culpa del algoritmo GRASP en sí, si no de una serie de sucesos: al aleatorizar el algoritmo heurístico de la fase de contrucción se está dando pie a que no se elija el mejor enlace que de el mejor coste ciego, si esto se repite

en cada paso de formación del árbol, la solución obtenidda puede distar bastante de la que se desearía. Esto sumado a la mala elección de las heurísticas de mejora implementadas que, como se ha dicho anteriormente, no producen grandes cambios sobre la solución de partida, hace que no se obtengan resultados interesantes.
Por último señalar que, aunque no es la tónica general, los problemas de 80 nodos tardan un tiempo considerable en ejecutarse al aplicar metaheurísticas.

Bibliografía

[1] .Vaca ”Estructuras de datos Informática UVa"

[2] B. Gavish “Topological design of telecommunication networks- Local access design methods.

[3] George Nemhauser Laurence Wolsey ”Integer and Combinatorial Optimization

[4] L. Gouveia “A comparison of directed formulations for the capacitated minimal spanning tree problem."

[5] L.Gouveia “A 2n constraint formulation for the capacitated minimal spanning tree problem."

[6] G. Clarke and J. Wright “Scheduling of vehicles from a central depot to a number of delivery points.”

[7] S. Voß “Capacitated minimum spaning trees."

[8] T Oncan and IK Altınel “Parametric enhancements of the Esau–Williams heuristic for the capacitated minimum spanning tree problem."

[9] R. Jothi y B. Raghavachari "Design of local access networks."

[10] T. Oncan Atinel ”Parametric enancements of the Esau-Williams heuristic for the capacitated minimum spanning tree problem."

[11] R.Jothi B. Raghavachari ”Revisiting Esau-EWilliam's Algorithm: On the Design of Local Access Networks."

[12] Amberg A, Domschke W, Voss S “Capacitated minimum spanning trees: Algorithms using intelligent search."

[13] Sharaiha, Y. M., M. Gendreau, G. Laporte, and I. H. Osman. “A tabu search algorithm for the capacitated shortest spanning tree problem."

[14] Ahuja, R.K., J. B. Orlin and D. Sharma. “A composite very large-scale neighborhood structure for the capacitated minimum spanning tree problem."

[15] Ravindra K. Ahuja James B. Orlin Dushyant Sharma ”Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem"

[16] Editors: Glover, Fred W., Kochenberger, Gary A.”Handbook of Metaheuristics"

[17] Mauricio G.C. Resende Celso C. Ribeiro ”Optimization by GRASP. Greedy Randomized Adaptive Search Procedures".