Máster Interuniversitario en Técnicas Estadísticas

Universidade Vigo

Cooperación en los problemas del viajante (TSP) y de rutas de vehículos (VRP): una panorámica.

AIDA CALVIÑO MARTÍNEZ

JUNIO 2011

Autorización de entrega

D.ª María Luisa Carpente Rodríguez y D.ª Silvia Lorenzo Freire

CERTIFICAN

Que el proyecto titulado "**Cooperación en los problemas del viajante (TSP) y de rutas de vehículos (VRP): una panorámica**"ha sido realizado por D.ª Aida Calviño Martínez, con D.N.I. 53.659.355-X, bajo la dirección de D.ª María Luisa Carpente Rodríguez y D.ª Silvia Lorenzo Freire. Esta memoria constituye la documentación que, con nuestra autorización, entrega dicho alumno como Proyecto Fin de Máster.

Firmado.

D.ª María Luisa Carpente Rodríguez D.ª Silvia Lorenzo Freire

Santiago de Compostela, a 28 de Junio de 2011

Índice general

Introducción

Los problemas del viajante y de rutas de vehículos son dos de los problemas más estudiados en Investigación Operativa. El primero de ellos se centra en estudiar problemas de la siguiente clase: un comercial debe visitar varios clientes y desea conocer cuál es el camino de mínima distancia que, partiendo de su lugar de trabajo, vaya a todas las ciudades y regrese. Por otro lado, los problemas de rutas de vehículos (más conocidos por sus siglas en inglés VRP) tratan de resolver problemas como el que sigue: una empresa debe repartir cierto producto entre sus clientes y desea encontrar la ruta (o rutas) de menor coste que, partiendo del almacén, visite cada cliente satisfaciendo su demanda y regrese al almacén. Como se puede deducir, ambos problemas están muy relacionados entre sí. De hecho, el VRP surgió como una extensión del TSP (problema del viajante) para el caso en el que la capacidad de los vehículos que realizan la ruta sea limitada y sea, por tanto, necesario realizar varias rutas.

Estos dos problemas de apariencia sencilla son famosos por su gran complejidad computacional. Como se verá mas adelante, ambos se encuadran dentro de la categoría NP-duro por lo que, a día de hoy, no se ha encontrado ningún algoritmo que logre resolverlos en un tiempo polinómico. No obstante, su importancia no se debe únicamente a la complejidad de su resolución, sino a la gran variedad de situaciones prácticas en las que pueden ser aplicados. La mayor parte de éstas se encuentran en el campo de la logística (reparto de mercancías, correo, rutas escolares), aunque también podemos encontrar aplicaciones de estos problemas en industria (producción de circuitos integrados) o en genética.

La primera parte de este trabajo se centra precisamente en esos problemas: en su historia, sus características, las distintas formas que existen de modelizarlos y en las formas de resolverlos. En el segundo capítulo del mismo se hace un repaso de los principales conceptos de la teoría de juegos cooperativos. Dicha teoría se encarga de estudiar el problema de repartir los costes o beneficios generados a partir de la cooperación entre agentes o empresas. Cada día, muchas empresas,

municipios o personas deciden coaligarse y trabajar juntos para conseguir un bien común. Gracias a dicha cooperación, se conseguirá reducir costes o aumentar beneficios, pues de lo contrario los agentes no estarán dispuestos a cooperar. El problema surge cuando dichos costes (o beneficios) deben repartirse entre los agentes implicados. La teoría cooperativa de juegos se dedica, entre otras cosas, a buscar repartos "justos"que hagan que los involucrados tengan incentivos para cooperar.

La tercera y última parte del trabajo trata de mezclar los dos temas anteriores, centrándose en los juegos cooperativos asociados al problema del viajante y al problema de rutas de vehículos. La idea es buscar formas de repartir los gastos generados de la cooperación entre varias empresas, ciudades o clientes que deben ser visitadas y que deciden unirse para así reducir los gastos asociados a esas "visitas". Para la coalición total, es decir, para todos los agentes que cooperan, debe plantearse un VRP o un TSP, según el contexto específico, para poder así determinar cuál sería el coste mínimo a repartir.

En este último campo, las publicaciones son relativamente escasas debido a la complejidad del asunto. Por ello, en la tercera parte se muestra un resumen de las publicaciones existentes y se sugieren nuevas líneas de investigación.

Capítulo 1

Problemas de rutas

1.1. Introducción

Entendemos por problema de rutas todo aquel problema de optimización que se plantea cuando existen unos clientes que demandan un servicio y se debe encontrar la mejor ruta para satisfacerles. La importancia de esta clase de problemas se debe al gran número de situaciones reales en las que se puede aplicar. Algunos ejemplos típicos son el reparto de correo, la recogida de basuras o el transporte escolar; aunque estos problemas no sólo pueden ser aplicados en logística y distribución, sino que también sirven para modelar otras situaciones como la producción de circuitos electrónicos integrados o la secuenciación de tareas. Debido al potencial ahorro de estas técnicas, las inversiones en investigación y software de las empresas relacionadas con el sector logístico han ido aumentando considerablemente en los últimos años.

Existen muchos tipos de problemas de rutas según las restricciones adicionales que se impongan (número de vehículos, ubicación de los clientes, ventanas de tiempo, capacidad de los vehículos, tipo de servicio demandado, etc). La principal diferencia existente entre estos problemas y los problemas de caminos es que en el primer caso hay un subconjunto de nodos y/o arcos que se deben visitar y, en el segundo, se busca una ruta que una el origen y el destino sin importar los nodos o arcos intermedios.

En el Cuadro 1.1 se encuentran los tipos más importantes de problemas de rutas junto con los nombres que históricamente han recibido.

<table>
<tr><th>Demanda</th><th>Restricciones de capacidad</th><th>Nombre habitual del problema</th><th>Otras restricciones</th></tr>
<tr><td rowspan="2">Nodos</td><td>NO</td><td>Viajante de Comercio
TSP</td><td rowspan="3">Recogida/distribución

Ventanas de tiempo

Otras</td></tr>
<tr><td>SÍ</td><td>Problema de rutas de vehículos
VRP</td></tr>
<tr><td rowspan="3">Arcos</td><td rowspan="2">NO</td><td>Una componente conexa
(Problema del Cartero Chino CPP)</td></tr>
<tr><td>Varias componentes conexas
(Problema del Cartero rural RPP)</td></tr>
<tr><td>SÍ</td><td>Problema de rutas con capacidades
CARP</td></tr>
</table>

Cuadro 1.1: Clasificación Problemas de Ruta

Como se puede observar en el cuadro, existen dos grandes tipos de problemas de rutas según los clientes se encuentren sobre los nodos o sobre los arcos. En el primero de los casos, la ruta óptima a determinar debe visitar todos los nodos, mientras que, en el segundo, se deben recorrer todos los arcos del grafo que define el problema. En otras palabras, en los problemas sobre los nodos se entiende que cada cliente está representado por un nodo mientras que en los problemas sobre los arcos se entiende que los arcos son calles que deben ser visitadas.

Los problemas de rutas sobre nodos tienen su origen en el siglo XIX cuando el irlandés W.R. Hamilton y el británico T. Kirkman inventaron el denominado "Icosian Game". Este juego consistía en encontrar una ruta entre los 20 puntos del juego usando sólo los caminos permitidos y regresando al nodo origen (una imagen del juego original comercializado años más tarde puede encontrarse en la Figura 1.1). Obviamente, este juego no se centraba en la búsqueda del camino óptimo, sino en la búsqueda de un camino que visitase todos los nodos una única vez (años más tarde, este tipo de caminos o ciclos recibirían el nombre de Hamiltonianos en honor a W.R. Hamilton).

La historia de los dos grandes problemas de rutas sobre nodos (el problema del viajante de comercio y los problemas de rutas de vehículos) está íntimamente relacionada. De hecho, históricamente se ha entendido el VRP como una generalización del TSP, como se verá más adelante. Estos dos problemas serán los temas centrales de este primer capítulo del trabajo, por lo que información más detallada sobre su historia, tipos y métodos de resolución puede encontrarse en sucesivas secciones.

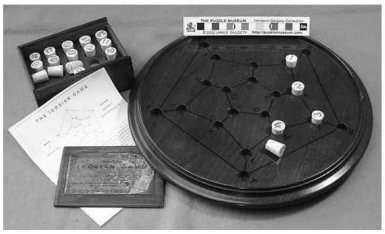


Figura 1.1: Hamilton's Icosian Game

Los problemas de rutas sobre arcos tienen su origen en el siglo XVIII cuando los habitantes de Königsberg, un pequeño pueblo de la actual Rusia, empezaron a debatir si existía alguna ruta que pasase una única vez por los 7 puentes que atravesaban el río Pregel y volviese al punto de origen (Figura 1.2). Este problema se propuso al matemático suizo Leonhard Euler, el cual demostró que no existía ninguna en un artículo del año 1736 (para más información, véase [18]).

Figura 1.2: Los puentes de Konigsberg

El problema de los puentes de Königsberg se refiere exclusivamente a la existencia de un camino y no a la búsqueda del óptimo, que es la filosofía de los problemas de rutas. En este sentido, el primer problema planteado en este campo es el Problema del Cartero Chino (CPP), definido en 1962 por Meigu Guan [28]. Este problema se resume en la búsqueda de un camino de distancia mínima que recorra todos los arcos del grafo al menos una vez. El método de resolución de este problema que propone Guan consiste en añadir arcos de coste mínimo al grafo original de manera que se logre convertir este grafo en un grafo euleriano.

Un grafo euleriano, cuyo nombre se debe al matemático Leonhard Euler, es un grafo en el que hay al menos un ciclo euleriano, es decir, un ciclo que contiene todas las aristas de un grafo una única vez. Para que un grafo sea euleriano todos sus vértices deben tener grado par[1], a excepción de un número par de nodos que puede tener grado impar, como demostró Hierholzer (1873) [30]. Una vez se ha conseguido un grafo euleriano, es relativamente fácil determinar un ciclo que atraviese cada arco una sola vez.

Otro de los grandes problemas de arcos es el conocido como Problema del Cartero Rural (RPP). Este problema fue introducido por primera vez en el año 1974 [46] y consiste en determinar el camino de mínima distancia que recorre sólo algunos de los arcos del grafo (el resto de los arcos puede ser o no recorrido). En 1976 se demostró que el RPP es un problema NP-duro (por lo que no se ha encontrado por el momento un algoritmo que resuelva cualquier ejemplo de dicho problema en tiempo polinomial), a no ser que el subgrafo formado por los arcos requeridos sea un grafo completamente conexo, en cuyo caso el RPP se reduce a un CPP, problema para el cual sí se han definido algoritmos que lo resuelven en un tiempo polinómico [38].

Por último, en el problema de rutas con capacidades (CARP), a cada arco (v_i, v_j) del grafo se le asocia una cantidad no negativa q_{ij}, que representa la demanda de cada uno de los clientes. Una flota m de vehículos con capacidad Q debe visitar todos los arcos repartiendo (o recogiendo) las cantidades correspondientes sin exceder nunca la cantidad Q. El CARP fue introducido por Golden y Wong (1981) [26] y una variante del mismo en el que las demandas han de ser estrictamente positivas fue investigado años antes, en 1973, por Christofides [6]. Cabe destacar que el problema estudiado por Golden y Wong puede verse como un RPP con capacidades (si la demanda de algún arco es 0 no será necesario atravesarlo), mientras que el definido por Christofides puede verse como un CPP con capacidades restringidas.

[1]Se dice que un nodo tiene grado par si el número de arcos que salen de él es par

1.2. Problema del viajante de comercio

1.2.1. Introducción

"Si un viajante parte de una ciudad y las distancias a otras ciudades son conocidas, ¿cuál es la ruta óptima que debe elegir para visitar todas las ciudades y volver a la ciudad de partida?" Esta podría ser una primera definición, por supuesto informal, del problema del viajante.

El problema del viajante (en inglés Traveling Salesman Problem TSP) es uno de los problemas más famosos y más estudiados en su área. A pesar de la aparente sencillez de su planteamiento, el TSP es uno de los más complejos de resolver y existen demostraciones que equiparan la complejidad de su solución a la de otros problemas aparentemente mucho más complejos que han retado a los matemáticos desde hace siglos, como veremos más adelante.

Historia

El origen del término "Traveling Salesman Problem"permanece aún desconocido, pues no existe documentación que apunte a ningún autor en concreto. No obstante, en una entrevista concedida por Merrill Flood (1984) [22], éste afirmó haberselo oído a A. W. Tucker quien, a su vez, lo había oído de Hassler Whitney en la Universidad de Princeton (esta teoría no ha sido confirmada por el propio Tucker al no recordarlo). La primera referencia a este término parece ser un artículo de 1949 de Julia Robinson [50], "On the Hamiltonian game (a traveling salesman problem)", pero parece claro por el título que no fue ella la que introdujo el término. En lo que sí parece estar de acuerdo la comunidad científica es en que este término fue acuñado entre 1931 y 1932 en la Universidad de Princeton.

A pesar de que los orígenes de este problema desde el punto de vista matemático se remontan a principios de la década de 1930, en 1832 se publicó un libro en Alemania titulado "El viajante de comercio: cómo debe ser y qué debe hacer para conseguir comisiones y triunfar en el negocio. Por un viajante de comercio veterano"[16], que puede ser considerado como la primera referencia bibliográfica al TSP. Pese a que se trata de un libro que se centra principalmente en otros aspectos de la profesión, en el último capítulo se define, de manera explícita, el Problema del Viajante de Comercio. Según esta guía, gracias a la experiencia y

a la correcta elección del orden en el que se visiten los clientes, se puede ahorrar tanto tiempo que los autores se vieron obligados a editar esta guía. Para ellos, lo importante es cubrir las máximas localizaciones posibles sin visitar el mismo lugar dos veces. Nótese la importancia de este libro: el TSP fue definido por un vendedor casi un siglo antes de que este tipo de problemas comenzara a estudiarse por la comunidad científica.

Además, el libro incluye cinco rutas que recorren regiones de Alemania y Suiza, una de las cuales es efectivamente la solución óptima al TSP. En cuanto a las otras cuatro, en las que sí se visita alguna ciudad más de una vez, se cree que teniendo en cuenta los medios de transporte existentes en la época podrían haber sido también rutas óptimas. Los medios de transporte utilizados a lo largo del tiempo han ido variando (a pie, a caballo, en tren, en automóvil, etc.), por lo que la planificación de rutas en cada caso debe tener en cuenta otros factores distintos a la distancia entre las ciudades.

Como ya se ha mencionado anteriormente, en la década de 1930 se comenzó a trabajar sobre el problema del viajante. En la universidad de Harvard, Merrill Flood, quien tuvo un papel muy importante en la labor de divulgación de este problema, se interesó por el TSP cuando empezó a trabajar en la búsqueda de una ruta óptima para un autobús escolar. Mientras tanto, en Viena, el matemático Karl Menger enunció lo que entonces se denominaba el problema del mensajero: buscar el camino más corto que una todos los puntos de un conjunto finito cuyas distancias entre sí son conocidas. Para Menger, este problema podía ser resuelto en un número finito de pruebas pero se desconocía la existencia de reglas que permitiesen reducir este número de pruebas por debajo del número de permutaciones existentes entre los puntos. Quizás sin saberlo, Menger [41] enunció una de las propiedades más importantes del TSP: es NP-duro, como se verá más adelante. Además, propuso lo que más tarde pasaría a llamarse *Algoritmo del vecino más próximo*: comenzar por el nodo origen e ir visitando cada vez el punto más cercano sin volver a un punto ya visitado; y observó que, generalmente, este algoritmo no da lugar al camino más corto .

En las décadas de los 50 y de los 60 el problema se hizo muy popular y comenzaron a estudiarse problemas para un número mayor de ciudades. Es destacable el artículo de Dantzig et al. del año 1954 "Solution of a large-scale traveling-salesman problem"[14]. En este artículo, que se considera uno de los principales eventos en la historia de la optimización combinatoria, se resuelve el problema

del viajante para 49 ciudades: una por cada estado de EEUU (Alaska y Hawai se convirtieron en estados en 1959) y Washington. Este libro resulta de gran importancia principalmente por 2 motivos:

- Supuso un gran avance en la historia del TSP al resolver un problema con un número tan alto de ciudades, teniendo en cuenta la falta de programas informáticos.

- Se utilizó un algoritmo (pese a que los autores se negaron a considerarlo un método de resolución) que sirvió de inspiración a muchos otros en las siguientes décadas. La idea fue aplicar las recientes técnicas de programación lineal al problema (Dantzig había desarrollado el algoritmo del símplex en 1947, lo que supuso un gran avance en las técnicas de optimización del momento) de una forma innovadora dando lugar al método de los cortes de plano, que más tarde evolucionaría hasta el algoritmo de ramificación y acotación (ambos procedimientos serán tratados con detenimiento posteriormente).

En línea con el artículo de Dantzig et al., muchos autores comenzaron a desarrollar otros algoritmos que fuesen aplicables a problemas con un número cada vez más grande de ciudades. Gracias a esto y al gran desarrollo de la informática en las últimas décadas, ha habido grandes avances en la resolución de los TSP. En el Cuadro 1.2 pueden observarse los problemas más destacables que fueron resueltos desde 1954 hasta 1990.

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

Cuadro 1.2: Hitos en la resolución del TSP

En 1990 se comenzó a desarrollar un programa informático llamado Concorde consistente en más de 130000 líneas de código en C que ha permitido resolver problemas de hasta 85900 ciudades en el año 2006. Matemáticos y otros profesionales trabajan cada día en la mejora de este programa compuesto por las mejores técnicas disponibles hasta el momento. En la Figura 1.3 se puede observar la progresión en el número de ciudades hasta el año 2006.

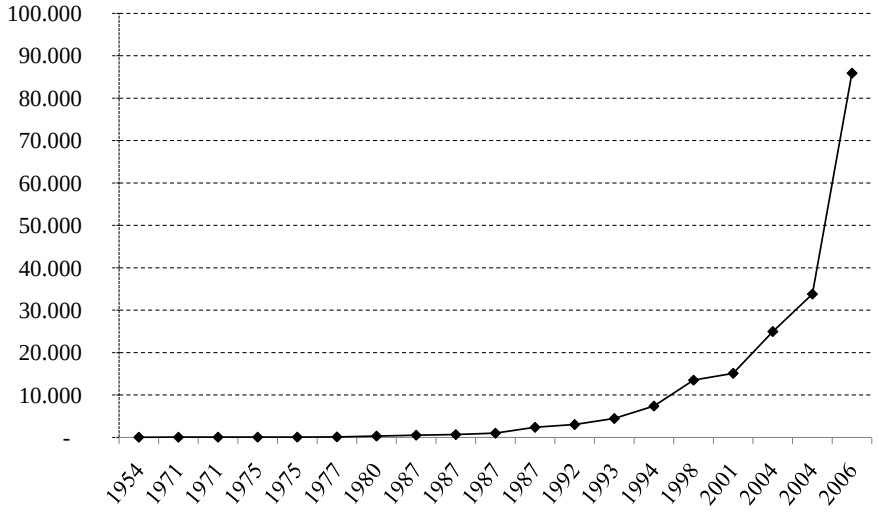


Figura 1.3: Progresos en la historia del TSP

Aplicaciones

La mayor parte de las mejoras en TSP durante los primeros años estaban motivadas por aplicaciones directas del mismo. Entre otros, Flood [21] trabajó sobre rutas de autobuses escolares y Morton y Land [43] aplicaron el TSP a la planificación de rutas de una empresa de lavandería. Hasta el día de hoy, el TSP se ha aplicado sobre una gran variedad de problemas que van desde rutas de vendedores hasta la genética. A continuación, se comentan brevemente algunas de las aplicaciones más importantes del problema del viajante:

- **Logística.-** Las aplicaciones más directas y más abundantes del TSP se centran en el campo de la logística. El flujo de personas, mercancías y vehículos en torno a una serie de ciudades o clientes se adapta perfectamente a la filosofía del TSP, como ya demostraron los primeros estudiosos del problema. Entre las múltiples aplicaciones logísticas del problema del viajante, destacamos:

- Vendedores y turistas.- Aunque los viajes que se realizan por placer o por negocio rara vez se plantean como un TSP, la mayor parte de los vendedores y turistas utilizan algún planificador de rutas para determinar cuál es el mejor camino para visitar los puntos que desean y volver al punto de origen (nótese que los turistas desean visitar los monumentos o lugares emblemáticos y después regresar al hotel). Estos planificadores generalmente incluyen algún algoritmo de resolución del TSP.
- Rutas escolares.- Las rutas escolares representan una de las primeras aplicaciones del TSP (Merrill Flood se interesó por el problema del viajante cuando estaba intentando determinar una ruta escolar óptima). Actualmente, muchas empresas dedicadas al transporte de personas adquieren software de resolución de TSP que les permite reducir gastos de una manera significativa.
- Reparto de correo.- Aunque generalmente el reparto de correo se ajusta mejor a un problema de rutas sobre arcos, como ya se vio anteriormente, en ocasiones el reparto de correo puede modelizarse como un TSP. Se trata de los casos en los que las casas están muy alejadas unas de otras o cuando sólo se debe visitar algunas de ellas (sería el caso de las empresas de paquetería). Este esquema es aplicable al reparto de cualquier otro tipo de mercancía.

- **Industria.-** Las aplicaciones en industria no son tan numerosas como en logística, pero la aplicación del problema en este ámbito también ha dado lugar a una significativa reducción de los costes. Entre las aplicaciones a la industria encontramos:

  - Secuenciación de tareas.- Supongamos que una máquina debe realizar una serie de tareas en el mínimo tiempo posible y sin importar el orden de las mismas. Supongamos que se tarda un tiempo t_{ij} en poner a punto la máquina para realizar la tarea j si la última tarea que realizó fue la i. En ese caso, podemos aplicar un TSP suponiendo que cada tarea es uno de los nodos a visitar, han de realizarse todas las tareas para producir el producto y que la distancia entre ellos es t_{ij}. El nodo origen y destino serían el estado de la máquina cuando empieza o termina el producto. Dado que el tiempo que se emplea en realizar cada tarea no depende del orden, no será necesario incluir estos tiempos en el modelo,

pues la suma de todos es constante independientemente del orden. Una variación de este modelo fue estudiada por Gilmore y Gomory (1964) [25].

- Producción de circuitos electrónicos.- La utilización del TSP para la producción de circuitos electrónicos se centra en dos aspectos: el orden óptimo de taladrar las placas y los caminos óptimos necesarios para conectar los chips entre sí.
  - Problemas de perforado.- Los circuitos integrados se encuentran en muchos dispositivos electrónicos, por lo que la producción de las placas sobre las que se montan dichos circuitos es un problema cotidiano. Dichas placas han de ser perforadas un número relativamente grande de ocasiones. Los orificios resultantes sirven para introducir los chips correspondientes. Generalmente, son taladros automáticos los que realizan, uno tras otro, las perforaciones correspondientes. Si estas máquinas no son programadas correctamente, el tiempo que se tarda en recorrer la placa de un orificio a otro puede aumentar significativamente, dando lugar a pérdidas económicas (si se tarda mucho en producir cada placa, produciremos menos placas en el mismo tiempo). Por tanto, la aplicación del TSP en este campo consiste en, tomando como ciudades cada una de las posiciones donde debe realizarse una perforación y las distancias entre ellas como el tiempo que necesita la máquina en trasladarse de una a otra, minimizar el tiempo que pierde la taladradora en moverse de una posición a otra. La ciudad de origen y destino será un punto adicional que represente el lugar donde permanece la perforadora mientras las placas se cambian. Nótese que si el tiempo que se tarda en perforar es muy superior al tiempo de desplazamiento, no tendrá sentido plantear un TSP, pues la disminución del tiempo será casi imperceptible. Estas aplicaciones llevan años siendo estudiadas (existe un artículo de Lin y Kernighan (1973) [39] donde se trata este tema) y ya han sido utilizadas por grandes empresas, como son Siemens e IBM, dando lugar a mejoras de aproximadamente el 10 % del rendimiento total de las líneas de producción.
  - Conexión de chips.- Este tipo de ejemplos se da frecuentemente en el diseño de ordenadores y de otros dispositivos digitales. Den-

tro de muchos de estos dispositivos existen placas que cuentan con chips que deben ser conectados entre sí por cables. Para evitar problemas de interferencias y debido al pequeño tamaño de los chips, no se pueden poner más de dos cables en un único pin. La idea es, por tanto, minimizar la cantidad de cable necesaria para unir todos los puntos. Claramente este modelo puede ser modelizado como un TSP tomando los pins como las ciudades y la distancia entre ellas, la cantidad de cable necesario para unirlas. Obsérvese que de no existir la restricción de sólo dos cables por chip, este problema debería ser modelizado como la búsqueda del árbol de mínima expansión, problema para el cual existen algoritmos eficientes.

- **Creación de cluster de datos.-** La organización de datos en grupos (clusters) de elementos con propiedades similares es un problema básico en análisis de datos. El problema del viajante ha sido aplicado frecuentemente en problemas de este tipo cuando existe una buena medida de la similitud $s(a, b)$ entre cada pareja de datos (a, b). La idea es que, usando $s(a, b)$ como distancias, un camino Hamiltoniano de coste máximo situará las observaciones más parecidas cerca unas de otras y se podrá, por tanto, utilizar intervalos del camino como clusters. Cabe destacar que se busca un camino de coste máximo, puesto que la medida de similitud toma un valor mayor cuanto más próximas estén las observaciones entre sí. Podría pensarse que la búsqueda de un camino de coste máximo implica la implementación de un nuevo algoritmo de resolución del TSP, pero si se multiplican las distancias por el valor (-1) y se aplican las técnicas correspondientes se obtendrá un camino de coste mínimo, de coste negativo, que será equivalente al camino de coste máximo. Una vez obtenido el camino, la selección de los clusters puede hacerse a mano, buscando los puntos de corte naturales según la naturaleza de los datos, o puede hacerse de manera automática. Existen muchas formas de buscar estos clusters de manera automática, pero una manera muy elegante de hacerlo, que propusieron S. Climer y W. Zhang [10], consiste en añadir k ciudades cuya distancia al resto de ciudades sea 0. Así, estas ciudades adicionales servirán para identificar los k clusters, puesto que un camino óptimo usará las conexiones de coste cero para reemplazar las grandes distancias existentes entre los clusters. Este método permite variar el valor de k, observando así el impacto de distintos números

de grupos. Climer y Zhang usaron este método para crear clusters de genes.

Estas no son las únicas áreas en las que se ha aplicado el TSP; existen una multitud de problemas que han podido ser resueltos gracias a su aplicación. De hecho, en los últimos años se han multiplicado las aplicaciones del TSP en problemas genéticos.

¿Es el TSP NP-duro?

Habitualmente, cuando se habla del problema del viajante se hace referencia a la dificultad de su resolución pero lo cierto es que, a pesar de no haber encontrado todavía un algoritmo "bueno", no podemos afirmar que este algoritmo no exista.

En ese sentido, podemos afirmar que el TSP tiene solución, pues siempre pueden evaluarse todas las posibles soluciones y escoger la de menor coste. El problema es que el número de posibles soluciones aumenta de manera significativa cuando aumenta el tamaño del problema. El número de ciclos posibles puede calcularse de manera sencilla: el origen viene determinado por lo que restarán $(n-1)$ puntos para empezar, a continuación deberemos elegir cualquiera de los $(n-2)$ restantes y así sucesivamente. De esta forma, multiplicando todas estas cantidades obtenemos el número total de rutas o soluciones posibles:

$$(n-1)! = (n-1) \cdot (n-2) \cdot (n-3) \cdots 3 \cdot 2 \cdot 1$$

Por lo tanto, el método directo implica evaluar $(n-1)!$ soluciones posibles. En el caso particular de 10 ciudades esto significaría evaluar 362880, es decir, para un problema no excesivamente grande el número de pruebas aumenta considerablemente. Así, aunque el problema puede resolverse en un número finito de pasos, esto no es suficiente y deberán buscarse otros algoritmos que reduzcan este número de pruebas.

De acuerdo con lo anterior, para evaluar y comparar algoritmos podría utilizarse el número de iteraciones (o pruebas) que requieren para alcanzar una solución. En el caso del método directo, el número de pasos es siempre $(n-1)!$ pero existen otros algoritmos para los que, debido a su naturaleza, es muy difícil, o incluso imposible, calcular este número de pasos. Debemos, entonces, buscar otro método de comparación. El método que generalmente se emplea consiste en comparar el tiempo que los métodos emplean en encontrar dicha solución. El tiempo que se asocia a cada algoritmo debe ser una función de n (el tamaño del

problema), que represente una cota del tiempo máximo que necesite para alcanzar la solución. Como ocurre con todo método, éste también tiene sus desventajas: estamos juzgando un algoritmo según el tiempo que necesita para resolver el problema que peor resuelve, pero puede ocurrir que generalmente necesite un tiempo muy inferior a esta cota.

En el año 1962, M. Held y R. Karp [29] descubrieron un nuevo algoritmo que, basado en programación dinámica, requiere de un tiempo proporcional a $n^2 2^n$ (es destacable, además, que este algoritmo tiene el mejor tiempo computacional de entre todos los algoritmos capaces de resolver cualquier TSP descritos hasta el momento). Utilizando la técnica de comparación definida anteriormente, podemos concluir que este nuevo método es significativamente mejor que el método directo (para el caso particular de $n = 10$ el número de soluciones se reduce de 362880 a 102400). Mientras que ningún ordenador del mundo era capaz de resolver un problema del viajante de 50 ciudades con el método directo, con el algoritmo definido por Karp esto se hizo posible. No obstante, si el número de ciudades se duplica, la tecnología actual tampoco permite calcular el óptimo a través de ese algoritmo. Entonces, ¿qué es lo que hace a un algoritmo "bueno"?

El matemático J. Edmonds [15] propuso una definición formal de lo que es un "buen"algoritmo de resolución. Para él, un algoritmo puede clasificarse como bueno o eficiente si el tiempo necesario para que alcanze el máximo, entendido según la definición anterior, es Kn^c, siendo K y c dos números constantes. Generalmente, la constante K se elimina y se habla de algoritmos $O(n^c)$. Según las definiciones anteriores, el método propuesto por Held y Karp es un algoritmo $O(n^2 2^n)$ y, por tanto, no es un buen algoritmo. Dado que el término "bueno"tiene muchos significados, muchos investigadores prefieren denominar estos algoritmos como algoritmos de tiempo polinómico. De esta forma, los problemas pueden clasificarse como fáciles o difíciles según exista o no un método eficiente para resolverlos.

Una cuestión muy importante, que fue planteada por Edmonds en los años sesenta, es si existe o no un "buen"algoritmo para el problema del viajante. Hasta el día de hoy esta cuestión no ha sido resuelta. De hecho, el Instituto Clay de Matemáticas [9] ha ofrecido una recompensa de un millón de dólares al que descubra un algoritmo eficiente para el TSP o demuestre la no existencia del mismo.

Por esta razón, el problema del viajante ha alcanzado una importante posición

en la teoría de la complejidad. Según esta teoría, los problemas para los que existe un buen algoritmo de resolución son clasificados como P, de tiempo polinómico. Por el contrario, los problemas para los que aún no se ha encontrado un algoritmo eficiente son clasificados como NP, de tiempo polinómico no determinista. Un resultado muy importante en este campo procede de un artículo de S. Cook (1971)[12]. Cook demostró que muchos de los problemas denominados como "difíciles"son computacionalmente equivalentes, en el sentido de que un algoritmo polinómico para uno de ellos puede utilizarse para resolver los demás en un tiempo también polinómico. Los problemas que se engloban en este grupo reciben el nombre de NP-duro. El método utilizado por Cook ha servido de inspiración para muchos otros autores, que han catalogado así cientos de problemas dentro de esta categoría.

Así, una de las principales cuestiones de la teoría de la complejidad es si existe o no un algoritmo de tiempo polinómico para cualquier problema NP-duro. De existir, entonces todos los problemas dentro de la categoría NP podrían resolverse en un tiempo polinómico y se concluiría que las clases P y NP son iguales. Este es uno de los problemas más destacados de matemáticas y es en este sentido que el Instituto Clay ha ofrecido un millón de dólares.

Cada año se presentan múltiples propuestas intentando demostrar que $P = NP$, normalmente proporcionando algún algoritmo eficiente para el TSP, lo que resulta lógico, pues el problema del viajante es probablemente el más estudiado de todos lo problemas clasificados como NP-duro. Hasta el momento, muchas de estas propuestas siguen sin estudiarse, pues muchos investigadores se inclinan más por la teoría de que $P \neq NP$. Esta cuestión ha hecho de la teoría de la complejidad un campo mucho más activo.

A pesar de no haberse encontrado ningún algoritmo eficiente para el problema general del viajante, sí se han encontrado algoritmos para algunos casos particulares del mismo, incluso en algunos casos se ha logrado demostrar la no existencia de dichos algoritmos. Este es uno de los aspectos más sorprendentes de la teoría de la complejidad: problemas de aparente mayor dificultad son resueltos frente a otros de apariencia sencilla que no lo son. En ese sentido, encontramos dos casos especiales de TSP:

- TSP euclídeo.- Se trata de un problema del viajante ficticio, en el sentido de que las ciudades se sitúan aleatoriamente en un cuadrado de tamaño prefijado según una distribución uniforme. Las distancias entre los puntos

se calculan según la métrica euclídea. La importancia de este problema es que se ha podido demostrar que pertenece a la categoría NP-duro.

- TSP con matriz de distancia triangular.- Se ha demostrado que los problemas del viajante con matriz de distancias triangulares ($c_{ij} = 0$ si $i > j$) pueden resolverse en un tiempo polinómico, a pesar de su gran parecido con el TSP general.

El primer caso particular de TSP cuya estructura permitió aplicar un algoritmo de tipo polinómico fue un problema de secuenciación de tareas [25]. En los últimos años ha sido destacable la literatura rusa que se ha concentrado en la identificación de casos "sencillos"del problema del viajante.

1.2.2. Modelización y Formulación del TSP

Existen múltiples formulaciones distintas del TSP. En este apartado se van a revisar algunas de las más utilizadas.

El problema del TSP puede ser descrito según la teoría de grafos de la siguiente manera: Sea $G = (V, A)$ un grafo completo, donde $V = 1, ..., n$ es el conjunto de vértices y A es el conjunto de arcos. Los vértices $i = 2, ..., n$ se corresponden con los clientes a visitar y el vértice 1 es la ciudad de origen y destino. A cada arco (i, j) se le asocia un valor no negativo c_{ij}, que representa el coste de viajar del vértice i al j. El uso de los arcos (i, i) no está permitido, por lo que se impone $c_{ii} = \infty$ para todo $i \in V$. Si G es un grafo dirigido, la matriz de costes c es asimétrica mientras que, si $c_{ij} = c_{ji}$ para todo $(i, j) \in A$, la matriz de costes será simétrica y el problema recibirá el nombre de TSP simétrico (STSP). En ese caso, el conjunto A se sustituye por un conjunto E de arcos no dirigidos (i,j) tales que $i < j$.

El objetivo del problema del viajante es encontrar una ruta que, comenzando y terminando en una ciudad, en este caso denotada por la ciudad 1, pase una sola vez por cada una de las ciudades y minimice la distancia recorrida. Si definimos las variables dicotómicas de decisión x_{ij} para todo $(i, j) \in A$, de forma que tomen el valor 1 si el arco (i, j) forma parte de la solución y 0 en otro caso; tenemos que el problema de programación lineal asociado al problema del viajante consiste en minimizar la siguiente función objetivo:

$$\sum_{ij} c_{ij} x_{ij}$$

sujeto a las siguientes restricciones:

$$\sum_{j \in \delta^-(i)} x_{ji} = 1 \quad \forall i \in V,$$

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

donde

$$\delta^-(i) = \{a = (j, i) \in A\}; \quad \delta^+(i) = \{a = (i, j) \in A\}.$$

La primera restricción se refiere a que sólo un arco puede entrar en cada vértice, mientras que la segunda se refiere a que sólo un arco puede salir de

cada nodo. Estas restricciones son necesarias pero no suficientes, pues pueden dar lugar a subcircuitos, como se puede observar en la Figura 1.4. Obsérvese que $x_{12} = x_{23} = x_{31} = x_{54} = x_{46} = x_{65} = 1$, por lo que no se viola ninguna de las restricciones.

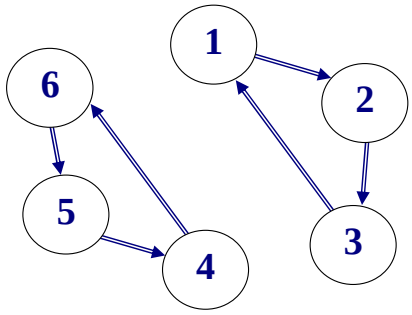


Figura 1.4: Subcircuitos en TSP

Por lo tanto, debemos incluir alguna restricción más de ruptura de subcircuito. En la Figura 1.4 se puede observar que en el subconjunto $\{1, 2, 3\}$ hay 3 arcos que unen los nodos entre sí. Si limitásemos este número de arcos a 2, evitaríamos que se pudieran dar situaciones como esta. Para poder modelizar estas últimas restricciones, es necesario nuevos conjuntos:

$$\forall W \subset V, \qquad A(W) = \{a = (i,j) \in A : i, j \in W\};$$

$$\delta^-(W) = \{a = (i,j) \in A : i \notin W, j \in W\}$$

$$\delta^+(W) = \{a = (i,j) \in A : i \in W, j \notin W\}.$$

Así, las restricciones de ruptura de subcircuito pueden escribirse de la siguiente manera:

$$\sum_{(i,j)\in A(W)} x_{ij} \leq |W| - 1 \quad \forall W \subset V.$$

Que es equivalente a:

$$\sum_{i\in W, j\notin W} x_{ij} \geq 1 \quad \forall W \subset V.$$

Esta restricción indica que para todo subconjunto de los nodos debe haber al menos un arco que "salga"del subconjunto.

Otra forma de evitar la formación de subcircuitos es a través de nuevas variables de decisión. De esta forma, definimos las variables $u_i, \forall i \in V$, que representan el lugar de la secuencia en el que se visita el nodo i. Para el nodo 1, el origen,

prefijamos el valor de u_0 en 1, pues es el primer nodo que se debe visitar. Para el resto de vértices, estas variables toman un valor entre 2 y n. Para construir el problema de programación lineal debemos añadir a las dos primeras restricciones la siguiente:

$$u_i - u_j \leq (n-1)(1-x_{ij}) - 1 \quad \forall (i,j) \in A, j \geq 1.$$

La interpretación de esta restricción es la siguiente: si el viajante va de i directamente a j, entonces x_{ij} valdrá 1 y $u_i - u_j \leq -1$. Dado que visitamos antes el nodo i que el j, u_i tomará un valor menor que u_j y, por tanto, su diferencia valdrá (-1), cumpliéndose así la inecuación. Si el viajante no va de i a j, x_{ij} tomará el valor 0, con lo que $u_i - u_j \leq n-2$. En este caso, no se tiene información adicional acerca de u_i y u_j, por lo que nos centramos en el extremo: si i se visita antes que j, la máxima diferencia entre u_i y u_j será $2-n$, mientras que si j se visita antes que i, $u_i - u_j$ tomará un valor mínimo de $n-2$. En todos los casos, por tanto, se cumple que $u_i - u_j \leq n-2$. Obviamente, de existir subcircuitos estas variables no podrían tomar valores que cumplieran esta restricción, pues no se puede establecer qué vértices se visitan antes y cuáles se visitan después.

En la primera de las formulaciones se requieren $2^n + 2n - 2$ restricciones ($2^n - 2$ por las de ruptura de subcircuito, n por la restricción de "entrada"y n por la restricción de "salida"de cada nodo) y $n(n-1)$ variables dicotómicas. Por el contrario, para la segunda formulación se necesitan $n^2 - n + 2$ restricciones ($(n-1)(n-2)$ por las restricciones asociadas a las variables u_i, y 2n por los otros dos tipos de restricciones), $n(n-1)$ variables dicotómicas y $(n-1)$ variables continuas. Si comparamos ambas formulaciones, en el primero de los casos hay más restricciones pero menos variables que en el segundo caso. En cada caso particular y, dependiendo de la forma de resolución de este problema de programación lineal entera, merecerá la pena utilizar una u otra formulación. No obstante, se puede demostrar que la formulación con las restricciones de ruptura de subcircuito domina a la formulación con las u_i.

Por último, veamos la primera formulación para el caso del TSP simétrico. De nuevo, las variables de decisión x_{ij} toman el valor 1 si el arco que une i y j pertenece al circuito solución y 0 en otro caso. La formulación es la que sigue:

$$Min \quad \sum_{ij} c_{ij} x_{ij}$$

$$s.a. \quad \sum_{j \in \delta(i)} x_{ij} = 2 \quad \forall i \in V$$

$$\sum_{(i,j) \in E(W)} x_{ij} \leq |W| - 1 \qquad \forall W \subset V \ , n/2 \leq |W| \geq 3$$

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

siendo $\delta(i) = \{e \in E : e = (i,j) \text{ ó } e = (j,i)\}$; $E(W) = \{(i,j) \in E : i,j \in W\}$ y $\delta(S) = \{e = (i,j) \in E : (i \in S, j \notin S) \text{ ó } (i \notin S, j \in S)\}$.

Como ocurre con el caso asimétrico, la segunda restricción puede sustituirse por:

$$\sum_{(i,j) \in \delta(W)} x_{ij} \geq 2.$$

1.2.3. Variantes del TSP

Existen multitud de variantes al problema de viajante general, tal cual se ha explicado anteriormente. Seguidamente se enumeran algunas de ellas:

- MAX-TSP.- Consiste en encontrar un circuito hamiltoniano de coste máximo.

- TSP con cuello de botella.- Consiste en encontrar un circuito hamiltoniano tal que minimice el mayor coste de entre todas las aristas del mismo, en vez de minimizar el coste total.

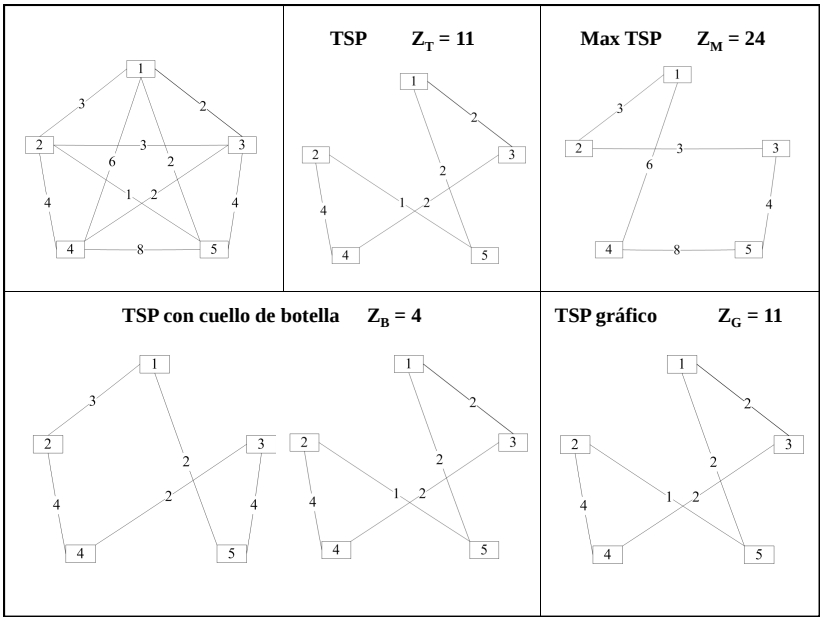
- TSP gráfico.- Consiste en encontrar un circuito de coste mínimo tal que se visiten las ciudades al menos una vez.

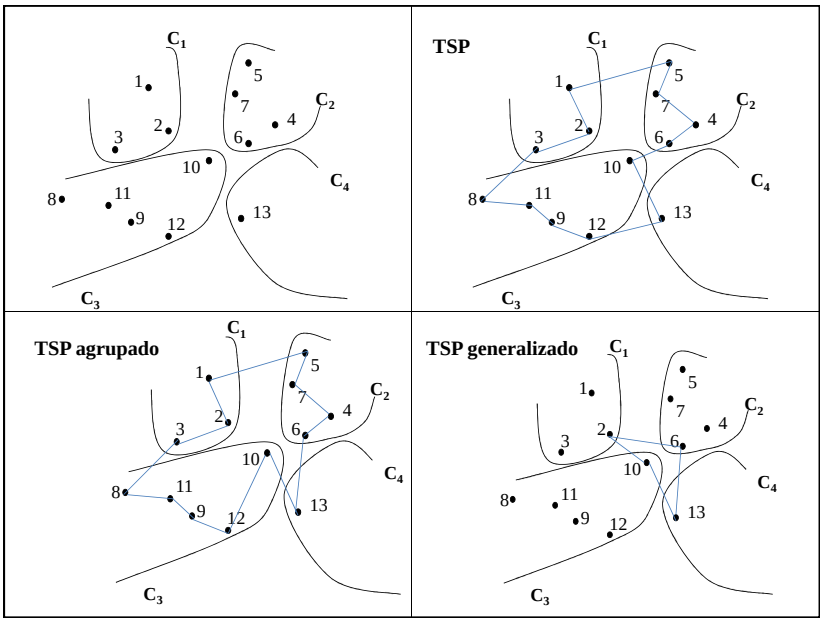
- TSP agrupado.- Los nodos o ciudades están divididos en "clusters"o grupos, de manera que lo que se busca es un circuito hamiltoniano de coste mínimo en el que se visiten los nodos de cada grupo de manera consecutiva.

- TSP generalizado.- Los nodos o ciudades también están divididos en grupos, pero lo que se busca es un circuito de coste mínimo que visite exactamente un nodo de cada grupo.

- TSP con múltiples viajantes.- Existen un número m de viajantes, cada uno de los cuales debe visitar algunas de las ciudades. El problema se transforma, por tanto, en la búsqueda de una partición de los nodos a visitar $X_1, ..., X_m$ y de m ciclos, uno para cada X_i, de manera que la suma de las distancias recorridas por los m viajantes sea mínima. Esta variante del TSP puede ser vista también como una simplificación de los problemas de rutas de vehículos, que serán estudiados en la siguiente sección.

En la Figura 1.5 puede observarse un ejemplo de las variantes del TSP explicadas anteriormente.





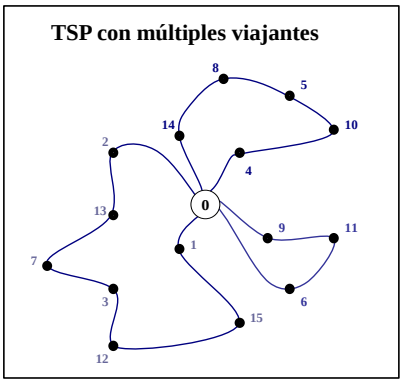


Figura 1.5: Variantes del TSP

1.2.4. Algunos métodos de resolución

La teoría de la complejidad ha servido para confirmar la gran dificultad de algunos problemas como el TSP. Para esos casos, la búsqueda de soluciones se limita a tres estrategias: intentar buscar casos particulares del problema que sí permitan la utilización de algoritmos eficientes (como es el caso del TSP con matriz de distancias triangular), continuar con la búsqueda de buenos algoritmos (arriesgándonos así a invertir demasiado tiempo) o concentrarse en la búsqueda de soluciones rápidas (y aceptar la posibilidad de que no sean óptimas).

La primera de las posibilidades ha dado lugar a importantes resultados, pero no nos centraremos en ella en este trabajo; la segunda (la búsqueda de un algoritmo eficiente) es la estrategia a la que más tiempo se ha dedicado históricamente y algoritmos como el de ramificación y acotación, de los planos de corte o programación dinámica se encuadran en ésta; por último, la tercera, una de las más escogidas en casos prácticos, comenzó a utilizarse a finales de la década de los cincuenta y a principios de la de los sesenta y los métodos o algoritmos que se definen reciben el nombre de heurísticas.

En esta sección se van a explicar, en primer lugar, algunas de las técnicas para obtención de solución exacta mencionadas anteriormente (branch and bound, programación dinámica, etc.) y, a continuación, se explicarán detalladamente algunas heurísticas aplicadas a este problema.

Método de los planos de corte

La primera de las técnicas empleadas para resolver el problema del viajante fue definida en el artículo de Dantzig et al. [14] en el año 1954 y recibió el nombre de "cutting plane method" (el método de los planos de corte). Como ya se ha mencionado anteriormente, la base de esta técnica son las técnicas de programación lineal. Este método puede ser aplicado a multitud de problemas. En particular, se desea minimizar la función $c^t x$ sujeto a $x \in S$, donde S es un conjunto finito de vectores m-dimensionales. En el caso particular del TSP, este conjunto S es el conjunto total de circuitos que unen las ciudades. Dantzig et al. definían cada ruta como un vector de dimensión $n(n-1)/2$ en el que cada componente (x_e) del mismo toma el valor 1 si el arco e correspondiente pertenece al circuito, y 0 en caso contrario.

En lugar de resolver el problema anterior, la idea de este método es resolver un problema más sencillo y, poco a poco, ir añadiendo restricciones hasta conseguir

una solución que pertenezca a S. En el caso del STSP, el problema que se resuelve es el siguiente:

$$\text{Minimizar } c^t x \text{ sujeto a:}$$

$$0 \leq x_e \leq 1, \text{ para todo arco e}$$

$$\sum(x_e : \text{v es un extremo de e}) = 2, \text{ para todo vértice v.}$$

Como se puede observar, no se han incluido las restricciones de ruptura de subcircuito y se permite, además, que las variables de decisión tomen un valor no entero. Por lo tanto, las soluciones a este problema podrán no ser circuitos pero lo que sí es cierto es que cualquier solución factible del TSP es también solución factible de este problema. Además, la resolución de este último nos proporciona una cota mínima, en el sentido de que ningún circuito podrá tener un coste inferior a $c^t x^*$. Esta cota nos servirá, además, para evaluar la calidad de cualquier circuito propuesto.

No se incluyen las restricciones de ruptura de subcircuito, puesto que esto implicaría trabajar con un problema de programación lineal con demasiadas restricciones, aumentando de manera significativa la complejidad computacional del mismo. En su lugar, lo que propusieron Dantzig et al., es añadir dichas restricciones al problema a medida que fuesen haciendo falta, es decir, se resuelve el problema anterior y se buscan subcircuitos. Entonces, se plantea un nuevo problema con las restricciones anteriores más la restricción de rotura de subcircuito asociada al conjunto de vértices donde se encuentra el subcircuito. Dicha restricción se plantea de la misma forma que se vio en el apartado anterior.

De forma general, el método de los planos de corte plantea un problema de programación lineal simplificado (de la forma minimizar $c^t x$ sujeto a $Ax \leq b$), de tal manera que todas las soluciones $x \in S$ cumplan dicho conjunto de restricciones. Recordemos que, según el método del símplex, la solución a cualquier problema de programación lineal se encuentra en el conjunto convexo definido por las restricciones. Por lo tanto, si la solución hallada no se encuentra en S (no pertenece al conjunto convexo definido por sus restricciones), será posible definir un hiperplano que separe dicha solución x^* del conjunto S. El citado hiperplano debe ser definido por una serie de restricciones o condiciones que cumplan los puntos de S y no cumpla x^*. En este punto, se plantea el problema anterior junto con la nueva restricción y se calcula una nueva solución x^*. Este procedimiento se repite hasta conseguir que la solución x^* esté efectivamente dentro del conjunto

convexo definido por S. En el caso del TSP, dichos hiperplanos se corresponden con las restricciones de ruptura de subcircuito.

Los hiperplanos definidos anteriormente reciben el nombre de planos de corte (cutting planes) y es por ello que el método recibe el nombre de "Cutting-plane method".

Método de ramificación y acotación

El método de ramificación y acotación (en inglés, Branch & Bound) empezó a desarrollarse en los años posteriores al artículo de Dantzig et al. [14]. De hecho, en dicho artículo aparecen algunas pistas referentes a este método, que es clasificado como una extensión del método de los planos de corte. El término "branch-and-bound"fue acuñado por Little et al. [40] y definido por ellos en el sentido más general. Como ocurre con el método anterior, el método de ramificación y acotación sirve para una multitud de problemas y no sólo para el TSP.

El algoritmo comienza intentando resolver el problema: "Minimizar $c^t x$ sujeto a $x \in S$"(Nótese que la notación es la empleada en el método anterior). Ante la imposibilidad de resolver dicho problema, el problema se subdivide en dos problemas, de tal manera que el primero de ellos sea el problema anterior más una restricción adicional, que será $\alpha^T x \leq \beta'$, y el segundo será de nuevo el problema original más la siguiente restricción: $\alpha^T x \geq \beta'$. Ante la imposibilidad de resolver alguno de estos problemas, o ante la posibilidad de que la solución obtenida no sea válida, los subproblemas pueden ser, a su vez, divididos en dos subproblemas para otros valores de α y β'. Estos subproblemas reciben el nombre de ramificaciones, pues generalmente este método se representa por un árbol de decisión, siendo la raíz el problema original y las ramas los subproblemas. El proceso continúa de esta forma hasta obtener alguna solución satisfactoria o hasta que alguna rama sea "podada", lo que ocurre cuando el subproblema correspondiente da lugar a una solución tal que la función objetivo tome un valor superior a algún elemento anterior del árbol.

En el caso particular del TSP, este método se ha aplicado desde diferentes puntos de vista. A continuación, se explica la propuesta por Eastman en su tesis doctoral [17]. En dicha tesis, se propone comenzar el árbol con el problema:

$$\text{Minimizar } c^t x \text{ sujeto a:}$$

$$0 \leq x_e \leq 1, \text{ para todo arco e}$$

$$\sum(x_e : \text{v es un extremo de e}) = 2, \text{ para todo vértice v}$$

Lógicamente, la solución a este problema puede contener subcircuitos. Para esos casos, se elige un subcircuito de k arcos y se definen k ramas, cada una de las cuales se corresponde con el problema anterior más la restricción de que cada una de las variables asociadas a los k arcos (x_e) tome el valor 0. De esta forma, se consigue eliminar los subcircuitos. Se continúa en cada rama con este esquema hasta obtener soluciones que sean efectivamente circuitos. Si en alguna de las ramas se obtiene una solución, aunque no se trate de un subcircuito, con un coste superior al de alguna solución válida (circuito completo) encontrada por alguna otra rama, dicha rama ha de ser "podada", pues cualquier circuito derivado de esa solución tendrá un coste superior. Nótese, al igual que ocurre con el método de los planos de corte, que el valor de la función objetivo de una rama será una una cota mínima de las soluciones de los subproblemas que se deriven de él.

Programación dinámica

Una tercera clase de algoritmos de resolución para el TSP surgió a principios de los años sesenta, recurriendo a la teoría de programación dinámica definida por R. Bellman [2]. La idea de este método es que en un circuito óptimo, tras haber recorrido una serie de ciudades, el camino que atraviese las restantes ciudades debe ser también óptimo. Este hecho permite construir el circuito paso a paso: a partir de una lista de los caminos de mínimo coste entre las ciudades de todos los subconjuntos de tamaño k, especificando en cada caso el origen y el destino, se puede crear una lista de todos esos caminos para conjuntos de k ciudades. Esta estrategia fue estudiada, entre otros, por Held y Karp [29].

En el artículo de Held y Karp se explica este método y se informa de una implementación informática capaz de resolver problemas de hasta 13 ciudades. Esta modesta cifra se debe al rápido incremento en la cantidad de datos que deben ser procesados y guardados para poder crear dicha lista de k caminos. Además, Held y Karp demostraron que este algoritmo era capaz de resolver problemas del viajante de cualquier tamaño n en un tiempo que era, como máximo, proporcional a $n^2 2^n$ (este algoritmo es el que se mencionaba en el apartado de complejidad computacional). El principal inconveniente de este método es la cantidad de tiempo computacional que requiere, restringiendo su uso a pequeños problemas, incluso con la tecnología actual.

Heurísticas

La utilización de heurísticas (algoritmos que dan lugar a soluciones casi-óptimas[2]) es la opción más elegida cuando se trabaja con el TSP desde el punto de vista práctico. Las heurísticas pueden ser divididas en dos grandes grupos: los que buscan tratan de buscar una única solución factible y los que tratan de mejorar una solución dada. Entre las primeras se encuentra el "algoritmo del vecino más próximo"y, entre las últimas, las heurísticas de intercambio.

El diseño de algoritmos de aproximación lleva consigo el análisis de la solución que ofrecen. Dada una heurística, ¿qué se puede decir de la solución que produce en relación a la solución óptima? Existen principalmente tres formas de responder a esta pregunta:

- La primera forma de evaluar heurísticas, llamémosla empírica, con la que se trabajó consiste en comparar la solución óptima de una serie de problemas de "prueba"con la solución ofrecida por la heurística. Se trata de una forma muy sencilla de medir la bondad de un algoritmo, pero tiene asociado un gran problema: cómo elegir los problemas que serán tomados como test de modo que sean representativos.

- Una manera alternativa de juzgar las heurísticas nació a finales de la década de los sesenta. La idea es calcular una cota máxima de la desviación, asegurando así que la aplicación del algoritmo nunca dará lugar a soluciones que se desvíen del máximo más de esa cantidad. Esta cota se calcula aplicando la heurística a problemas donde se sabe que no funcionará bien y suele ofrecerse en término de tanto por ciento de desviación. En el año 1976, Christofides [7] presentó un algoritmo de aproximación con un error máximo del 50 %, cota que fue calculada con el método anterior. El principal problema de este método es que estamos basando nuestro juicio sobre un algoritmo en cómo funciona en la peor de las situaciones. En ocasiones, haciendo uso del método anterior puede verificarse que esa cota máxima se alcanza en problemas extremadamente raros, por lo que podría ser interesante tener información de la heurística en términos probabilísticos.

- La tercera forma de evaluar un algoritmo de aproximación es el análisis probabilístico del mismo. Los resultados que se deriven de este método serán

[2]Las heurísticas tratan de encontrar en un tiempo razonable buenas soluciones, que normalmente se acercan a la solución óptima.

extremadamente útiles pues nos permitirán saber, entre otras, si la cota máxima se alcanza muy frecuentemente o no. La primera dificultad con la que nos encontramos es que este método ha de presuponer una distribución de probabilidad sobre el conjunto de los problemas del viajante. El problema de decidir hasta qué punto es razonable aplicar una u otra distribución es equiparable al problema de decidir qué problemas utilizar como "test". El segundo de los problemas es que, generalmente, los resultados de esta naturaleza tienden a ser asintóticos ("para n suficientemente grandes"), por lo que la bondad de un método sobre problemas pequeños se hace desconocida.

A continuación, se van a explicar algunas de las heurísticas más utilizadas en este campo: heurística del vecino más próximo, heurística de Christofides y heurísticas de intercambio. Para simplificar la notación y las explicaciones, nos vamos a centrar en el TSP simétrico.

Heurística del vecino más próximo

Esta heurística fue una de las primeras en ser definidas. De hecho, en la década de 1930, K. Menger [41] ya definió este método y observó que, generalmente, no daba lugar a soluciones óptimas. La importancia de este método se debe a su simplicidad. Este algoritmo se encuentra dentro de los algoritmos denominados "Greedy". Se trata de un conjunto de algoritmos que responden al siguiente esquema:

1. Sea L el conjunto de elementos elegibles y T el conjunto solución. Inicializar $T = \emptyset$.

2. Mientras $L \neq \emptyset$ o T no sea solución, hacer

   a) Elegir el "mejor"elemento de L (respecto de un criterio fijado a priori).

   b) Hacer $L = L \setminus e$ y $T = T \cup e$.

Este conjunto de algoritmos recibe el nombre de "Greedy"(en inglés codicioso), pues en cada iteración se busca "lo mejor", aunque esto implique obtener al final una solución que no sea óptima pues, en general, los algoritmos "Greedy"no generan soluciones óptimas.

En el caso del algoritmo más próximo, el criterio que fija el mejor elemento es la proximidad al último elemento incluido en T. De forma general, el algoritmo del vecino más próximo sigue el siguiente esquema:

1. Elegir $M = \{i_1\}$ arbitrario, $j = 1$, $T = \emptyset$.

2. Mientras $M \neq V$ ó T no pase por todos los nodos, hacer:

   a) Elegir $i_{j+1} \in V \setminus M$ tal que $c_{i_j,i_{j+1}} = min\left\{c_{i_j,k}/k \in V \setminus M\right\}$.

   b) Hacer $M = M \cup i_{j+1}$, $T = T \cup (i_j, i_{j+1})$, $j = j + 1$.

3. Terminar: Hacer $T = T \cup \{(i_n, i_1)\}$.

En la Figura 1.6 se puede observar un ejemplo de resolución a través de este método. Obsérvese que en la primera iteración existen 2 arcos con igual coste. Por ello, se han obtenido dos soluciones a través de este método, una comenzando con cada uno de los arcos. La solución (a), que además coincide con la solución óptima, tiene un coste menor que la solución (b).

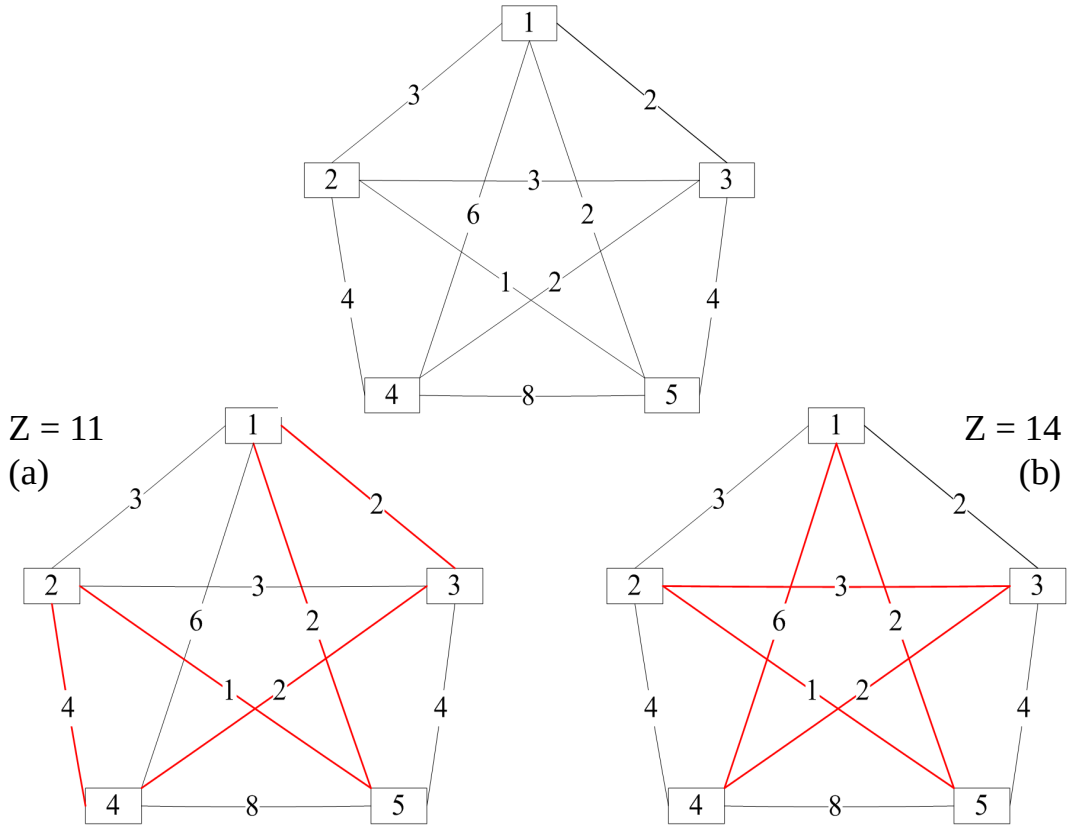


Figura 1.6: Heurística del vecino más próximo

Heurística de Christofides

Esta heurística, como se deduce de su nombre, fue definida por Christofides (1976)[7]. La base de este algoritmo es la relación existente entre árboles de mínimo coste y los problemas del viajante.

Un árbol de mínimo coste es una colección de $(n-1)$ arcos que unen todas las ciudades de un grafo con un coste mínimo. Existen varios algoritmos que resuelven este problema en tiempo polinomial. Si las distancias vienen dadas en una matriz C (de tamaño n x n), el árbol de mínimo coste puede encontrarse en un tiempo $O(n^2)$, cantidad deseable teniendo en cuenta que C tiene un tamaño proporcional a n^2. Por lo que los problemas de mínimo coste, al contrario que el TSP, pueden ser resueltos eficientemente. Además, la resolución de un problema de este tipo nos proporciona una cota mínima del coste de la solución óptima al TSP: nótese que si se elimina un arco del circuito solución, se obtiene un árbol (que consiste en un único camino entre todas las ciudades), por lo que el coste de la solución óptima deberá ser estrictamente mayor que el coste del árbol de mínimo coste (como mínimo ha de contener un arco más que cierre el circuito).

En el caso de matrices de distancias que cumplen la desigualdad triangular ($c_{ij} \leq c_{ik} + c_{kj} \forall k \in V$) se puede obtener también una cota máxima del coste óptimo. Supongamos que se desea buscar una ruta para visitar todas las ciudades y que sólo se pueden utilizar los arcos del árbol de mínimo coste. En ese caso, si duplicamos dicho árbol y le damos una orientación distinta a los 2 arcos que unen las mismas ciudades, obtendremos un circuito que visite todas las ciudades con un coste igual al doble del coste del árbol de mínimo coste. El problema de esta ruta, que es lo que hace que no sea la ruta óptima del TSP, es que algunas ciudades se visitan más de una vez. Pues bien, si aplicamos lo que se denomina atajos (seguir la ruta planteada hasta que éste nos haga visitar una ciudad ya visitada y en ese punto unir la ciudad en la que nos encontremos con la siguiente ciudad no visitada según el orden establecido por el circuito hasta que todos los vértices estén unidos) obtendremos un circuito que visite todas las ciudades una única vez y que tendrá un coste inferior a 2 veces el del árbol de mínimo coste y, por ende, inferior a 2 veces el coste de la ruta óptima. El algoritmo recién descrito recibe el nombre de algoritmo de árbol de mínimo coste y tiene múltiples variantes. Ahora bien, si lo que se busca es mejorar esa cota de 2 veces el coste óptimo, se debe recurrir al algoritmo de Christofides pues, como ya se mencionó anteriormente, la mejor cota de este tipo encontrada para una heurística del TSP está asociada a este algoritmo.

La idea de Christofides fue mejorar el circuito sobre el que luego se aplican los atajos, pues es lo que luego dará lugar a la cota máxima. Para ello, se recurre a los grafos eulerianos, que se definen como aquellos grafos conexos cuyos vértices tienen todos grado par. En un grafo euleriano es relativamente fácil encontrar un circuito euleriano, es decir, un circuito que recorra cada arco exactamente una vez, por lo que una vez obtenido un circuito euleriano puede obtenerse un circuito hamiltoniano (en el caso anterior, la forma de encontrar un circuito euleriano consistía en duplicar el árbol de mínimo coste). Christofides propuso recurrir al 1-emparejamiento perfecto para obtener dicho circuito euleriano. Un 1-emparejamiento perfecto es una colección de arcos tales que cada vértice es extremo de un único arco. Dado un conjunto de vértices, puede demostrarse que el 1-emparejamiento perfecto de coste mínimo puede obtenerse en un tiempo $O(n^3)$.

El algoritmo de Christofides comienza obteniendo el árbol de mínimo coste. Algunos de los vértices de este árbol tendrán grado par, por lo que no será necesario añadirles ningún arco más para conventirlos en un grafo euleriano. Pero no ocurre lo mismo con los nodos de grado impar (deberá haber un número par de éstos pues la suma de los grados de todos los vértices ha de ser par). Una forma simple de conseguir que todos los vértices tengan grado par es añadir el 1-emparejamiento perfecto de coste mínimo de los vértices de grado impar. Esto hará que el grado de los nodos impares se incremente en una unidad, convirtiéndolos así en nodos de grado par. De esta forma, se obtiene un grafo euleriano. Si aplicamos los atajos explicados anteriormente, obtendremos un circuito hamiltoniano que será solución del TSP. Christofides demostró que, como máximo, la longitud de la ruta obtenida será 1.5 veces la de la ruta óptima si la matriz de distancias cumple la desigualdad triangular. La demostración se basa en la idea de que el 1-emparejamiento perfecto de los vértices de grado impar debe tener un coste inferior a la mitad del coste óptimo (obsérvese en la Figura 1.7 que los nodos de grado impar dan lugar a dos 1-emparejamientos distintos (M y M') que de forma conjunta definen un circuito entre dichos vértices). Dado que se cumple la desigualdad triangular, el coste de los dos 1-emparejamiento debe ser inferior al coste de la ruta óptima y, por tanto, ambos 1-matching deben tener coste menor o igual que la mitad del coste del circuito óptimo. Como, además, el árbol de mínimo coste tiene coste estrictamente menor que la ruta óptima, obtenemos que el circuito obtenido a través de este algoritmo debe tener coste inferior a 3/2 el coste óptimo.

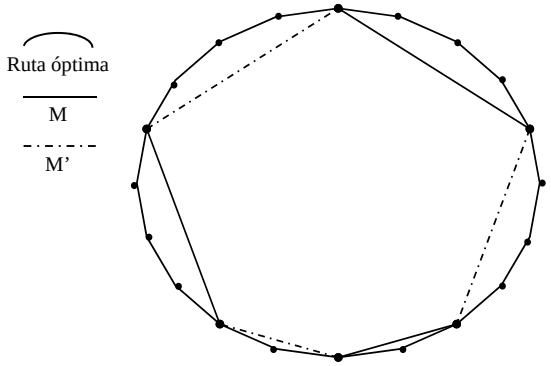


Figura 1.7: $coste(M) + coste(M') \leq coste(\text{Ruta óptima})$

Veamos, por último, un ejemplo paso a paso de resolución de un TSP a través del algoritmo de Christofides. Nótese en la Figura 1.8 que una vez se unen T y M ya se obtiene un circuito hamiltoniano y, por tanto, el algoritmo termina en ese punto.

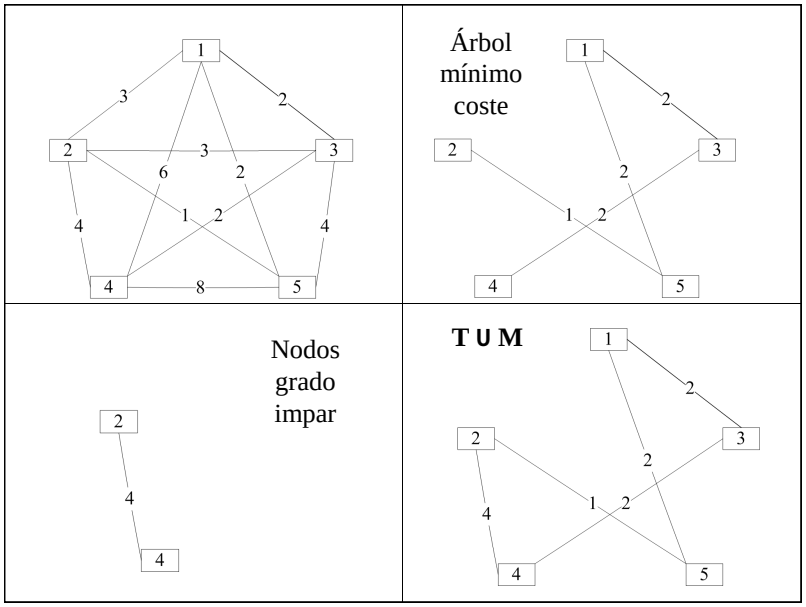


Figura 1.8: Ejemplo algoritmo Christofides

Heurísticas de intercambio

Las heurísticas de intercambio se engloban dentro de ese otro grupo de heurísticas definido anteriormente que tienen como fin mejorar soluciones ya existentes. Las heurísticas de intercambio fueron definidas por primera vez por Flood [21], que observó que en ocasiones un buen circuito puede obtenerse a base de intercambiar pares de arcos por alternativas de menor coste. En la Figura 1.9 puede

observarse que la suma de las longitudes de los arcos AC y BD es superior que la de los arcos AB y CD por lo que se obtendrá un circuito de menor coste si se unen los arcos AB y CD en lugar de AC y BD. Esa es precisamente la idea que subyace en las heurísticas de intercambio: examinar cada posible intercambio, lo que tiene una complejidad $O(n^2)$, e intercambiar los arcos que den lugar a una reducción del coste total. Este procedimiento puede llevarse a cabo para intercambios de 3 o más arcos, pero a efectos prácticos no se suele realizar para valores superiores a 3.

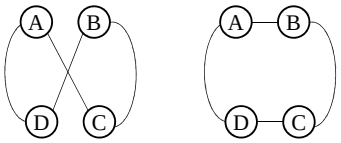


Figura 1.9: Intercambio de arcos

Para terminar, a modo de ejemplo se va a aplicar esta heurística de mejora sobre el resultado obtenido al aplicar el algoritmo del vecino más próximo (Figura 1.6 (b)). Obsérvese que los arcos (2,3) y (1,4) tienen un coste total de 9 unidades, mientras que los arcos (1,3) y (2,4) tienen un coste de 6 unidades. Aplicando esta heurística de mejora se ha obtenido, por tanto, una reducción de aproximadamente el 13 % del coste total.

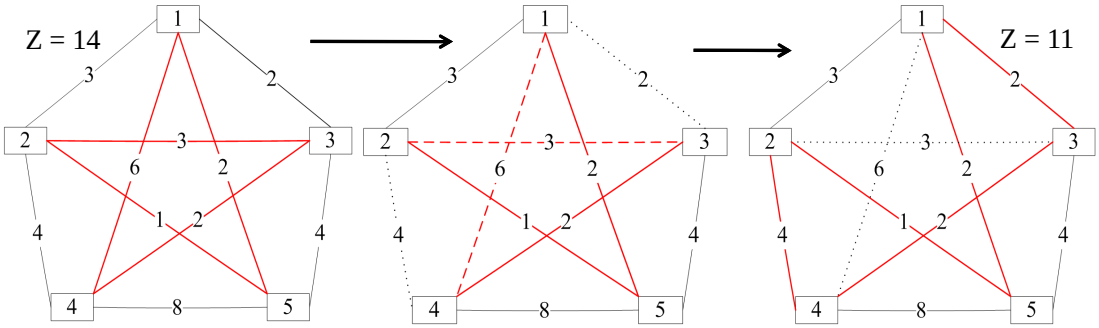


Figura 1.10: Ejemplo de heurística de intercambio

1.3. Problemas de Rutas de vehículos

1.3.1. Introducción

El Problema de Rutas de Vehículos (en inglés Vehicle Routing Problem VRP) es otro de los problemas más estudiados en Investigación Operativa. Definido hace más de 50 años, este problema consiste en diseñar el conjunto óptimo de rutas para una flota de vehículos que deben servir (o visitar) a un determinado número de clientes, como se ha visto anteriormente. El gran interés en este tipo de problemas, como ocurre con el TSP, se debe a su gran utilidad en problemas reales, así como a su considerable dificultad.

El Problema de Rutas de Vehículos es un nombre genérico que se da a una clase muy extensa de problemas consistentes en encontrar rutas óptimas de reparto desde uno o varios almacenes a un determinado número de ciudades o clientes de manera que se satisfagan ciertas restricciones. En otras palabras, el objetivo de este tipo de problemas es repartir (o recoger) una cierta mercancía a un conjunto de clientes con demandas conocidas de manera que el coste total originado de este reparto sea mínimo. Como se puede deducir, el VRP es una extensión del TSP donde la ciudad de origen es el depósito, denotado generalmente como nodo 0 (nótese que la notación varía con respecto al TSP donde la ciudad de origen es denotada como nodo 1), y se añaden nuevas restricciones.

Como se deduce de la definición, el VRP resulta extremadamente útil no sólo en problemas relacionados con el reparto y la recogida de bienes, sino también en una gran variedad de problemas reales ligados con los campos del transporte, la logística y la distribución. Ejemplos típicos donde se ha empleado este tipo de problemas son: rutas escolares, sistemas de recogida de basuras, limpieza de calles, reparto de mercancías y de correo, rutas de vendedores, etc. En general, y no sólo en los ejemplos anteriores, el proceso de transporte de mercancías se encuentra presente en muchos de los sistemas de producción, representando una parte importante (entre el 10 % y el 20 %) del coste final del producto. Así, la utilización de este tipo de problemas o procedimientos ha dado lugar a un ahorro de entre el 5 % y el 20 % en el coste total de transporte [Toth y Vigo (2001) [60]].

El problema de rutas de vehículos es un problema de programación entera que también se encuadra dentro de la categoría de los problemas NP-duro. Para este tipo de problemas, cuando el tamaño del mismo es excesivamente grande,

es deseable obtener soluciones aproximadas que puedan ser obtenidas con una rapidez relativa y que sean lo suficientemente parecidas a la solución óptima. Generalmente, la literatura sobre VRP se centra en este tema: encontrar una solución exacta que no requiera tanto esfuerzo computacional o encontrar una solución aproximada que, en un tiempo razonable, dé lugar a soluciones aceptables.

En la literatura científica, Dantzig y Ramser [13] fueron los primeros autores que trataron este tema, cuando estudiaron la aplicación real en la distribución de gasolina para estaciones de carburante. En su artículo "The Truck Dispatching Problem (1959)"proponían una solución basada en una formulación de programación lineal que daba lugar a un solución casi-óptima. El objetivo era encontrar una forma de asignar los camiones a las estaciones de servicio de manera que se satisficieran las demandas de éstas y la distancia recorrida por la flota de camiones fuese mínima. Para ellos, el problema no es más que una generalización del problema del viajante en el que se obliga a éste a visitar la ciudad de origen cada vez que haya visitado m de las n-1 ciudades restantes. Para valores de m y de n conocidos, el objetivo es encontrar bucles de forma que todos ellos tengan un punto en común y la distancia recorrida sea mínima. El problema así definido pasaría a denominarse "Clover Leaf Problem"(el problema de las hojas de trébol). Si m fuese pequeño, el problema podría resolverse fácilmente observando un plano en el que se encontraran las ciudades a visitar y buscando conjuntos (o clusters) de ciudades. Sin embargo, cuando los clusters son difíciles de determinar y m es grande (o incluso indeterminado), el problema se complica y se debe recurrir a otro tipo de soluciones, como la que se presentó en el artículo.

1.3.2. Variantes del VRP

Como ya se ha comentado, el término VRP se refiere a una colección muy extensa de problemas que se encuadran dentro de la definición anterior. En este apartado se van a explicar algunas de las variantes del VRP y se dará su definición formal dentro de la teoría de grafos.

VRP clásico

En el caso del VRP clásico, los clientes, sus demandas y las distancias entre ellos son conocidas. Así, el objetivo de este tipo de problemas es encontrar un conjunto de rutas de mínimo coste que cumplan:

- Cada ciudad o cliente han de ser visitados una única vez por un único vehículo.
- Todas las rutas deben comenzar y terminar en el depósito.

Según la teoría de grafos, el VRP puede ser definido formalmente de la siguiente manera: Sea $G = (V, A)$ un grafo, donde $V = \{0, ..., n\}$ es el conjunto de vértices, o clientes, con el depósito ubicado en el vértice 0, y A es el conjunto de arcos que los unen. Para cada arco $(i, j), i \neq j$, existe un coste no negativo asociado c_{ij}, que suele ser interpretado como el coste o el tiempo en el que se incurre al viajar de i a j. Generalmente, estos costes vienen dados en forma de matriz C. Cuando C es simétrica, normalmente A es reemplazado por un conjunto E de arcos no dirigidos. Además, suponemos que hay m vehículos disponibles en el depósito y se cumple que $m_L < m < m_U$. Cuando $m_L = m_U$, se dice que m está fijo; mientras que si $m_L = 1$ y $m_U = n - 1$, se dice que m es libre.

Definido en estos términos, el VRP clásico es equivalente al TSP con varios viajantes. Por tanto, no es de extrañar que el VRP clásico se haya clasificado también como NP-duro.

VRP con capacidad limitada (CVRP)

Este problema es un VRP clásico en el que los vehículos de la flota tienen una capacidad determinada e igual para todos ellos. El objetivo es el mismo pero debemos añadir la restricción de la capacidad: la suma de las demandas de las ciudades que se visitan en cada ruta no puede exceder la capacidad del vehículo.

Formalmente, el CVRP se define como: Sea $G = (V, A)$ un grafo donde $V = \{0, ..., n\}$ es el conjunto de vértices, o clientes, con el depósito ubicado en el vértice

0, y A es el conjunto de arcos que los unen. Para cada arco $(i, j), i \neq j$, existe un coste no negativo asociado c_{ij} que suele ser interpretado como el coste o el tiempo en el que se incurre al viajar de i a j. Generalmente, el uso de arcos de la forma (i, i) no está permitido lo que se suele imponer fijando $c_{ii} = \infty \forall i \in V$. Si G es un grafo dirigido, entonces la matriz de costes es asimétrica y el VRP recibe el nombre de VRP asimétrico (o simplemente VRP); por el contrario, si $c_{ij} = c_{ji} \forall (i, j) \in A$, el problema pasa a denominarse VRP simétrico (SVRP) y el conjunto A es sustituido por un conjunto E de arcos no dirigidos.

Además, a cada cliente i $(i = 1, .., n)$ se le asocia una demanda no negativa d_i que le debe ser entregada. Al depósito se le asigna una demanda ficticia $d_0 = 0$. Dado un conjunto $S \subseteq V$, se define $d(S) = \sum_{i \in S} d_i$ como la demanda total del conjunto.

Un conjunto de K vehículos iguales, todos con la misma capacidad C, está disponible en el depósito. Para asegurar la viabilidad, se asume que $d_i \leq C$ y que K no es más pequeño que K_{min}. El valor de K_{min} se determina a través del problema de cubicación (Bin Packing Problem) asociado al CVRP. El BPP también es un problema NP-duro y consiste en determinar el número mínimo de cajas, con un volumen determinado, que son necesarias para transportar ciertos objetos de los que se conoce su volumen. K_{min} puede ser sustituido por $\lceil d(V)/C \rceil$, que es una cota mínima a la solución del BPP.

VRP con ventanas de tiempo (VRPTW)

Este problema es una variante del anterior en el que se añade la restricción de que las entregas a los clientes se deben hacer en unos intervalos de tiempo precisos, llamados ventanas de tiempo. Por tanto, no sólo bastará con saber las demandas de los clientes y las distancias entre ellos, sino que también será necesario conocer la ventana de tiempo de cada uno de ellos, el momento en el que comienzan las rutas, el tiempo necesario para viajar de un cliente a otro y para servir a cada uno de ellos.

Formalmente, definimos el VRPTW de la siguiente manera: Sea $G = (V, A)$ un grafo donde $V = \{0, ..., n\}$ es el conjunto de vértices, o clientes, con el depósito ubicado en el vértice 0, y A es el conjunto de arcos que los unen. Para cada arco $(i, j), i \neq j$, existe un coste no negativo asociado c_{ij}. Asociado a cada arco también existe una cantidad t_{ij}, que se corresponde con el tiempo necesario para ir de i a j. Por otro lado, a cada vértice se le asocia una demanda d_i; un tiempo de servicio s_i necesario para servir al cliente y una ventana de tiempo $[a_i, b_i]$ tal

que el vehículo debe llegar al nodo i en ese intervalo. Generalmente, los costes c_{ij} y los tiempos t_{ij} coinciden y las ventanas de tiempo se definen suponiendo que los vehículos parten del depósito en el instante 0.

Por lo tanto, el VRPTW consiste en encontrar una colección de exactamente K circuitos de coste mínimo tales que:

- Cada circuito visite el depósito.
- Cada cliente sea visitado por un único circuito.
- La suma de las demandas de los nodos visitados por cada circuito no supere C.
- Para cada cliente i, el servicio comience dentro de la ventana de tiempo $[a_i, b_i]$ y el vehículo se detenga s_i unidades de tiempo.

Lógicamente, el VRPTW también es NP-duro, puesto que generaliza el VRP clásico. El VRP y el VRPTW coincidirán cuando $a_i = 0$ y $b_i = \infty, \forall i \in V \setminus \{0\}$ y $C = \infty$.

VRP de ida y vuelta (VRPB)

El VRP de ida y vuelta (en inglés, VRP with Backhauls) es una extensión del CVRP, en el que los clientes $V \setminus \{0\}$ están divididos en dos subconjuntos. El primero de ellos, L, contiene n clientes "de ida"(en inglés *linehaul*) para los que una cantidad de producto debe ser distribuida. El segundo, B, contiene m clientes "de vuelta"(en inglés *backhauls*) para los que una cantidad de producto debe ser recogida. Los clientes se numeran de manera que $L = \{1, \ldots, n\}$ y $B = \{n+1, \ldots, n+m\}$. En este tipo de problemas existe, además, una relación de precedencia entre los dos tipos de clientes: siempre que una ruta contenga los dos tipos de clientes, los *linehaul* deben ser servidos antes que los *backhaul*. Una cantidad no negativa d_i, que será repartida o recogida según su tipo, se asocia a cada cliente i, y al depósito se le asocia una demanda ficticia $d_0 = 0$. La Figura 1.11 muestra un ejemplo de este problema.

Formalmente, el VRPB se define como: Sea $G = (V, A)$ un grafo, donde $V = \{0, ..., n\}$ es el conjunto de vértices, o clientes, con el depósito ubicado en el vértice 0, y A es el conjunto de arcos que los unen. El conjunto de clientes $V \setminus \{0\}$ está dividido en dos subconjuntos, que forman una partición, denominados L y B. Para cada arco $(i, j), i \neq j$, existe un coste no negativo asociado c_{ij} y para

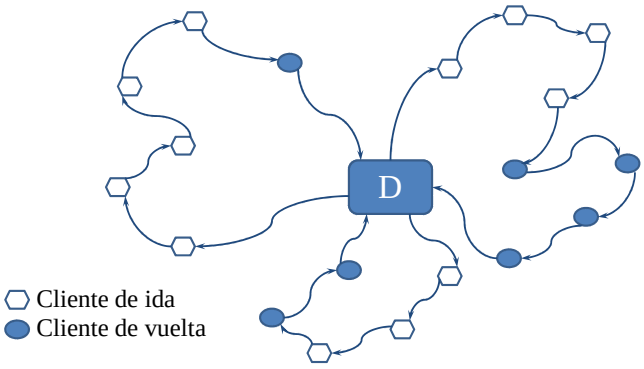


Figura 1.11: Ejemplo de VRPB

cada vértice existe una demanda d_i. Además, existen K vehículos disponibles en el depósito. Para asegurar la existencia de soluciones factibles, debe cumplirse que $K \geq max\{K_L, K_B\}$, siendo K_L y K_B el número mínimo de vehículos necesarios para servir a los clientes *linehaul* y *backhaul*, respectivamente. Estos valores se obtienen resolviendo los BPP asociados a los subconjuntos de clientes.

Por lo tanto, El VRPB consiste en encontrar K circuitos de mínimo coste tales que:

- Cada circuito visite el depósito.
- Cada cliente sea visitado por un único circuito.
- La demanda total de los clientes *linehaul* y *backhaul* visitados en cada circuito no supere, de forma separada, la capacidad C del vehículo.
- En cada circuito, los clientes *linehaul* preceden a los clientes *backhaul*.

De nuevo, el VRPB es NP-duro, en tanto que es una generalización del VRP básico. Ambos problemas coincidirán cuando $B = \emptyset$.

VRP con Recogida y Reparto (VRPPD)

De nuevo, el VRPPD es una extensión del CVRP. En este caso, a los clientes se les permite enviar y recibir cierta cantidad de producto entre ellos. Por lo tanto, deberá conocerse las cantidades que desean enviar y recibir y el destino u origen de dicho producto. Se asume que en cada vértice el reparto se realiza antes que la recogida, por lo que la carga del vehículo al llegar a un cierto cliente será la carga inicial menos las demandas ya repartidas más las demandas recogidas. La Figura 1.12 contiene un ejemplo de un VRPPD.

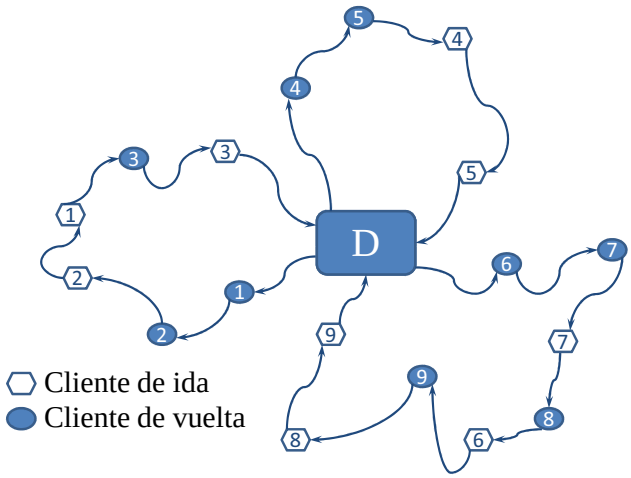


Figura 1.12: Ejemplo de VRPPD

Formalmente, el VRPPD se define como: Sea $G = (V, A)$ un grafo donde $V = \{0, ..., n\}$ es el conjunto de vértices, o clientes, con el depósito ubicado en el vértice 0, y A es el conjunto de arcos que los unen. Para cada arco $(i, j), i \neq j$, existe un coste no negativo asociado c_{ij}. Para cada vértice, existe una demanda d_i asociada a la cantidad que le será repartida y una cantidad p_i asociada a la cantidad que deberá ser recogida. En ocasiones, sólo se cuenta con una cantidad $d'_i = d_i - p_i$, que se refiere a la diferencia neta entre la cantidad entregada y la recogida (puede tomar por tanto un valor negativo). Además, se definen los valores O_i y D_i como los vértices que son origen y destino de las demandas de i, respectivamente.

El VRPPD consiste en encontrar los K circuitos de mínimo coste tales que:

- Cada circuito visite el depósito.
- Cada cliente sea visitado por un único circuito.
- La carga del vehículo durante el circuito no debe ser negativa y nunca puede exceder la capacidad C del vehículo.
- Para cada cliente i, el cliente O_i, cuando sea diferente del depósito, debe ser servido en el mismo circuito y antes que i.
- Para cada cliente i, el cliente D_i, cuando sea diferente del depósito, debe ser servido en el mismo circuito y después que i.

De nuevo, el VRPPD es NP-duro, pues generaliza el CVRP, lo que ocurre cuando $O_i = D_i = 0$ y $p_i = 0, \forall i \in V$.

1.3.3. Formulación del VRP

Como ocurre con el TSP, existen múltiples formas de plantear el problema de programación lineal entera asociada al VRP. En este apartado se van a revisar algunas de las más frecuentes. Las dos primeras se encuentran dentro de lo que se conoce como Formulaciones de flujo de vehículos y se utilizan variables enteras asociadas a cada nodo o arco que miden el número de veces que se visitan esos nodos o arcos. La última de ellas, recibe el nombre de Modelos de partición de conjuntos (Set Partitioning (SP)).

Empezamos describiendo una formulación de dos índices para el CVRP asimétrico, que utiliza $O(n^2)$ variables binarias x tales que x_{ij} toma el valor 1 si el arco $(i, j) \in A$ pertenece a la solución y 0, en otro caso. Así, el problema se define de la siguiente forma:

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

sujeto a

$$\sum_{j \in V} x_{ij} = 1, \ \forall i \in V \setminus \{0\} \tag{1.1}$$

$$\sum_{i \in V} x_{ij} = 1, \ \forall j \in V \setminus \{0\} \tag{1.2}$$

$$\sum_{i \in V} x_{i0} = K \tag{1.3}$$

$$\sum_{j \in V} x_{0j} = K \tag{1.4}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S) \ \forall S \subseteq V \setminus \{0\}, S \neq \emptyset \tag{1.5}$$

$$x_{ij} \in \{0, 1\} \ \forall i, j \in V. \tag{1.6}$$

Las restricciones (1.1) y (1.2) imponen que sólo un arco entre y salga de cada vértice, respectivamente. Análogamente, (1.3) y (1.4) imponen que del depósito salgan y vuelvan K vehículos. Las llamadas restricciones de capacidad (Capacity-Cut Constraints (CCC)) de (1.5) imponen las restricciones de capacidad y de conexión de las soluciones. Estas restricciones estipulan que cada partición $(S, V \setminus S)$ debe ser atravesada por un número de arcos que no puede ser inferior a $r(S)$ (número mínimo de vehículos necesarios para servir al conjunto S). Generalmente,

este valor $r(S)$ se calcula a partir de un BPP, pero las restricciones siguen siendo válidas si $r(S)$ es sustituido por $\lceil d(S)/C \rceil$.

Una formulación alternativa puede obtenerse transformando las CCC (1.5) en las ya mencionadas restricciones de rotura de subcircuito (estas restricciones fueron estudiadas detalladamente en el apartado correspondiente al TSP):

$$\sum_{i \in S} \sum_{j \in S} \leq |S| - r(S) \tag{1.7}$$

lo que implica que al menos $r(S)$ arcos deben abandonar el conjunto S. Tanto las restricciones (1.5) como las (1.7) tienen un cardinal que crece exponencialmente, con n lo que hace prácticamente imposible resolver directamente este problema de programación lineal entera.

Para evitar este problema, se puede definir otra familia de restricciones con cardinal polinómico:

$$u_i - u_j + C x_{ij} \leq C - d_j \ \forall i, j \in V \setminus \{0\}, i \neq j \tag{1.8}$$

$$d_i \leq u_i \leq C \ \forall i \in V \setminus \{0\} \tag{1.9}$$

donde $u_i, i \in V \setminus \{0\}$, es una variable continua adicional que representa la carga del vehículo tras visitar el nodo i. Es fácil observar que las restricciones (1.8) y (1.9) imponen las restricciones de capacidad y de conexión del CVRP. De hecho, cuando $x_{ij} = 0$, la restricción (1.8), no "restringe", pues $u_i \leq C$ y $u_j \geq d_j$; y cuando $x_{ij} = 1$ imponen que $u_j \geq u_i + d_j$. Nótese que de esta forma también se consigue eliminar los subcircuitos.

Este clase de formulación, en la que se usan variables de decisión de doble índice, ha sido muy utilizada para variantes sencillas del VRP (CVRP o VRPB), pero generalmente es inadecuada para variantes más complejas del VRP. De hecho, sólo puede ser utilizada cuando el coste total de la solución pueda ser expresado como suma de los costes asociados a los arcos visitados. Por lo tanto, este tipo de formulación no es apto para problemas donde el coste de la solución dependa de la secuencia global de ciudades o del tipo de vehículo que se asigne a cada ruta (no hay una forma directa de saber qué vehículo atraviesa un arco en particular).

Una forma de solventar algunos de los inconvenientes de la formulación anterior es indicar explícitamente el vehículo que recorre cada arco. Una de las múltiples formas de hacerlo es recurrir a una formulación de 3 índices. Dicha for-

mulación tiene $O(n^2K)$ variables binarias x: x_{ijk} toma el valor 1 si el arco (i, j) es atravesado por el vehículo k en la solución. Además, hay $O(nK)$ variables binarias $y_{ik}(i \in V; k = 1, ..., K)$, que toman el valor 1 si el nodo i es visitado por el vehículo k en la solución, y 0 en otro caso. El CVRP viene dado entonces por:

$$min \sum_{i \in V} \sum_{j \in V} c_{ij} \sum_{k=1}^{K} x_{ijk}$$

sujeto a

$$\sum_{k=1}^{K} y_{ik} = 1, \ \forall i \in V \setminus \{0\} \tag{1.10}$$

$$\sum_{k=1}^{K} y_{0k} = K \tag{1.11}$$

$$\sum_{j \in V} x_{ijk} = \sum_{j \in V} x_{jik} = y_{ik}, \ \forall i \in V, k = 1, ..., K \tag{1.12}$$

$$\sum_{i \in V} d_i y_{ik} \leq C \ \forall k = 1, ..., K \tag{1.13}$$

$$\sum_{i \in S} \sum_{j \notin S} x_{ijk} \geq y_{hk} \ \forall S \subseteq V \setminus \{0\}, h \in S, k = 1, ..., K \tag{1.14}$$

$$y_{ik} \in \{0, 1\} \ \forall i \in V, k = 1, ..., K \tag{1.15}$$

$$x_{ijk} \in \{0, 1\} \ \forall i, j \in V, k = 1, ..., K. \tag{1.16}$$

Las restricciones (1.10)-(1.12) imponen que cada cliente sea visitado una vez, que K vehículos abandonen el depósito y que el mismo número de vehículos entren y salgan de cada cliente, respectivamente. Las ecuaciones (1.13) son las restricciones de capacidad de los vehículos y (1.14) son las restricciones de rotura de subcircuito. De nuevo, esta última puede ser reemplazada por:

$$\sum_{i \in S} \sum_{j \in S} x_{ijk} \leq |S| - 1 \ \forall S \subseteq V \setminus \{0\}, |S| \geq 2, k = 1, ..., K, \tag{1.17}$$

que impone que, para cada vehículo k, al menos un arco abandone cada conjunto de vértices visitado por k y que no contenga el depósito. Alternativamente, y de forma análoga a las restricciones (1.8) y (1.9), se pueden usar las siguientes

restricciones:

$$u_{ik} - u_{jk} + Cx_{ijk} \leq C - d_j,\ \forall i, j \in V \setminus \{0\}, k = 1, ..., K \tag{1.18}$$

$$d_i \leq u_{ik} \leq C\ \forall i \in V \setminus \{0\}, k = 1, ..., K \tag{1.19}$$

Nótese que estas restricciones también reemplazan las restricciones de capacidad (1.13).

La formulación con tres índices ha sido ampliamente utilizada para modelizar versiones más complejas y con mayor número de restricciones del VRP, tales como el VRPTW. La principal desventaja de esta formulación es el número tan grande de variables requeridas. Por otro lado, generalizan los modelos de dos subíndices, lo que puede ser obtenido de la siguiente manera: $x_{ij} = \sum_{k=1}^{K} x_{ijk} \forall (i, j) \in A$. Veamos, a continuación, la formulación del VRP con ventanas de tiempo utilizando esta formulación de tres subíndices (obsérvese que para esta notación se ha supuesto que sólo hay un vehículo disponible que realiza las k rutas):

$$min \sum_{i \in V} \sum_{j \in V} \sum_{k=1}^{K} c_{ij} x_{ijk}$$

sujeto a

$$\sum_{j \in V} x_{ijk} = y_{ik},\ \forall i \in V \setminus \{0\}, k = 1, ..., K \tag{1.20}$$

$$\sum_{k=1}^{K} y_{ik} = 1,\ \forall i \in V \setminus \{0\} \tag{1.21}$$

$$\sum_{i \in V} x_{ihk} = \sum_{j \in V} x_{hjk},\ \forall h \in V, k = 1, ..., K \tag{1.22}$$

$$\sum_{i \in V} x_{i0k} = 1,\ \forall k = 1, ..., K \tag{1.23}$$

$$\sum_{j \in V} x_{0jk} = 1,\ \forall k = 1, ..., K \tag{1.24}$$

$$\sum_{i \in V} d_i y_{ik} \leq C,\ \forall k = 1, ..., K \tag{1.25}$$

$$t_i^k + s_i + t_{ij} - M(1 - x_{ijk}) \leq t_j^k,\ \forall (i, j) \in A, k = 1, ..., K \tag{1.26}$$

$$a_i y_{ik} \leq t_i^k \leq b_i y_{ik},\ \forall i \in V \setminus \{0\}, k = 1, ..., K \tag{1.27}$$

$$t_0^1 \geq \sigma^1 \tag{1.28}$$

$$t_{n+1}^k + \sigma^{k+1} \leq t_0^{k+1},\ \forall k = 1, ..., K-1 \tag{1.29}$$

$$\sigma^k = \beta \sum_{i \in V} s_i y_{ik},\ \forall k = 1, ..., K \tag{1.30}$$

$$y_{ik} \in \{0, 1\}\ \ \forall i \in V, k = 1, ..., K \tag{1.31}$$

$$x_{ijk} \in \{0, 1\}\ \ \forall i, j \in V, k = 1, ..., K \tag{1.32}$$

donde x_{ijk} toma el valor 1 si el arco (i, j) es atravesado por la ruta k en la solución y 0 en otro caso; y_{ik} toma el valor 1 si la ruta k visita el cliente i y 0, en otro caso; t_i^k es el instante en el que empieza el servicio al cliente i por la ruta k; t_0^k es el instante en el que empieza la ruta k y t_{n+1}^k es el instante en el que termina la ruta k. Además, σ^k es el tiempo necesario para "cargar"el vehículo que realice la ruta k. Las ecuaciones (1.26)-(1.29) sirven para asegurar que se cumplan las restricciones temporales.

Por último, veamos otro tipo de formulación que recibe el nombre de Modelos de partición de conjuntos (SP). La utilización de estos modelos para los problemas de rutas de vehículos fue propuesto por primera vez por Balinski y Quandt[1]. Se utiliza un número exponencial de variables binarias, cada una de las cuales representa una solución factible del problema. En el caso del VRP, estas soluciones factibles son circuitos que cumplen todas las restricciones que deban imponerse. Sea $\mathcal{H} = \{H_1, ..., H_q\}$ el conjunto de todos los circuitos de G con $q = |H|$. Cada circuito H_j tiene un coste asociado c_j. Además, se definen las variables dicotómicas a_{ij} que toman el valor 1 si el cliente i es visitado por la ruta j y 0 en otro caso. La variable binaria x_j toma el valor 1 si la ruta j pertenece a la solución y 0 en otro caso. El modelo es, por tanto:

$$min \sum_{j=1}^{q} c_j x_j$$

sujeto a

$$\sum_{j=1}^{q} a_{ij} x_j = 1,\ \ \forall i \in V \setminus \{0\} \tag{1.33}$$

$$\sum_{j=1}^{q} x_j = K \tag{1.34}$$

$$x_j \in \{0,1\}\,,\ \forall j = 1,...,q. \tag{1.35}$$

La restricción (1.33) impone que cada cliente sea visitado por uno de los circuitos seleccionados y (1.34) requiere que K circuitos sean seleccionados.

Este modelo tan general permite modelizar situaciones muy diversas, pues se pueden tener en cuenta restricciones de muy distinta índole, dado que la factibilidad viene implícita en la definición de los circuitos H. Por ello, el número de variables se dispara. Además, en ocasiones el problema surge a la hora de calcular dichos circuitos pues, de haber un número muy elevado de restricciones, la tarea puede complicarse.

Como se ha podido observar, existe una multitud de formas distintas de modelizar los VRP. Cada una de estas modelizaciones surge a la hora de intentar resolver esta clase de problemas que, como ya se ha mencionado anteriormente, es NP-duro. De un modo u otro, se intenta trabajar con un número razonable de variables y de restricciones que hagan viable la utilización de algún software informático donde implementarla.

1.3.4. Algunos métodos de resolución

Al igual que ocurre con el TSP, al tratarse ambos de problemas NP-duro, no se ha descubierto aún ningún algoritmo capaz de resolver estos problemas en un tiempo polinómico. El método consistente en evaluar todas las soluciones posibles para luego optar por la de menor coste se hace inviable para problemas de rutas de vehículos con muchas restricciones, como el VRPTW, o con un número de ciudades elevadas.

Como ocurre con todos los problemas complejos computacionalmente hablando, y como ya se explicó en el apartado anterior, la solución del VRP pasa por dos opciones: continuar con la búsqueda de algoritmos polinómicos y trabajar con los ya elaborados aunque se deba emplear mucho tiempo para obtener una solución; o trabajar con heurísticas que den lugar a soluciones casi-óptimas que no se alejen significativamente del valor óptimo pero que alcanzen una solución en una cantidad de tiempo razonable.

Entre los métodos desarrollados en la búsqueda de un algoritmo que ofrezca la solución óptima, podemos destacar la utilización de algoritmos de ramificación y acotación, muchos de ellos basados en la relación existente entre el TSP y el VRP, o de Programación dinámica, cuya filosofía ya ha sido explicada. Las siguientes publicaciones recogen muchos de los algoritmos exactos descritos hasta el momento: Laporte (1992) [36], Laporte y Osman (1995) [37] y Toth y Vigo (2002) [60]. Este último recoge los métodos exactos más efectivos propuestos en la literatura hasta el año 2002.

Heurísticas

Muchos de los algoritmos heurísticos definidos para el VRP se derivan de procedimientos establecidos para el TSP debido a la gran relación existente entre estos dos tipos de problemas. El algoritmo del vecino más próximo o las heurísticas de intercambio pueden ser aplicados al VRP casi sin modificaciones. No obstante, cuando se aplican estos métodos debe tenerse cuidado a la hora de crear las rutas, en el sentido de que sólo las rutas factibles, aquellas que cumplen con todas las restricciones impuestas por el problema, deben ser creadas.

Existen muchos tipos distintos de heurísticas, que pueden clasificarse en dos grupos: los algoritmos que buscan soluciones factibles relativamente buenas; y los algoritmos, llamémosles heurísticas de mejora, que tratan de, a partir de una solución factible, encontrar soluciones mejores que la dada. Dentro del primer

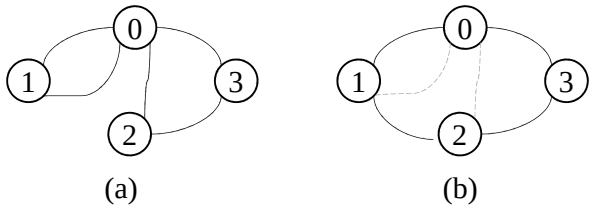


Figura 1.13: Fusión de dos rutas

grupo destacan el algoritmo de Clarke y Wright y el de Fisher y Jaikumar; mientras que, dentro de las heurísticas de mejora, destacamos: algoritmos genéticos (GAs), búsqueda tabú (TS), optimización de colonias de hormigas (AC) y recocido simulado (SA).

- Algoritmo de Clarke y Wright

Uno de los algoritmos clásicos en este ámbito fue definido por Clarke y Wright (1964) [8] para resolver CVRP en los que el número de vehículos es libre. Este algoritmo comienza con (n-1) rutas, cada una de ellas conteniendo el depósito y uno de los vértices. En cada paso, se unen dos rutas de acuerdo con el máximo ahorro que puede ser generado. La idea viene plasmada en la figura 1.13: en la parte (a) aparecen las dos rutas por separado; mientras que en la (b) las dos rutas se han unido evitando así el coste de los arcos marcados con línea discontinua. Los pasos a seguir por este algoritmo son los siguientes:

1. Crear n rutas de la forma $(0, i, 0), \forall i \in V \setminus \{0\}$.
2. Calcular los ahorros $s_{ij} = c_{i0} + c_{0j} - c_{ij}, \forall i, j = 1, ..., n$
3. Ordenar los ahorros de forma no decreciente.
4. Considerar las dos rutas que contengan $(i, 0)$ y $(j, 0)$ que hayan dado lugar a un s_{ij} mayor, respectivamente. Si $s_{ij} > 0$, unir las dos rutas introduciendo el arco (i, j) y eliminando los arcos $(i, 0)$ y $(j, 0)$. Si la ruta resultante es factible, implementar la fusión.
5. Repetir el paso anterior hasta que no existan más fusiones posibles.

Este algoritmo ha sido muy utilizado, pues da lugar a resultados relativamente buenos y, además, emplea un tiempo $O(n^2 log(n))$. Por otro lado, el algoritmo de Clarke y Wright no permite, o mejor dicho ignora, un número de vehículos fijo.

No obstante, de ser éste el caso, se puede obtener una solución repitiendo el paso 4 hasta llegar al número deseado de vehículos, incluso si esto implica incurrir en ahorros negativos.

- Algoritmo de Fisher y Jaikumar

El algoritmo de Fisher y Jaikumar (1981)[20] pertenece al grupo de los algoritmos de dos fases. Dichas heurísticas constan, como su propio nombre indica, de dos fases: la primera de ella consiste en hacer clusters de clientes y la segunda en establecer rutas dentro de los grupos creados en la fase anterior. Los distintos métodos empleados para crear dichos clusters y para establecer las rutas han dado lugar a una gran variedad de heurísticas de este tipo. Otros algoritmos de este grupo son: "The Petal algorithm", definido por Ryan et al. (1993) [51] o "The sweep algorithm", definido por primera vez por Wren (1971) [64]. El algoritmo de Fisher y Jaikumar fue diseñado para problemas CVRP con el número de vehículos fijo. Los pasos de este algoritmo son los siguientes:

1. Elegir aleatoriamente j_k vértices para inicializar cada uno de los K clusters.
2. Estimar el coste de asignar cada cliente i a cada cluster k: $h_{ik} = c_{0i} + c_{ij_k} + c_{j_k 0} - c_{0j_k}$.
3. Resolver un problema de asignación generalizada (GAP) con costes h_{ik}, demandas de los clientes d_i y capacidad de los vehículos C.
4. Resolver un TSP por cada cluster correspondiente a una solución del GAP.

Una de las ventajas de este algoritmo es que cada vez que se ejecuta se obtiene una solución diferente, pues los vértices iniciales se eligen aleatoriamente y pueden no elegirse los mismos cada vez, por lo que ejecutándolo un número suficiente de veces pueden obtenerse soluciones relativamente buenas. Por el contrario, tiene una gran desventaja: requiere la resolución de un problema TSP, el cual, como ya se ha explicado, es NP-duro.

- Heurísticas de mejora

Las heurísticas de mejora, denominadas en ocasiones metaheurísticas, en el VRP funcionan de dos formas: tomando cada ruta de forma independiente o tomando varias rutas a la vez. En el primero de los casos, dada una ruta, se

Figura 1.14: Cruce de arcos (SC)

podrá aplicar cualquiera de los procedimientos existentes para el TSP; mientras que para el segundo, se tendrán que definir nuevos algoritmos. A continuación, se van a describir brevemente algunas heurísticas del segundo tipo.

Heurísticas de intercambio

La filosofía de las heurísticas de intercambio para los problemas de rutas de vehículos es la misma que para el problema del viajante: cambiar ciertos arcos de forma que se consiga disminuir el coste de la solución. Varios autores, entre los que destacan Thomson y Psaraftis [59], Van Breedam [62] o Kindervater y Savelsbergh [33], han diseñado distintos planes de intercambio que han sido ampliamente utilizados posteriormente. Van Breedam clasifica las distintas operaciones posibles en cruce de arcos (String Cross), intercambio de arcos (String Exchange), reubicación de arcos (Strinf Relocation) y mezcla de arcos (String Mix). Estas operaciones se definen de la siguiente forma:

- Cruce de arcos (SC).- Dos cadenas de vértices se intercambian mediante el cruce de dos arcos de cada ruta (véase Figura 1.14).

- Intercambio de arcos (SE).- Dos cadenas de como máximo k vértices se intercambian entre las dos rutas (véase Figura 1.15).

- Reubicación de arcos (SR).- Una cadena de como máximo k vértices es trasladado de una ruta a otra, normalmente k vale 1 ó 2. (véase Figura 1.16).

- Mezcla de arcos (SM).- Se elige el mejor movimiento entre SE y SR.

Recocido simulado

Este algoritmo comienza con una solución x_1 y en cada iteración t busca una nueva solución x_{t+1} en el vecindario $N(x_t)$ de x_t. Si la solución x cumple que $f(x) \leq f(x_t)$, siendo f la función objetivo, entonces $x_{t+1} = x$. En caso contrario,

Figura 1.15: Intercambio de arcos

Figura 1.16: Reubicación de arcos

$$x_{t+1} = \begin{cases} x \text{ con probabilidad } p_t \\ x_t \text{ con probabilidad } 1 - p_t \end{cases}$$

donde p_t es una función decreciente en t y en $f(x) - f(x_t)$. Normalmente, p_t se define como:

$$p_t = exp(-\left[f(x) - f(x_t)\right]/\theta_t)$$

donde θ_t es la temperatura de la iteración t. La regla empleada para definir θ_t recibe el nombre de esquema de enfriamiento. Típicamente, θ_t es una función decreciente de t: inicialmente a θ_t se le asigna un valor predeterminado $\theta_1 > 0$ y se multiplica por un factor $\alpha(0 < \alpha < 1)$ cada T iteraciones, por lo que la probabilidad de aceptar una solución peor decrece con el tiempo.

Se continúa con este esquema de búsqueda hasta que se cumple uno de los siguientes criterios de parada: el valor de la función objetivo no ha descendido como mínimo π_1 % en las últimas k_1 iteraciones consecutivas; el número de movimiento válidos (se denomina así cuando se consigue que descienda $f(x_t)$) ha sido inferior al π_2 % de las últimas k_2 iteraciones consecutivas; o, se han realizado más de k_3 iteraciones.

Búsqueda Tabú

Igual que en el recocido simulado, la búsqueda tabú evalúa secuencias de soluciones pero, en este caso, la nueva solución (x_{t+1}) se elige por ser el mejor vecino de la solución x_t. Además, para evitar ciclos, las soluciones examinadas recientemente son prohibidas (se convierten en tabú) durante un número deter-

minado de iteraciones. para ahorrar espacio y tiempo computacional, en lugar de "recordar"la solución completa, se graban sólo algunas de sus características.

Uno de los primeros intentos de aplicar la búsqueda tabú a problemas VRP se debe a Willard [63]. En este caso, la vecindad se define como el conjunto de todas las soluciones factibles que pueden alcanzarse a partir de intercambios básicos de arcos, como los descritos para el TSP. En cada iteración, la nueva solución viene determinada por el movimiento no-tabú con mejores resultados. Desafortunadamente, este algoritmo no ofreció buenos resultados, pero sirvió de guía para que otros autores desarrollarán algoritmos más sofisticados. En particular, años más tarde Osman [47] definió un nuevo algoritmo tabú que volvía a utilizar las heurísticas de mejora. Esta vez, en lugar de recurrir a los intercambios más sencillos, se permiten cualquiera de los intercambios definidos recientemente (intercambio de arcos, cruce de arcos,...). En cada iteración se explora todo el vecindario y se selecciona la mejor solución no tabú. Esta nueva búsqueda tabú sí dio lugar a buenos resultados, aunque todavía era posible mejorar dicho procedimiento.

Durante los posteriores años se han ido desarrollando nuevos procedimientos de búsqueda tabú más sofisticados que han dado lugar a mejores resultados. Entre otros, destacan el algoritmo de Taillard [57] o la búsqueda granular tabú (GFT) de Toth y Vigo [61].

Algoritmos genéticos

Los algoritmos genéticos examinan en cada paso un conjunto de soluciones. Cada conjunto, o población, se deriva del anterior combinando los mejores elementos y descartando los peores. Estos algoritmos están basados en la teoría de la evolución y de la genética: la combinación de los cromosomas (entendidos como parte de una solución) correctos dará lugar a una nueva especie (solución) mejor que la anterior. Esta forma de resolver problemas, que no sólo se emplea para VRP, fue propuesta inicialmente por Holland [31] pero tardó 10 años en ser reconocida por la comunidad científica.

Una forma clásica de utilizar este tipo de algoritmos es tomar dos "padres"y generar dos nuevos "hijos"en cada iteración a través de un esquema de corte de los cromosomas. Un esquema sencillo de corte, denominado corte en un punto, consiste en determinar aleatoriamente un punto de corte y generar un hijo con la primera mitad de los cromosomas de un progenitor y la otra mitad del otro. El problema de este esquema es que no es válido para problemas de rutas, pues

Progenitor 1	:	0	1	2	3	4	5
Progenitor 2	:	**0**	**4**	**3**	**2**	**1**	**5**
Hijo 1	:	0	1	2	**2**	**1**	**5**
Hijo 2	:	**0**	**4**	**3**	3	4	5

Cuadro 1.3: Corte en un punto

Progenitor 1	:	0	1	2	3	4	5
Progenitor 2	:	**0**	**4**	**3**	**2**	**1**	**5**
Hijo 1	:	-	-	-	3	4	-
Hijo 2	:	**0**	**2**	**1**	3	4	**5**

Cuadro 1.4: Orden cruzado

puede darse que los hijos dejen de ser rutas (véase el Cuadro 1.3 donde los padres son dos circuitos que parten del 0 y visitan todos los nodos).

Para el contexto de problemas de rutas se han definido muchos esquemas de corte distintos. En el Cuadro 1.4 se representa un ejemplo de uno de ellos: orden cruzado (Oliver et al. [45]). Este esquema consiste en determinar aleatoriamente dos puntos de corte y asignar al hijo la cadena situada entre esos dos puntos y, para completar la secuencia, tomar el orden en el que aparecen los nodos restantes en el otro progenitor a partir del segundo punto de corte. En el ejemplo, los puntos de corte se encuentran después de la tercera y la quinta posición. El segundo hijo se obtendría intercambiando el papel de los padres.

Optimización de colonias de hormigas

Los sistemas de colonias de hormigas están inspirados en las colonias reales de hormigas. En su búsqueda de comida, las hormigan marcan sus caminos dejando un rastro de feromonas, cuya cantidad depende de la calidad de la comida y de la longitud del recorrido necesario para alcanzar dicha comida. Con el tiempo, los mejores caminos, aquellos más próximos al nido y con mejor comida, son marcados con grandes cantidades de feromonas pues se vuelven los más frecuentados.

Esa observación es la que les llevó a Colorni et al. [11] a proponer una nueva clase de algoritmos basados en la siguiente idea: se pueden crear hormigas artificiales que exploren el conjunto de soluciones y, dependiendo del valor que tome la función objetivo, marquen los valores mediante una función que imite a las feromonas. Para ilustrar los principios básicos de este tipo de heurísticas describiremos brevemente el primer ejemplo (un problema del viajante) al que

fue aplicado.

Con cada arco (i, j) hay asociado un parámetro estático n_{ij} (la inversa de la longitud del arco) denominado visibilidad y un parámetro dinámico, la feromona, Γ_{ij} que varía a lo largo del algoritmo. En cada iteración, las hormigas artificiales, partiendo cada una de un nodo del grafo, trazan n nuevas rutas usando una heurística del vecino más próximo probabilística. El parámetro probabilístico se calcula a partir de n_{ij} y de Γ_{ij} para favorecer la selección de ciudades cercanas y con un alto nivel de feromonas. Al final de cada iteración se recalculan las feromonas del siguiente modo: se permite que una parte $(1 - \rho, 0 < \rho < 1)$ de la feromona se evapore para evitar que malas soluciones encontradas previamente condicionen demasiado la búsqueda en operaciones posteriores; y se añaden nuevas feromonas que dependen del valor de la función objetivo. Así, si el arco (i, j) ha sido utilizado por la hormiga k y ésta ha construido una ruta de longitud L_k, el nuevo rastro de feromónas se verá incrementado en $\Delta_{ij}^k = 1/L_k$. El nuevo rastro de feromonas del arco (i,j) será:

$$\Gamma_{ij} = \rho\Gamma_{ij} + \sum_{k=1}^{N} \Delta_{ij}^k,$$

donde N es el número de hormigas. Este proceso de construcción de rutas se repite durante un número predeterminado de iteraciones.

Para la aplicación de los sistemas de hormigas en el VRP el esquema es similar. Algunos autores como Kawamura et al. [32] o Bullnheimer et al. [4] han desarrollado algoritmos híbridos utilizando los sistemas de colonias de hormigas y dando lugar a muy buenos resultados.

Capítulo 2

Juegos cooperativos

2.1. Introducción

En muchas situaciones reales, cuando varios individuos trabajan juntos se generan costes (o beneficios) que deben ser repartidos entre ellos. Existen multitud de ejemplos de este tipo:

- Las universidades comparten los gastos informáticos entre los distintos departamentos.
- Si varios municipios comparten una depuradora, tendrán que llegar a un acuerdo sobre cómo repartir los costes de su construcción y su mantenimiento.
- Las autoridades aeroportuarias determinan la tasa que los aviones deben pagar en función de su tamaño y del desgaste que producen en la pista.
- Cuando dos médicos comparten una consulta, deben compartir los gastos de material médico, suministros y servicio de secretaria.

En todas las situaciones anteriores surge un gran problema: Cómo repartir de forma "justa"los costes/beneficios. Precisamente eso es lo que trata de estudiar la teoría de juegos cooperativa: la búsqueda de un mecanismo que permita repartir los gastos de manera eficiente, justa y que otorgue incentivos a los agentes para que deseen cooperar.

Existe una gran variedad de soluciones propuestas en la literatura de la teoría cooperativa de juegos. La razón de esta gran variedad de soluciones es la ambigüedad del término justo. Según la situación y los intereses particulares de cada agente (en teoría de juegos los participantes de la cooperación reciben este nombre) el concepto de idoneidad puede variar, por lo que una solución que a priori era buena, puede dejar de serlo si las circunstancias varían. Es importante destacar que en muchos de estos casos, la clave está en que los agentes quieran cooperar, por lo que deben llegar a un acuerdo de reparto que les satisfaga a todos. De esta forma, la teoría de juegos cooperativos se centra en definir propiedades que sería interesante que cumplieran las reglas de reparto y a partir de ahí, encontrar dichas reglas. En este contexto, son más importantes los principios que cumplen las soluciones, que las soluciones en sí mismas o la definición de las soluciones.

Otro de los problemas con los que se enfrenta esta teoría es que varias soluciones pueden cumplir las especificaciones y entonces ¿cómo se decide cuál de ellas es mejor? O, lo que es peor, ¿y si no existe ninguna solución que tenga todas las propiedades deseadas? Claramente, la existencia de una solución "perfecta"está relacionada con el nivel de exigencia que apliquemos a la misma. Se deduce, por tanto, la ardua tarea que es encontrar una forma de reparto de los costes o de los beneficios que satisfaga a todos los agentes.

Ejemplo ilustrativo

A continuación vamos a considerar un ejemplo (extraído de Young([65]) que nos servirá para hacernos una idea de cómo funcionan los problemas de este tipo.

Dos ciudades cercanas consideran construir una depuradora de agua de forma conjunta. La ciudad A podría construirla por 11 millones de euros, mientras que la ciudad B podría hacerlo por 7 millones. No obstante, si cooperasen, podrían construir una única depuradora que les sirviera a los dos a un coste de 15 millones de euros. Obviamente, a ambos les interesaría cooperar pues podrían ahorrarse conjuntamente 3 millones de euros pero esto sólo ocurrirá si llegan a un acuerdo sobre cómo repartir los costes.

Una posible solución consistiría en dividir el coste a partes iguales, esto es, 7.5 millones cada uno. El argumento a favor de esta división sería que ambos tienen el mismo poder a la hora de firmar el contrato. Sin embargo, B nunca estará de acuerdo en repartir así los costes, pues 7.5 millones excede el precio de construir el sistema por sí misma. Además, si las ciudades no tienen el mismo tamaño

(supongamos que A tiene 36000 habitantes y B, 12000) el coste per-cápita de construir la depuradora sería 3 veces superior para los habitantes de B que para los de A. Así, otra forma de repartir los 15 millones de euros podría consistir en hacerlo en función del número de habitantes. El coste per-cápita será, por tanto, de 312.50 €, lo que equivale a 11.25 millones para A y 3.75 para B. Pero esta solución no satisfaría a A, pues implicaría gastar más de lo que haría construyendo la depuradora sin B.

Ninguna de las dos soluciones propuestas anteriormente daría lugar a un acuerdo, por lo que habrá que pensar en alguna otra solución que ofrezca incentivos a los agentes para operar. La forma más simple de asegurar la existencia de incentivos consiste en centrarse en lo que las ciudades pueden ahorrar si cooperan, en lugar de centrarse en los gastos. En línea con las soluciones anteriores, puede plantearse repartir el ahorro (3 millones) a partes iguales entre las ciudades o repartirlo proporcional al número de habitantes. La primera de las opciones daría lugar a un reparto de 9.50 para A y 5.50 para B; mientras que la segunda indicaría 8.75 para A y 6.25 para B. Otra solución que podría ocurrírsenos, es repartir los costes de forma proporcional a los costes individuales. Esta solución coindice además con el reparto de los ahorros proporcional a los costes individuales. En ese caso, A debería abonar 9.17 millones de euros y B, 5.83.

Como se puede observar, cualquiera de las tres últimas soluciones otorga incentivos para cooperar a los dos agentes. De hecho, en ese sentido, cualquier reparto que asignará una cantidad menor o igual que 11 a A y menor o igual que 7 a B sería una buena solución. El conjunto de todas esas soluciones recibe el nombre de núcleo del juego, un concepto que será definido más adelante.

Este ejemplo nos ha permitido comprobar varias cosas. En primer lugar, no existe una respuesta obvia y simple al problema del reparto de costes. Y, en segundo, este problema no puede ser evitado pues, en ocasiones, la no cooperación hace inviable la realización de algún proyecto.

2.2. El modelo cooperativo

En teoría de juegos, un juego cooperativo es un juego en el cual dos o más jugadores no compiten, sino más bien se esfuerzan por conseguir el mismo objetivo. Por el contrario, un juego no cooperativo es uno cuyos jugadores toman decisiones independientemente para su beneficio personal, lo cual no impide que en algunos casos dicha toma de decisiones pueda favorecerlos a todos, como es lo que se busca en los juegos cooperativos. En otras palabras, si los jugadores pueden comunicarse entre ellos, negociar y llegar a acuerdos vinculantes, hablaremos de juegos cooperativos; mientras que si los jugadores no pueden llegar a acuerdos previos, trataremos con juegos no cooperativos. En este trabajo nos centraremos en la primera clase de juegos.

Dentro de los juegos cooperativos existen dos grandes clases: los juegos TU (con utilidad transferible) y los juegos NTU (con utilidad no necesariamente transferible). Cuando los costes o los beneficios pueden repartirse de cualquier modo entre los jugadores nos encontraremos con juegos cooperativos del primer tipo. Por el contrario, si existe alguna restricción sobre la forma de repartir la utilidad, deberemos trabajar con juegos del segundo tipo. A partir de ahora supondremos que no existen restricciones de este tipo, por lo que sólo trabajaremos con juegos cooperativos TU.

Un juego cooperativo TU viene dado por un par (N, v) tal que $N = \{1, 2, ..., n\}$ es un conjunto finito de jugadores (o de proyectos) que pueden trabajar (o realizarse) de manera individual o conjunta; y v es la función característica o de costes[1] ($v : 2^N \to \mathbb{N}$). El coste del jugador cuando actúa de forma individual es $v(i)$, y $v(S), \forall S \subseteq N$, es el coste asociado a la coalición S. Por convención, $v(\emptyset) = 0$.

Como ya se mencionó en la introducción, el principal objetivo de la teoría de juegos cooperativos es encontrar soluciones al problema de repartir los costes o los beneficios. Así, formalmente, un reparto es un vector $(x_1, ..., x_n)$ tal que la componente x_i representa la cantidad que será abonada (o percibida si se trata de beneficios) por el jugador i. De igual modo, una regla de reparto es una función $\varphi(N, v)$ tal que asocia a cada juego un único reparto.

La razón por la que se llevan a cabo los proyectos conjuntamente es que

[1]En los juegos TU, puede trabajarse indistintamente con beneficios o con costes en función de la situación. A partir de ahora, asumiremos que siempre se trabaja con costes.

se generan ahorros. Por lo tanto, en ocasiones, tendrá sentido trabajar con los ahorros que se generan en lugar de con los costes. Para cada subconjunto S, se define el ahorro de S como: $w_v(S) = \sum_{i \in S} v(i) - v(S)$. De esta forma, asociado a cada juego de costes (N, v) se puede definir el coste de ahorros (N, w_v).

Tomemos un nuevo ejemplo que nos servirá para ilustrar las propiedades y soluciones que se van a definir a continuación: Supongase que tres medicos quieren abrir una consulta. Se plantean abrirla de forma conjunta para así ahorrar gastos (podrán realizar los pedidos de material médico conjuntamente consiguiendo así descuentos y podrán compartir los gastos derivados del alquiler, la luz, el servicio de secretaria, etc). Dado que se trata de médicos de distintas especialidades y con distinto número de clientes, sus costes individuales no son los mismos. Se han estimado los gastos mensuales en los que incurrirían si trabajasen individual y conjuntamente y se ha obtenido lo siguiente:

Subconjunto S	Coste $v(S)$		Subconjunto S	Coste $v(S)$
$\emptyset$	0		$\{1,2\}$	6000
$\{1\}$	5000		$\{1,3\}$	10000
$\{2\}$	3000		$\{2,3\}$	7000
$\{3\}$	5000		$\{1,2,3\}$	10500

2.2.1. El núcleo

A continuación se van a definir algunas de las propiedades deseables para un reparto:

Definición 2.1 *Se dice que un reparto $x = (x_1, ..., x_n) \in \mathbb{R}^n$ satisface* ***racionalidad individual*** *si*

$$x_i \leq v(i), \forall i \in N.$$

Definición 2.2 *Un reparto $x = (x_1, ..., x_n) \in \mathbb{R}^n$ será* ***eficiente*** *si*

$$x(N) = x_1 + ..., x_n = v(N).$$

Estas dos propiedades ya fueron definidas, aunque de un modo informal, en la introducción de este capítulo. La primera de ellas hace referencia a que ningún

jugador debe pagar más de su coste individual; y la segunda, a que las soluciones deben repartir el coste de la coalición total.

En teoría cooperativa de juegos existen dos tipos de soluciones: las soluciones tipo conjunto que, como su propio nombre indica, son conjunto de repartos que cumplen unas ciertas características; y las soluciones puntuales, repartos únicos con una serie de propiedades. En ocasiones, es interesante saber si una solución puntual está contenida en otra solución tipo conjunto. Por ello, vamos a comenzar definiendo algunas soluciones tipo conjunto.

Definición 2.3 *Sea (N, v) un juego TU, se define el* ***conjunto de preimputaciones*** *del juego (N, v) como el conjunto de todos los repartos eficientes:*

$$I^*(N,v) = \left\{ x = (x_i)_{i \in N} \in \mathbb{R}^n : \sum_{i \in N} x_i = v(N) \right\}.$$

Definición 2.4 *Sea (N, v) un juego TU, se define el* ***conjunto de imputaciones*** *del juego (N, v) como el conjunto de todos los repartos eficientes que verifiquen además la propiedad de racionalidad individual:*

$$I(N,v) = \{ x = (x_i)_{i \in N} \in I^*(N,v) : x_i \leq v(\{i\}), \forall i \in N \}.$$

Estos dos conjuntos tienes propiedades interesantes pero son conjuntos demasiado grandes, por lo que habrá que definir otras propiedades que nos permitan obtener conjuntos más "pequeños". A pesar es eso, existen ocasiones en las que el conjunto de imputaciones es vacío. Veamos una proposición que nos indica cuándo el conjunto de imputaciones es no vacío:

Definición 2.5 *Un juego (N, v) es* ***esencial*** *si*

$$v(N) \leq \sum_{i=1}^{n} v(i).$$

Proposición 2.1 *Sea (N, v) un juego coalicional, el conjunto de imputaciones es no vacío, $I(N, v) \neq \emptyset$, si, y sólo si, el juego (N, v) es esencial.*

Uno de los conceptos más importantes en la teoría cooperativa de juegos es el **núcleo**, en inglés core, que se representa como $C(N, v)$ y se define como el conjunto de imputaciones que cumplen la propiedad de racionalidad coalicional.

Definición 2.6 *Se dice que un reparto* $x = (x_1, ..., x_n) \in \mathbb{R}^n$ *satisface* ***racionalidad coalicional*** *si*

$$\sum_{i \in S} x_i \leq v(S), \forall S \subset N.$$

La propiedad anterior se refiere a que ningún subconjunto de jugadores deberá pagar más que lo que le correspondería si sólo cooperasen entre ellos.

Definición 2.7 *Dado un juego de costes* (N, v)*, se define el* ***núcleo*** *del juego* $C(N, v)$ *como:*

$$C(N,v) = \left\{ x \in I(N,v) : \sum_{i \in S} x_i \leq v(S), \forall S \subset N, S \neq \emptyset \right\}.$$

Por lo tanto, cualquier reparto que se encuentre en el núcleo ofrecerá incentivos a los jugadores para cooperar. Además, otra propiedad interesante que cumplen los repartos del núcleo es que ningún agente o coalición deberá pagar menos de su aporte marginal al coste total. En términos matemáticos, cualquier reparto $x \in \mathbb{R}^n$ que pertenezca al núcleo cumple que:

$$x(S) \geq v(N) - v(N - S), \forall S \subseteq N.$$

Además, el núcleo es un poliedro convexo y compacto de $\mathbb{R}^n$ con dimensión máxima $n - 1$, pues está contenido en el hiperplano $\sum_{i=1}^{n} x_i = v(N)$.

Ejemplo 2.1 *Calculemos a continuación el conjunto de imputaciones y el núcleo del ejemplo planteado anteriormente.*

$$I(N,v) = \{x = (x_i)_{i \in N} \in I^*(N,v) : x_i \leq v(\{i\}), \forall i \in N\}$$

$$I(N,v) = \left\{(x,y,z) \in \mathbb{R}^3 : x + y + z = 10500; x \leq 5000; y \leq 3000; z \leq 5000\right\}$$

$$C(N,v) = \left\{ x \in I(N,v) : \sum_{i \in S} x_i \leq v(S), \forall S \subset N, S \neq \emptyset \right\}$$

$$C(N,v) =$$
$$\left\{ \begin{array}{l} (x,y,z) \in \mathbb{R}^3 : x + y + z = 10500; 0 \leq x \leq 5000; 0 \leq y \leq 3000; 0 \leq z \leq 5000; \\ x + y \leq 6000; x + z \leq 10000; y + z \leq 7000 \end{array} \right\}$$

$$C(N,v) = \left\{(x,y,z) \in \mathbb{R}^3 : 3500 \leq x \leq 5000; 500 \leq y \leq 3000; 4500 \leq z \leq 5000\right\}$$

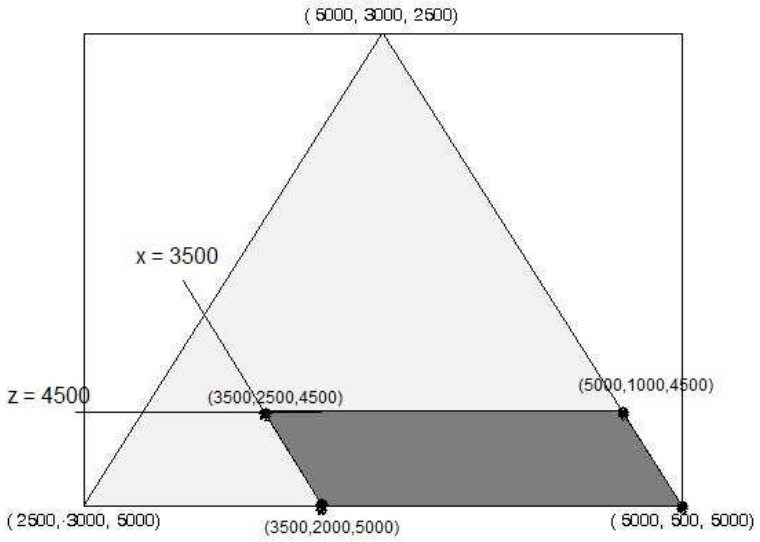


Figura 2.1: Núcleo del ejemplo

En la Figura 2.1 puede observarse una representación gráfica del conjunto de imputaciones (en gris claro) y del núcleo (gris oscuro).

Como se puede observar, en este caso el núcleo es un conjunto relativamente grande pero existen juegos para los que el núcleo es vacío.

Ejemplo 2.2 *Un ejemplo clásico dentro de la teoría de juegos es el juego del dolar: Tres amigos desean repartirse un dolar y sólo podrán hacerlo si al menos dos de ellos se ponen de acuerdo. La función característica será:* $v = (0, 0, 0, 1, 1, 1, 1)$. *(Nótese que se trata de un juego de beneficios, por eso la notación y el concepto de núcleo varían.) En este caso, es imposible encontrar un reparto* (x, y, z) *tal que* $x + y + z = 1$ *y además se cumpla que* $x + y \geq 1$; $x + z \geq 1$ *y* $y + z \geq 1$. *Por tanto, el núcleo de este juego es vacío.*

Una vez comprobada la existencia de juegos con núcleo vacío, será interesante encontrar una condición que nos asegure la existencia del núcleo.

Definición 2.8 *Un juego* (N, v) *es* ***cóncavo*** *si*

$$v(S \cup \{i\}) - v(S) \leq v(T \cup \{i\}) - v(T), \forall S, T : T \subset S \subseteq N \setminus \{i\}\,.$$

De forma intuitiva esta propiedad implica que el coste marginal de incluir un proyecto i decrece cuantos más proyectos haya.

Proposición 2.2 *El núcleo de un juego de costes cóncavo es siempre no vacío. (Shapley(1971), [54])*

2.2.2. El valor de Shapley

El valor de Shapley es una de las soluciones puntuales más famosas dentro de la teoría de juegos cooperativos. Fue definida en 1953 por Shapley en su tesis doctoral [53]. Para ilustrar la idea que se esconde detrás del valor de Shapley vamos a trabajar con un ejemplo:

Ejemplo 2.3 *Tomemos el clásico ejemplo del aeropuerto donde un aeropuerto con una única pista de aterrizaje es utilizado por una serie de aviones de distintos tamaños. La longitud de la pista, y por ende el coste de la misma, viene determinado por el avión de mayor tamaño que la utilice, es decir, cuanto mayor sea el avión, más larga ha de ser la pista. Supongamos entonces que m aviones de distinto tamaño hacen uso de la pista y que dichos aviones están ordenados de la siguiente forma: los de tipo 1 necesitan una pista pequeña, los de tipo 2 una un poco más grande, y así sucesivamente. Podríamos pensar, entonces, en dividir la pista en m segmentos: todos los tipos de aviones harán uso del primer segmento; todos menos el más pequeño utilizarán el segundo, etc.*

Lo justo en este caso sería descomponer los gastos asociados a la pista en los segmentos anteriores y que dichos costes los compartan los aviones que hacen uso de cada segmento. Así, por ejemplo, el tercer avión más pequeño debería abonar $v_1/m + v_2/(m-1) + v_3/(m-2)$.

Esta idea recibe el nombre de principio de descomposición y establece que, en el caso de que la función de costes pueda descomponerse, la solución debe pasar por dividir cada elemento a partes iguales entre los usuarios y después sumar los resultados obtenidos. Además, este principio da lugar a tres ideas: cada agente que use un elemento debe pagar equitativamente por él; aquellos agentes que no hagan uso de un elemento, no deberán costearlo; y por último, los resultados de compartir distintos costes pueden ser agregados.

Veamos, a continuación, cómo estas tres ideas se extienden al caso de funciones de coste que no puedan ser descompuestas de la forma anterior. Para ello, vamos a definir tres propiedades que deberían cumplir los repartos: simetría, jugador nulo y aditividad.

Definición 2.9 *Sea (N, v) un juego de costes. Se dice que el jugador i es* ***nulo*** *si $v(S \cup \{i\}) = v(S), \forall S \subset N \setminus \{i\}$.*

Definición 2.10 *Sean (N, v) un juego de costes e $i, j \in N$ dos jugadores. Los jugadores i y j son* ***simétricos*** *si $\forall S \subset N \setminus \{i, j\}$ se verifica que $v(S \cup \{i\}) - v(S) = v(S \cup \{j\}) - v(S)$.*

Definición 2.11 *La regla de reparto φ satisface* ***jugador nulo*** *si para todo juego (N, v) e i jugador nulo, $\varphi_i(N, v) = 0$.*

Esta propiedad está relacionada con la idea de que ningún jugador debería sufragar los gastos de algo que no utiliza. En particular, si su aportación marginal al coste es nula para cualquier coalición, incluida la total, dicho jugador no debería pagar nada.

Definición 2.12 *La regla de reparto φ satisface* ***simetría*** *si para todo juego (N, v) y todo par i, j de jugadores simétricos, $\varphi_i(N, v) = \varphi_j(N, v)$.*

La propiedad de simetría tiene que ver con el hecho de que todos los usuarios de un mismo elemento deben pagar lo mismo por él. En términos de teoría de juegos, si dos jugadores aportan la misma cantidad al coste de todas las coaliciones, ambos deberían abonar lo mismo.

Definición 2.13 *La regla de reparto φ satisface* ***aditividad*** *si para todo juego (N, v) y (N, w), se verifica que $\varphi(N, v + w) = \varphi(N, v) + \varphi(N, w)$.*

Si una función de costes puede ser descompuesta en otras dos, si aplicamos la regla de reparto sobre ambas funciones y sumamos los resultados, obtendremos lo mismo que si aplicamos directamente la regla de reparto sobre la función de costes original.

Teorema 2.1 *Para todo juego (N, v) dado, existe una única regla de reparto que satisfaga las propiedades de eficiencia, jugador nulo, simetría y aditividad. Dicha regla recibe el nombre de* ***Valor de Shapley*** *[53] y viene dado por la siguiente fórmula:*

$$Sh_i(N, v) = \frac{1}{n!} \sum_{S \cup N \setminus \{i\}} s!(n - s - 1)! \, [v(S \cup \{i\}) - v(s)].$$

Generalmente, el valor de Shapley no se calcula utilizando la expresión anterior, sino que se calcula la siguiente expresión alternativa. Supongamos que los agentes se adhieren a la gran coalición según un orden arbitrario $R = i_1, i_2, ..., i_n$. Podría definirse en ese caso la contribución al coste total del agente $i = i_k$, según el orden R, como:

$$\gamma_i(R) = v(i_1, i_2, ..., i_k) - v(i_1, i_2, ..., i_{k-1}).$$

Una expresión equivalente y más sencilla a la anterior se obtiene calculando la media de $\gamma_i(R)$ en las $n!$ posibles ordenaciones R. Calculemos, a continuación, el valor de Shapley para el ejemplo de los médicos:

Ejemplo 2.4 *Recordemos la función característica de este juego:*

$$v = (5000, 3000, 5000, 6000, 10000, 7000, 10500)$$

R	***J1***	***J2***	***J3***
123	*5000*	*1000*	*4500*
132	*5000*	*500*	*5000*
213	*3000*	*3000*	*4500*
231	*3500*	*3000*	*4000*
312	*5000*	*500*	*5000*
321	*3500*	*2000*	*5000*
suma	*25000*	*10000*	*28000*

Por lo tanto, el valor de Shapley del juego es: $Sh(N, v) = (\frac{25000}{6}, \frac{10000}{6}, \frac{28000}{6})$. *Además, se puede comprobar que dicho reparto pertenece al núcleo del juego:*

$$C(N, v) = \left\{(x, y, z) \in \mathbb{R}^3 : 3500 \leq x \leq 5000; 500 \leq y \leq 3000; 4500 \leq z \leq 5000\right\}$$

pues $3500 \leq \frac{25000}{6} \leq 5000$*;* $500 \leq \frac{10000}{6} \leq 3000$ *y* $4500 \leq \frac{28000}{6} \leq 5000$.

Acabamos de ver un ejemplo en el que el valor de Shapley pertenece al núcleo del juego. No obstante, esto no es siempre así. Supongamos que en el ejemplo anterior los gastos de las coalición total aumentasen hasta 12000. En ese caso, el valor de Shapley del juego sería: $Sh(N, v') = (\frac{28000}{6}, \frac{13000}{6}, \frac{31000}{6})$. Como se puede observar, el valor de Shapley del juego modificado no se encuentra en el núcleo,

pues, $\frac{31000}{6} > 5000$, por lo que dicho reparto no cumple la propiedad de racionalidad individual.

Será interesante encontrar en qué casos esta regla de reparto se encuentra en el núcleo. Shapley [54] demostró el siguiente resultado en 1971.

Teorema 2.2 *El valor de Shapley de cualquier juego (N, v) cóncavo está contenido en el núcleo del mismo.*

2.2.3. El nucleolo

Otra de las soluciones de juegos cooperativos más famosas es el nucleolo (Schmeidler [52]). Ya sabemos que el núcleo es el conjunto de repartos que preservan la cooperación y que el valor de Shapley no siempre está contenido en este conjunto, por lo que estaremos interesados en encontrar una solución alternativa que sí pertenezca al núcleo. El nucleolo selecciona uno de estos repartos siempre que el núcleo sea no vacío.

El nucleolo se define como aquella imputación, por lo que verificará eficiencia y racionalidad individual, que hace que las quejas de las coaliciones sean lo más pequeñas posibles. En general, no hay una fórmula explícita que nos permita calcular el nucleolo, sino que es necesario recurrir a resolver un número finito de problemas de programación lineal. Para la definición formal del nucleolo necesitamos definir dos conceptos previamente.

Definición 2.14 *Dado un juego (N, v), una imputación $x \in I(N, v)$ y un subconjunto de coaliciones $S \subset N$, definimos el* ***exceso de valor*** *de una coalición con respecto a una imputación dada como:*

$$e(S, x) = x(s) - v(S), \forall S \subset N$$

Dicho exceso $e(S, x)$ mide la queja (o infelicidad) de una coalición con respecto a x. Cuanto mayor sea este exceso, más "infeliz"está la coalición S. Obviamente, si $S = N$ o si $S = \emptyset$, $e(S, x) = 0$.

Definición 2.15 *Definimos el orden lexicográfico $\leq_L$ en $\mathbb{R}^n$ de la siguiente manera:*

- *Diremos que* $x <_L y$ *si existe* $k \in \mathbb{N}$, $1 \leq k \leq n$, *tal que:*

$$x_i = y_i, 1 \leq i < k$$

$$x_k < y_k$$

- *Diremos que* $x \leq_L y$ *si* $x = y$ *ó* $x <_L y$.

Dado un reparto $x \in \mathbb{R}^n$, sea $\theta(x)$ la 2^n-tupla cuyas componentes son los excesos, del valor de todas las coaliciones con respecto a x, ordenados de forma no creciente, es decir,

$$\theta_i(x) \geq \theta_j(x) \text{ si } 1 \leq i \leq i \leq 2^n.$$

Ya estamos en condiciones de definir el nucleolo.

Definición 2.16 *Se define el* ***nucleolo*** *del juego* (N, v) *como el conjunto*

$$Nu(N, v) = \{x \in I(N, v) : \theta(x) \leq_L \theta(y), \forall y \in I(N, v)\}.$$

Los elementos del núcleo verifican que $e(S, x) \leq 0, \forall S \subset N$. Por tanto, los vectores de excesos de los repartos del núcleo son lexicográficamente menores que los de cualquier punto de fuera del núcleo. Esto implica que, si el núcleo de un juego es no vacío, entonces el nucleolo está contenido en el núcleo. Nótese que la definición del nucleolo habla de un conjunto, pues es posible que haya varios puntos que cumplan la definición. El siguiente teorema indica en qué casos el nucleolo consiste en un único punto.

Teorema 2.3 *El nucleolo de un juego* ***esencial*** (N, v) *consiste en un único punto.*

Además, se puede demostrar que el nucleolo de un juego satisface las propiedades de jugador títere, jugadores simétricos, covarianza y monotonía coalicional débil.

Definición 2.17 *La regla de reparto* φ *satisface* ***covarianza*** *si dados dos juegos* (N, v) *y* (N, w), $r > 0$ *y* $\alpha = (\alpha_1, ..., \alpha_n) \in \mathbb{R}^n$ *tales que* $w(S) = rv(S) + \sum_{i \in S} \alpha_i, S \subset N$, *se cumple que:*

$$\varphi(N, w) = r\varphi(N, v) + \alpha.$$

Esta propiedad implica que, si la función de coste sufre un cambio de escala y una traslación, la solución de este nuevo problema obtenida a través de la regla es la misma que si calculamos la solución para el problema original y le aplicamos el cambio de escala y la traslación.

Definición 2.18 *La regla de reparto φ satisface* ***monotonía coalicional débil*** *si dados dos juegos (N, v) y (N, w) y $S, T \subset N$ tales que $w(T) > v(T)$ y $w(S) = v(S), \forall S \neq T$, entonces:*

$$\sum_{i \in T} \varphi_i(N, w) \geq \sum_{i \in T} \varphi_i(N, v).$$

Esta última propiedad hace referencia a que, si los costes de una coalición T aumentan y se mantienen constantes los demás, si se aplica la misma regla de reparto que en el caso original, la suma de los pagos asociados a los jugadores de dicha coalición debe ser superior que con la función de coste original.

Nótese la importancia de esta propiedad, pues generalmente los agentes no acuerdan unos pagos en concreto, sino que acuerdan utilizar una regla de reparto para los costes que se puedan generar y es muy frecuente que la función de coste varíe durante el proceso.

2.3. Otras reglas de reparto de costes

En los apartados anteriores se han definido dos de las más importantes reglas de reparto dentro de la teoría cooperativa de juegos (valor de Shapley y nucleolo) y algunas de las propiedades deseables para una regla de reparto, entre las que destaca pertenecer al núcleo del juego. A continuación, se van a definir otras reglas, las cuales, pudiendo no cumplir algunas de las propiedades "deseables", son ampliamente utilizadas debido a la sencillez de su cálculo. Lógicamente, existen muchas otras soluciones que no se van a considerar en este trabajo que son útiles para contextos específicos.

2.3.1. Regla proporcional a los costes individuales

Esta regla es una de las más sencillas dentro de este campo. Consiste en repartir el coste asociado a la coalición total de forma proporcional a los costes individuales. Formalmente, se define como:

$$PROP_i(N,v) = v(N)\frac{v(i)}{\sum_{j\in N} v(j)}.$$

Esta solución cumple las propiedades de eficiencia, siempre se reparte el coste total; racionalidad individual, si el juego es esencial; simetría y jugador nulo en los costes individuales; y monotonía, en el sentido de que si cualquiera de los costes individuales aumenta, también aumentará el coste asociado al agente cuyo coste ha aumentado que le asigne la regla.

El gran inconveniente de esta regla es que puede no estar contenida en el núcleo dado que no se tienen en cuenta los costes de las coaliciones. Calculemos la regla proporcional a los costes individuales para el ejemplo de los médicos.

Ejemplo 2.5 *Recordemos la función característica de este juego:*

$$v = (5000, 3000, 5000, 6000, 10000, 7000, 10500)$$

$$PROP_1(N,v) = 10500\frac{5000}{5000+3000+5000} = 4038{,}46$$

$$PROP_2(N,v) = 10500\frac{3000}{5000+3000+5000} = 2423{,}08$$

$$PROP_3(N,v) = 10500\frac{5000}{5000+3000+5000} = 4038{,}46$$

Como se puede comprobar, este reparto no pertenece al núcleo

$$C(N,v) = \left\{(x,y,z) \in \mathbb{R}^3 : 3500 \leq x \leq 5000; 500 \leq y \leq 3000; 4500 \leq z \leq 5000\right\},$$

pues $PROP_3 = 4038{,}46 < 4500$.

2.3.2. Regla de coste alternativo evitado

Esta regla es más conocida por sus siglas en inglés: *Alternative Cost Avoided (ACA)*, y es el principal método empleado por los ingenieros a la hora de repartir costes asociados a distintos proyectos. La regla ACA asigna los costes según la siguiente fórmula:

$$ACA_i(N,v) = s_i + [r_i/r(N)]\,[v(N) - s(N)]$$

donde $v(i)$ recibe el nombre de costes alternativos, $s_i = v(N) - v(N \setminus i)$ son los costes separables; $r_i = v(i) - s_i$ son los costes alternativos evitados; y $s(N)$ es la suma de los s_i.

Esta regla asigna a cada agente/proyecto su coste separable y la parte proporcional a los r_i de los costes no separables $v(N) - s(N)$. Así definida, la regla puede ser complicada de interpretar pero puede ser definida en términos del juego de ahorros asociado (N, w_v). De este modo, una vez obtenidos los ahorros y_i asignados a cada agente pueden calcularse los costes: $ACA_i(N,v) = v(i) - y_i$. La regla ACA calcula los ahorros según la siguiente fórmula:

$$y_i = \left[w^i(N) / \sum_{j \in N} w^j(N) \right] w(N)$$

donde $w^i(N) = w(N) - w(N \setminus i)$ son los ahorros marginales y $w(N)$ es el ahorro asociado a la coalición total.

Por lo tanto, la regla ACA reparte el ahorro total de forma proporcional a la contribución marginal de cada agente (Straffin y Heaney [55] y Gately [24]). Esta regla ofrece repartos en el núcleo, siempre y cuando éste no sea vacío, para juegos de como máximo 3 jugadores. Calculemos esta regla para nuestro ejemplo:

Ejemplo 2.6 *Sea* $v = (5000, 3000, 5000, 6000, 10000, 7000, 10500)$ *la función característica asociada al juego. Calculemos en primer lugar el juego de ahorros*

asociado, el cual, recordemos, viene dado por la siguiente fórmula:

$$w_v(S) = \sum_{i \in S} v(i) - v(S)$$

Así, el juego de ahorros será: $w_v = (0, 0, 0, 2000, 0, 1000, 2500)$ *y los ahorros marginales:*

$$w^1(N) = w(N) - w(N \setminus \{1\}) = 2500 - 1000 = 1500$$

$$w^2(N) = w(N) - w(N \setminus \{2\}) = 2500 - 0 = 2500$$

$$w^3(N) = w(N) - w(N \setminus \{3\}) = 2500 - 2000 = 500$$

Estamos ahora en condiciones de calcular el reparto según la regla ACA:

$$y_1 = \frac{1500}{1500 + 2500 + 500} 2500 = 833{,}33$$

$$y_2 = \frac{2500}{1500 + 2500 + 500} 2500 = 1388{,}89$$

$$y_1 = \frac{500}{1500 + 2500 + 500} 2500 = 277{,}78$$

por lo que el reparto de costes según este método será:

$$ACA(N, v) = (4166{,}67; 1611{,}11; 4722{,}22).$$

Como ya sabíamos, este reparto pertenece al núcleo:

$$C(N, v) = \left\{(x, y, z) \in \mathbb{R}^3 : 3500 \leq x \leq 5000; 500 \leq y \leq 3000; 4500 \leq z \leq 5000\right\},$$

pues $3500 \leq 4166{,}67 \leq 5000$*;* $500 \leq 1611{,}11 \leq 3000$ *y* $4500 \leq 4722{,}22 \leq 5000$.

2.3.3. Regla igual coste restringido

Como ocurre con la regla anterior, la regla de igual coste restringido es más conocida por sus siglas en inglés: *Constrained Equal Cost (CEC)*. Es una regla muy empleada en la práctica debido a su sencillez. Se define de la siguiente manera:

$$CEC_i(N, v) = min\left\{v(i), \lambda\right\}, \text{ donde } \lambda \text{ es tal que } \sum_{i \in N} CEC_i = v(N)$$

Tal y como ha sido definida, esta regla cumple las propiedades de eficiencia y de racionalidad individual pero puede no pertenecer al núcleo, pues no se tienen en cuenta los costes asociados a las coaliciones. Esta regla trata de que los agentes con costes pequeños asuman su coste y los agentes con mayores costes se repartan equitativamente los gastos restantes. Calculemos esta regla para el ejemplo:

Ejemplo 2.7 *Recordemos que la función característica asociada al problema es* $v = (5000, 3000, 5000, 6000, 10000, 7000, 10500)$. *Luego,*

$$CEC(N, v) = (3750, 3000, 3750)$$

por lo que esta regla no ofrece un reparto en el núcleo para este ejemplo.

Como ocurre con este ejemplo, la regla CEC puede asignar un coste demasiado alto a los agentes con menor coste, haciendo así que a los jugadores con costes mayores se les asignen cantidades inferiores a su aporte marginal a la gran coalición. Además, esta regla no cumple con el principio de monotonía, pues si el coste asociado a los agentes de mayor coste varía, esto no se verá reflejado en la solución. Supongamos que en el ejemplo anterior los costes de $\{1\}$ y de $\{3\}$ aumentan hasta 6000 unidades cada uno. En ese caso, la solución seguiría siendo la misma $CEC(N, v') = (3750, 3000, 3750)$.

2.3.4. Regla igual beneficio restringido

Esta regla también es conocida por sus siglas en inglés: *Constrained Equal Benefits (CEB)*. La idea que subyace en esta regla es que el ahorro de los agentes sea, en la medida de lo posible, el mismo. Viene dado por la siguiente fórmula:

$$CEB_i(N, v) = max\left\{v(i) - \beta, 0\right\}, \text{ donde } \beta \text{ es tal que } \sum_{i \in N} CEB_i = v(N).$$

Esta regla va descontando del coste de cada agente una cantidad de forma igualitaria entre ellos, beneficiando a los jugadores con menor coste. Los repartos derivados de esta regla cumplirán las propiedades de eficiencia y racionalidad individual pero pueden no cumplir racionalidad coalicional y, por tanto, no pertenecer al núcleo. Apliquemos esta regla a los datos del ejemplo:

Ejemplo 2.8 *La función de costes es* $(5000, 3000, 5000, 6000, 10000, 7000, 10500)$. *Por lo que la solución es* $CEB(N, v) = (4166{,}67; 2166{,}66; 4166{,}67)$. *Esta solución tampoco ofrece un reparto en el núcleo para este ejemplo pues,* $4166{,}67 < 4500$.

Aunque pueda parecer que esta regla sí es monótona, existen casos particulares en los que puede aumentar el coste individual de un agente y que no influya en la solución. Veamos un ejemplo:

Ejemplo 2.9 *Supongamos el caso de un juego cooperativo* (N, v) *con* $N = \{1, 2, 3\}$ *y* $v(1) = 20$*,* $v(2) = 80$*,* $v(3) = 150$ *y* $v(N) = 100$*. En este caso, la regla CEB indica repartir los costes de la forma* $(0, 15, 85)$*. Supongamos ahora que el coste de* $\{1\}$ *aumenta hasta 60, la regla indicaría un reparto* $CEB(N, v') = (0, 15, 85)$*, por lo que el gran aumento de* $\{1\}$ *no se vería reflejado en la solución.*

Por último, cabe destacar que tanto la regla CEC como la CEB cumplen la propiedad de simetría en tanto que asignan igual coste a los agentes con el mismo coste individual.

Capítulo 3

Cooperación en TSP y VRP

3.1. Introducción

En los dos capítulos anteriores se han tratado los problemas de rutas sobre nodos y los juegos cooperativos, respectivamente. En este último capítulo se va a tratar de unir las dos cuestiones analizando los denominados juegos del viajante (TSGame) y de rutas de vehículos (VRGame).

En general, y no sólo en los problemas de rutas, la investigación operativa (IO) analiza situaciones en las que un único decisor, guiado por una función objetivo, se enfrenta a un problema de optimización. La teoría se centra entonces en cuestiones sobre cómo actuar de forma óptima. Por otro lado, la teoría cooperativa de juegos analiza situaciones en las que varios decisores deciden trabajar juntos y, por ende, deben repartirse los gastos o los beneficios derivados de esa cooperación, como ya se ha visto en el capítulo anterior. La interrelación entre la investigación operativa y la teoría de juegos cooperativos es un campo de investigación relativamente nuevo y se resume bajo el nombre de "Juegos de Investigación Operativa".

Si se asume que al menos dos jugadores se encuentran o controlan partes (nodos, vértices, etc.) del sistema en estudio, entonces se puede asociar un juego cooperativo a dicho problema de optimización. Si trabajasen de forma conjunta, los agentes podrían conseguir mayores beneficios o menores costes que si trabajasen individualmente. Es en este contexto en el que es interesante plantearse la cuestión de cómo repartir los costes/beneficios.

Existen varias formas de analizar este tipo de situaciones. Una de ellas es

estudiar la propiedades generales (convexidad, equilibrio, ...) de los juegos obtenidos a partir del problema de IO correspondiente y aplicar soluciones apropiadas desarrolladas por la teoría cooperativa de juegos (núcleo, valor de Shapley, nucleolo, etc.). La otra forma de abordar estos problemas es crear nuevas reglas de reparto específicas para cada situación que satisfagan propiedades interesantes dado el contexto. En ocasiones, dichas reglas de reparto se desarrollan a partir de los algoritmos de resolución existentes para el problema en cuestión.

A continuación, nos vamos a centrar en el análisis de los juegos asociados a los problemas del viajante y de rutas de vehículos.[1]

[1]Para unificar la notación en este capítulo, y para simplificar la definición de los juegos TU, se va a denotar como nodo 0, tanto el origen del viajante, como el depósito de los problemas de rutas.

3.2. Juego del viajante

Existen varias versiones del juego del viajante de comercio, aunque las más importantes se reducen a dos: el denominado juego del viajante con ruta fija y el juego del viajante de comercio (TSG). El primero de estos juegos tiene buenas propiedades, como que su núcleo es siempre no vacío y requiere "poco"tiempo computacional; mientras que el segundo resuelve algunos de los inconvenientes del primero pero puede tener núcleo vacío para juegos de más de 5 jugadores. La literatura sobre juegos asociados al problema del viajante es relativamente escasa. Destacamos varios artículos sobre los que está basada esta sección: Fishburn y Pollak (1983) [19], Tamir (1988) [58], Potters et al.(1992) [49], Kuipers (1993) [34] y Borm et al. (2001) [3].

Ambos juegos responden al mismo esquema aunque se diferencian en la forma en que definen la función de costes para las coaliciones intermedias. Veamos un ejemplo del tipo de problema que puede ser modelizado a través de esta clase de juegos. Supongamos que un ponente ha sido invitado por varias universidades para que dé una conferencia en cada una de ellas. Dicho conferenciante puede visitar cada una de las universidades y regresar a su ciudad de origen, o bien, partiendo de su ciudad de origen, realizar un único viaje visitando cada una de las ciudades y regresando. Esta última opción tendrá asociado un menor coste para el conjunto de las universidades, por lo que será preferible a la primera. En ese caso, además, el coste total del viaje debe ser repartido entre las distintas universidades, por lo que el problema se resume en encontrar un reparto "justo"de los gastos del viajante.

3.2.1. El juego del viajante con ruta fija

Este juego fue definido por Fishburn y Pollak [19]. El problema que estudiaron es el siguiente: una persona, financiada por varios patrocinadores, visita, partiendo de su ciudad, las ciudades de los patrocinadores y regresa al origen. El coste total del viaje debe ser compartido por los patrocinadores. Como se puede comprobar, este ejemplo coincide, a grandes rasgos, con el definido anteriormente.

Los autores propusieron tres condiciones que, según su criterio, debía cumplir cualquier regla de reparto:

(a) La suma de los pagos de cada patrocinador debe coincidir con el coste total del viaje (eficiencia).

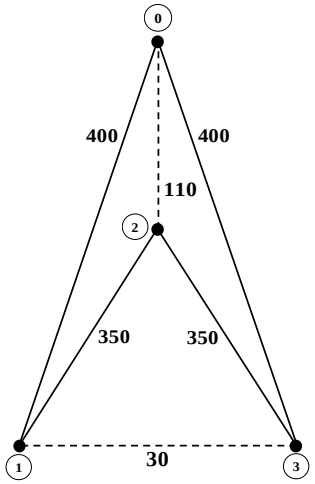


Figura 3.1: Ejemplo utilizado en [19]

(b) Ningún patrocinador pagará más de lo que cueste un viaje directo desde el origen del viajante hasta la ciudad del patrocinador y vuelta al origen (racionalidad individual).

(c) Si un patrocinador i tiene un coste marginal superior al de otro patrocinador j, entonces la contribución de i ha de ser superior que la de j.

En este contexto, el coste marginal de i se define como la diferencia entre el coste total y el coste del circuito obtenido al eliminar i del circuito pero manteniendo el orden del resto de ciudades. Inicialmente, Fishburn y Pollak pensaron en tomar el coste marginal como una cota mínima de la contribución pero, dado que la suma de dichos costes puede ser superior al coste total en el caso de que la ruta fijada no sea de mínimo coste, adoptaron el principio (c).

El problema de estas tres condiciones es que, en general, no pueden darse al mismo tiempo. Para ilustrar este resultado, Fishburn y Pollak tomaron la ruta $0-1-2-3-0$ del grafo de la Figura 3.1 cuyo coste total es 1500 y los costes marginales, $(640, 670, 640)$. Por (c), los agentes 1 y 3 deberían pagar lo mismo, y 2 debería pagar más que ellos. Pero por (b), 2 no estará dispuesto a pagar más de 220. Luego, lo máximo que pagará 2 será 220 y, dado que 1 y 3 no pueden pagar más que 2, será imposible alcanzar el coste total de 1500 por lo que se concluye que las tres condiciones son incompatibles.

Otra de las curiosidades de [19] es que los autores trataron de encontrar una regla de reparto para cualquier ruta y no sólo para las rutas de mínimo coste, como se deduce del ejemplo anterior. Lógicamente, en el caso de rutas no mínimas,

alguno de los patrocinadores podría negarse a pagar el coste extra de no realizar la ruta óptima y la cooperación podría no tener lugar. Esta observación fue realizada por Potters et al. [49] que decidieron redefinir el juego del viajante con ruta fija en términos de una ruta óptima.

Potters et al. reconsideraron el ejemplo propuesto por Fishburn y Pollak (Figura 3.1) y determinaron la ruta de mínimo coste: $0-2-1-3-0$ con coste 890 y costes marginales $(30, 60, 30)$. En este caso, (a), (b) y (c) continúan sin ser compatibles pero la suma de los costes marginales ya no excede el coste total (esto es así siempre que la ruta prefijada sea de mínimo coste y las distancias cumplan la desigualdad triangular) por lo que es posible replantearse tomar dichos costes marginales como una contribución mínima de cada agente. De este modo, en [49] se estudia la posibilidad de encontrar alguna regla de reparto que cumpla (a) y (b) y una nueva condición:

(c′) Cada patrocinador ha de pagar, como mínimo, su coste marginal.

Definamos formalmente el juego del viajante con ruta fija:

Sea $N_0 = \{0, 1, 2, ..., n\}$ el conjunto de ciudades a visitar y c una matriz $(n+1) \times (n+1)$, donde c_{ij} denota el coste asociado a viajar de i a j. Además, c cumple que:

$$c_{ii} = 0, \forall i \in N_0 \tag{3.1}$$

$$c_{ij} + c_{ik} \geq c_{ik}, \forall i, j, k \in N_0. \tag{3.2}$$

Las rutas del viajante con ruta fija vendrán determinadas por una permutación circular σ de los nodos de N_0 y $\sigma(i)$ será la ciudad visitada inmediatamente después de i. Además, $\sigma_{|S_0}$ serán las rutas que visiten sólo los elementos de S y se obtendrán eliminando de σ las ciudades de $N \setminus S$ y manteniendo el orden del resto de ciudades. Así, podemos definir el juego del viajante con ruta fija como el juego cooperativo TU $(N, v_{c,\sigma})$ con función de costes $v_{c,\sigma}$ definida como:

$$v_{c,\sigma}(S) = \sum_{i \in S_0} c_{i,\sigma_{|S_0}(i)}, \quad \forall S \subseteq N.$$

Recordemos que la idea de Potters et al. era encontrar un reparto que satisficiera (a), (b) y (c′) al mismo tiempo. Obsérvese que cualquier reparto que cumpla estas condiciones puede no pertenecer al núcleo del juego si $n \geq 4$, aunque el recíproco siempre es cierto. En [49] se demuestra el siguiente teorema de gran importancia:

Teorema 3.1 *Si la matriz c cumple las propiedades (3.1) y (3.2) y σ es una permutación circular de N_0 tal que*

$$\sum_{i\in N_0} c_{i,\sigma(i)} = min\left\{\sum_{i\in N_0} c_{i,\tau(i)}/\tau \text{ una permutación circular de } N_0\right\},$$

entonces el juego $(N, v_{c,\sigma})$ tiene núcleo no vacío.

Cabe destacar la importancia de este resultado pues implica que, así definido, los agentes del juego siempre tienen incentivos para cooperar. Además, no ha sido necesario exigir simetría en la matriz c por lo que en el caso de TSP asimétricos, este resultado sigue siendo válido.

Por otro lado, si la matriz c cumple las propiedades (3.1) y (3.2), Potters et al. propusieron la siguiente regla de reparto que generalmente no está en el núcleo pero que siempre cumple las condiciones (a), (b) y (c′):

$$\phi_i = \lambda v_{c,\sigma}(i) + (1-\lambda)(v_{c,\sigma}(N) - v_{c,\sigma}(N \setminus i)$$

donde λ se elige tal que $\sum_{i\in N} \phi_i = v_{c,\sigma}(N)$.

Calculemos el núcleo y el reparto según esta regla para el ejemplo propuesto en [19], reflejado en la Figura 3.1:

Ejemplo 3.1 *En primer lugar, recordemos que la ruta óptima, la solución al TSP, es $\sigma = (0,2,1,3,0)$ con coste 890. Así, el juego del viajante con ruta fija tendrá la siguiente función de costes: $v_{c,\sigma} = (800, 220, 800, 860, 830, 860, 890)$ y el núcleo vendrá dado por:*

$$C(N, v_{c,\sigma}) = \left\{(x,y,z) \in \mathbb{R}^3 / 30 \le x \le 800; 60 \le y \le 220; 30 \le z \le 800\right\}.$$

Además, para obtener la solución según la regla se debe resolver el siguiente sistema de ecuaciones lineales:

$$\begin{cases} x = \lambda 800 + (1-\lambda)30 \\ y = \lambda 220 + (1-\lambda)60 \\ z = \lambda 800 + (1-\lambda)30 \\ x + y + z = 890 \end{cases}$$

Por lo tanto, $\phi = (378{,}76; 132{,}48; 378{,}76)$, que está contenido en el núcleo y, por ende, cumple las condiciones (a), (b) y (c′).

Como se ha visto, este juego tiene buenas propiedades: su cálculo es relativamente sencillo y siempre tiene núcleo no vacío. Además, es posible calcular el valor del nucleolo en un tiempo exponencial $O(n^4)$. En Kuipers et al. (2000) [35] se muestra un algoritmo que calcula, en dicho tiempo, el nucleolo de un juego del viajante con ruta fija.

3.2.2. El juego del viajante de comercio

En el capítulo anterior se ha desarrollado el juego del viajante con ruta fija, el cual, a pesar de tener buenas propiedades, tiene un gran defecto: a cada coalición le asigna el coste de realizar un circuito que puede no ser óptimo. Lógicamente, si eliminamos de la permutación circular óptima algunos elementos no tiene por qué obtenerse una subruta óptima. En este sentido, podría plantearse encontrar un reparto eficiente (a), que además cumpla:

(b′) Ninguna coalición $S \subset N$, incluidas las unipersonales, pagará más del coste asociado a la ruta óptima entre las ciudades de S.

En otras palabras, ¿es el núcleo del juego del viajante (N, v_c) definido por:

$$v_c(S) = min \left\{ \sum_{i \in S_0} c_{i,\sigma(i)} / \sigma \text{ es una permutación circular de } S_0 \right\} \tag{3.3}$$

vacío o no?

El juego recién definido recibe el nombre de juego del viajante de comercio y ha sido objeto de estudio de varias publicaciones. Nótese que este juego tiene un gran inconveniente: para poderlo definir es necesario resolver un TSP para cada coalición $S \subseteq N$. Recordemos que el TSP es un problema NP-duro, por lo que el hecho de que sea necesario resolver un número tan elevado de estancias del TSP para definir v_c hace prácticamente imposible trabajar con problemas de este tipo con un gran número de agentes.

A la vista del Teorema 3.1, se podría pensar que para el juego definido por (3.3) es suficiente con que se cumplan las condiciones (3.1) y (3.2) para que el núcleo del mismo sea no vacío, pero esto no es así. Potters et al. [49] demostraron que cualquier juego de ese tipo con $n \leq 3$ y cuya matriz c cumpla las condiciones (3.1) y (3.2) tiene núcleo no vacío. Además, para $n = 4$ mostraron un contraejemplo de un juego con matriz c asimétrica que cumplía (3.1) y (3.2) pero que poseía núcleo vacío. El grafo asociado a dicho ejemplo está recogido en la Figura 3.2.

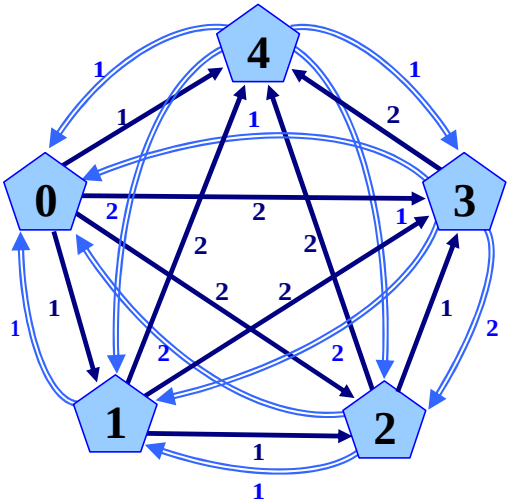


Figura 3.2: Ejemplo de TSG con 4 jugadores y núcleo vacío

Ejemplo 3.2 *Tómese el grafo de la Figura 3.2. El circuito hamiltoniano de coste mínimo que visita las 4 ciudades es* $0-1-2-3-4-0$ *y tiene un coste de* $v_c(N)=6$. *Además,* $v_c(\{1,2,3\})=4=c_{0,1}+c_{1,2}+c_{2,3}+c_{3,0}$, $v_c(\{1,2,4\})=4=c_{0,4}+c_{4,2}+c_{2,1}+c_{1,0}$ *y* $v_c(\{3,4\})=3=c_{0,4}+c_{4,3}+c_{3,0}$. *Si* $x=(x_1,x_2,x_3,x_4)$ *fuera un reparto del núcleo, entonces*

$$6=x_1+x_2+x_3+x_4=\frac{1}{2}(x_1+x_2+x_3)+\frac{1}{2}(x_1+x_2+x_4)+\frac{1}{2}(x_3+x_4)\leq\frac{1}{2}(4+4+3),$$

lo que es claramente una contradicción. Por lo que el núcleo es vacío.

Se acaba de mostrar un ejemplo de un juego de 4 jugadores cuya matriz cumple (3.1) y (3.2) pero que posee un núcleo vacío. Por tanto, queda demostrado que, para juegos de más de 3 jugadores, no es suficiente con cumplir (3.1) y (3.2) para tener núcleo no vacío.

En Tamir (1989) [58] se demuestra que los juegos del viajante de comercio de 4 jugadores con matriz de costes c simétrica que cumpla la desigualdad triangular (3.2) también tienen núcleo no vacío. Para dicha demostración, recurre a un resultado de Fonlupt y Naddef [23], según el cual un TSP dado por un grafo G producirá el mismo valor para la coalición total en el juego v_c definido anteriormente que en el juego definido por $\bar{v}$ si y sólo si dicho grafo G no contiene ningún menor[2] que sea isomorfo a los grafos incluidos en la Figura 3.3. El juego $\bar{v}$ viene

[2]En teoría de grafos, un grafo H es un menor de G si puede ser obtenido a partir de la eliminación y contracción de algunos de los arcos y nodos de G.

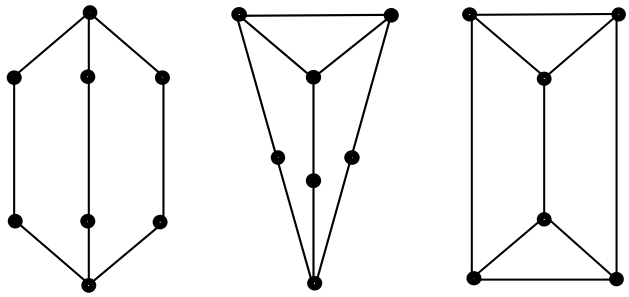


Figura 3.3: Grafos destacados en [23]

dado por la siguiente fórmula[3]:

$$\bar{v}(S) = min \sum_{e \in E} c_e x_e$$

$$\text{sujeto a} : \sum_{e \in \delta(T)} x_e \geq 2, \quad \forall T \subseteq S$$

$$x_e \geq 0, \quad \forall e \in E$$

La importancia de este resultado se debe a que si $v_c(N) = \bar{v}(N)$, entonces se puede demostrar que ambos juegos tienen núcleo no vacío [48]. Por lo tanto, para que el juego del viajante de comercio tenga núcleo no vacío bastará con que el grafo que lo defina no tenga ningún menor isomorfo a los de la Figura 3.3.

En [58] se demuestra que los juegos del viajante de comercio con 4 jugadores y con matriz c simétrica que cumpla la desigualdad triangular tienen núcleo no vacío. La demostración se basa en la idea de que, al contar con 5 únicos nodos, el grafo que define todos esos problemas nunca puede contener un menor isomorfo a los grafos "prohibidos", pues todos poseen un mínimo de 6 nodos. En la misma publicación, Tamir muestra un ejemplo de 6 jugadores con núcleo vacío, por lo que demuestra así también que los TSG con 6 ó más jugadores pueden tener núcleo vacío.

Ejemplo 3.3 *Tomemos el ejemplo utilizado por Tamir para demostrar que los TSG con 6 jugadores no siempre tienen núcleo no vacío. El grafo del ejemplo está plasmado en la Figura 3.4. En este ejemplo, el coste asociado a los arcos es 1 para los arcos del grafo y el coste del camino mínimo entre los extremos para el resto.*

[3]Se emplea la misma notación que en el Capítulo 1 de este trabajo.

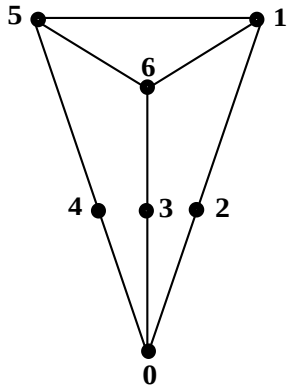


Figura 3.4: Contraejemplo utilizado en [58]

Así, el camino mínimo que visita todos los clientes es $(0, 4, 5, 6, 1, 2, 3, 0)$ *con un coste* $v_c(N) = 8$. *Además,* $v_c(\{1, 2, 4, 5\}) = v_c(\{3, 4, 5, 6\}) = v_c(\{1, 2, 3, 6\}) = 5$. *Supongamos que el reparto* $x \in \mathbb{R}^6$ *pertenece al núcleo del juego, entonces*

$$16 = 2v_c(N) = x(1, 2, 4, 5) + x(3, 4, 5, 6) + x(1, 2, 3, 6) \leq 5 + 5 + 5 = 15$$

es una contradicción y, por lo tanto, el núcleo del juego es vacío.

Hasta el momento se ha demostrado que los juegos del viajante de comercio con un número de jugadores menor o igual que 4, tienen siempre núcleo no vacío. Además, se ha comprobado que esto no es así para juegos de 6 ó más jugadores. No obstante, no se ha dicho nada acerca de los juegos de 5 jugadores. En [34], J. Kuipers demuestra que los juegos del viajante de comercio con 5 jugadores también tienen núcleo no vacío.

Aunque pueda parecer lo contrario, los resultados anteriores no implican que no haya familias particulares del TSG para las que se pueda asegurar la existencia de núcleo no vacío. En Okamoto (2004) [44], se demuestra la existencia de núcleo no vacío para dos juegos del viajante de comercio con matrices de distancias "especiales ". En particular, se demuestra que los TSG con matriz de distancias que cumplen la propiedad de Monge tienen el núcleo no vacío, y que aquellos cuyas matrices son de Kalmanson son cóncavos. Además, cabe destacar que los TSP con matrices de esas características son problemas del viajante que pueden ser resueltos en tiempo polinómico. Definamos, a continuación, la propiedad de Monge y las matrices Kalmanson.

Definición 3.1 *Diremos que una matriz* $c = [c_{ij}]_{N_0 \times N_0}$ *cumple la* ***propiedad de***

***Monge** si:*

$$c_{i,k} + c_{j,l} \leq c_{i,l} + c_{j,k}, \quad \forall i < j, k < l.$$

Como se puede deducir de la definición, que una matriz cumpla o no dicha propiedad depende en gran medida del orden de los índices de la matriz. Para resolver este problema, definiremos la matriz de Monge permutada. Una matriz será de Monge permutada si existe una permutación σ de sus índices tal que la matriz cuyas componente i, j son $\sigma(i)$ y $\sigma(j)$ cumple la propiedad de Monge.

Proposición 3.1 *El núcleo del juego de viajante de comercio (N, v_c) es no vacío si la matriz c de costes cumple la propiedad de Monge (o si es una matriz permutada de Monge).*

Los juegos del viajante de comercio (N, v_c) con matriz de costes simétrica y permutada de Monge son totalmente balanceados, esto es, todos sus subjuegos tienen núcleo no vacío.

Definición 3.2 *Diremos que una matriz $c = [c_{ij}]_{N_0 \times N_0}$ es una matriz de Kalmanson si es simétrica y para todo $i < j < k < l$ cumple que:*

$$c_{i,j} + c_{k,l} \leq c_{i,k} + c_{j,l}$$

$$c_{i,l} + c_{j,k} \leq c_{i,k} + c_{j,l}.$$

Análogamente a las matrices de Monge, se pueden definir las matrices permutadas de Kalmanson.

Proposición 3.2 *El núcleo del juego de viajante de comercio (N, v_c) es cóncavo si la matriz c de costes es una matriz permutada de Kalmanson.*

Esta última proposición implica que los juegos con ese tipo de matriz tendrán núcleo no vacío y el valor de Shapley del juego será un reparto del núcleo. Es importante destacar que estas dos condiciones son suficientes pero no necesarias para que el núcleo sea no vacío, esto es, existen juegos que no cumplen ninguna de las dos propiedades y tienen núcleo no vacío.

Otro resultado interesante asociado a los juegos con matrices de Kalmanson que se muestra en [44], es que las rutas óptimas para la gran coalición son rutas maestras, si la matriz c es además simétrica. Las rutas maestras son rutas en las que, para cada coalición $S \subseteq N$, el circuito de mínimo coste se obtiene eliminando de la secuencia las ciudades que no estén en S.

Por tanto, si la matriz de costes es simétrica y de Kalmanson, el juego del viajante de comercio coincidirá con el juego del viajante con ruta fija. Nótese que en [49] ya se demostró que estos juegos tienen núcleo no vacío.

3.2.3. Líneas de trabajo futuras

Hasta el momento se han definido dos tipos distintos de juegos asociados al TSP, pero prácticamente no se han desarrollado ni estudiado reglas de reparto para ellos. Es posible que esto se deba a que no está asegurado que el núcleo sea no vacío en muchos de los casos. Generalmente, al plantear una regla de reparto, lo que se busca es que siempre proporcione soluciones en el núcleo.

Por tanto, una línea de trabajo que podría explorarse es estudiar cómo funcionarían las reglas definidas en el capítulo 2 (u otras reglas nuevas que se definan *ex profeso*) en los casos particulares en los que sí se ha asegurado que el núcleo es no vacío (juego del viajante con ruta fija, juegos del viajante de comercio con matrices de tipo Monge o Kalmanson).

Otra opción sería intentar plantear nuevas fórmulas de reparto para los juegos con menos de 6 jugadores que posean buenas propiedades y extenderlas para juegos con un mayor número de agentes.

3.3. Juego de rutas de vehículos

El juego de rutas de vehículos se define como el problema de repartir los gastos derivados de un VRP entre los clientes o ciudades que se visitan. Recordemos que un VRP es un problema que consiste en encontrar una configuración de rutas de mínimo coste tales que se visite un conjunto de clientes una única vez y se satisfagan una serie de restricciones adicionales. Dado que que se busca la ruta de mínimo coste, a los distintos clientes les interesará cooperar y ser servidos de forma conjunta, pues así conseguirán reducir costes. El problema es, de nuevo, decidir cómo repartir dichos costes. En esta sección se va a estudiar este problema desde el punto de vista de la teoría de juegos.

Como ocurre con los juegos del viajante, la literatura sobre el juego de rutas de vehículos es relativamente escasa. No obstante, hay un artículo muy interesante de Göthe-Lundgren et al. [27] en el que nos vamos a centrar. En dicha publicación, se presenta lo que ellos denominan el juego básico de rutas de vehículos, se expone una condición suficiente para que dicho juego tenga núcleo no vacío y se propone un método para calcular el nucleolo de dicho juego. En un artículo posterior de Chardaire [5] se critica el método propuesto en [27], al no ser válido para juegos con núcleo vacío, y se muestran varios ejemplos que lo confirman. También destacamos un artículo reciente de Tae et al. [56], en el que se estudia el juego de problemas de rutas de vehículos con ventanas de tiempo y se propone un algoritmo para calcular el nucleolo, basado en el método expuesto en [27].

El juego en el que se centran en [27] es lo que nosotros hemos denominado CVRP, esto es, un VRP en el que las demandas (d_i) son conocidas y existe una flota con un número m de vehículos, todos con capacidad C. Entonces, el juego (N, v_c^m) se define sobre el conjunto N de clientes a visitar, siendo $v_c^m(S)$ el coste asociado a las rutas óptimas que surten a todos los agentes de S. Definamos formalmente este problema tal y como se hace en [27].

Supóngase que para todo subconjunto de clientes tales que su demanda total no exceda C, se conoce el coste de la ruta óptima. Denotemos por c_r dicho coste y por R, el conjunto de todas las rutas de mínimo coste. Además, la variable dicotómica a_{ir} tomará el valor 1 si el cliente i es visitado por la ruta r y 0, en otro caso; y x_r valdrá 1, si la ruta r es elegida en la ruta óptima, y 0, en otro caso. Por último, se define una coalición $S \subseteq N$ como el vector de componentes

binarias siguiente:

$$\forall i \in N, \quad s_i = \begin{cases} 1, \text{ si el agente i pertenece a la coalición} \\ 0, \text{ en otro caso} \end{cases}$$

Así, $\forall S \subseteq N$, podemos definir $v_c^m(S)$ a partir del siguiente problema de programación lineal. Para cualquier coalición $S \subseteq N$, $S \neq \emptyset$,

$$v_c^m(S) = \quad min \quad \sum_{r \in R} c_r x_r$$

$$\text{sujeto a } \sum_{r \in R} a_{ir} x_r = s_i, \quad \forall i \in N$$

$$x_r \geq 0, \quad \forall r \in R$$

$$x_r \text{ entero}, \forall r \in R.$$

Si la matriz de distancias c asociada al VRP cumple la desigualdad triangular, entonces el juego así definido es monótono ($v_c^m(S) \leq v_c^m(T), \quad \forall S \subset T \subseteq N$) y subaditivo ($v_c^m(S) + v_c^m(T) \geq v_c^m(S \cup T), \quad \forall S, T \subseteq N, S \cap T = \emptyset$), pero no siempre tiene núcleo no vacío. Veamos un ejemplo (extraído de [27]), que así lo demuestra.

Ejemplo 3.4 *En la Figura 3.5 se muestra la ubicación de los clientes y del depósito así como los costes de transporte. Supongamos que cada cliente tiene una demanda de una unidad y que existen tres vehículos disponibles, cada uno con una capacidad de dos unidades. Teniendo en cuenta la definición del juego anterior, podemos calcular la función característica del mismo:* $v_c^m = (2; 2; 2; 3{,}7; 3{,}7; 3{,}7; 5{,}7)$. *Supongamos que* $x \in \mathbb{R}^3$ *es un reparto en el núcleo. En ese caso,*

$$5{,}7 = v_c^m(N) = \sum_{i \in N} x_i = \frac{1}{2}(x_1{+}x_2){+}\frac{1}{2}(x_1{+}x_3){+}\frac{1}{2}(x_2{+}x_3) \leq \frac{1}{2}(3{,}7{+}3{,}7{+}3{,}7) = 5{,}55$$

lo que es una contradicción, por lo que el núcleo es vacío.

Acabamos de ver un ejemplo de un juego con núcleo vacío, pero esto no siempre es así. Modificando ligeramente el ejemplo anterior se obtiene un juego con núcleo no vacío.

Ejemplo 3.5 *Tomemos el ejemplo anterior, pero en este caso el cliente 1 tiene una demanda de dos unidades. En dicho caso, la función característica será* $v_c^m =$

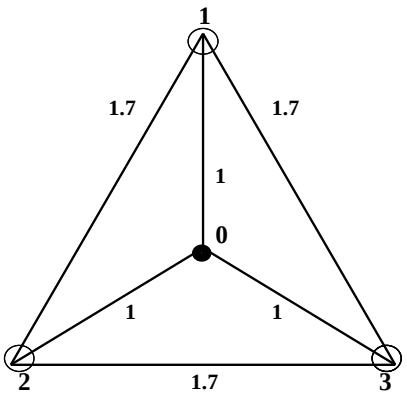


Figura 3.5: Grafo ejemplos 3.4 y 3.5

$(2;2;2;4;4;3{,}7;5{,}7)$ *y el núcleo será no vacío y vendrá dado por:*

$$C(N, v_c^m) = \left\{(x, y, z) \in \mathbb{R}^3 : x = 2, y + z = 3{,}7; \quad 0 \leq y, z \leq 2\right\}.$$

De este último ejemplo se deduce una propiedad interesante que cumplen los juegos de rutas de vehículos. Pero antes de centrarnos en dicha propiedad, debemos definir un nuevo concepto.

Definición 3.3 *Diremos que una coalición S es una coalición* ***factible*** *si:*

$$\sum_{i \in S} d_i \leq C.$$

Además, denotaremos por $\mathcal{S}$ el conjunto de todas las coaliciones factibles.

En otras palabras, una coalición es factible si los agentes que la conforman pueden ser servidos por un único vehículo. Nótese que, para estas coaliciones, el VRP que debe resolverse para calcular $v_c^m(S)$ se reduce a un TSP.

La propiedad que antes mencionamos se traduce en la siguiente proposición:

Proposición 3.3 *Consideremos una ruta óptima sobre N y supongamos que puede descomponerse en m rutas, que cubran cada una los clientes de las coaliciones factibles disjuntas $S_1, S_2, ..., S_m$. Entonces, cualquier reparto $x \in \mathbb{R}^n$ que pertenezca al núcleo debe cumplir:*

$$\sum_{j \in S_r} x_j = v_c^m(S_r), \quad \forall 1 \leq r \leq m.$$

Demostración. Si la ruta óptima con coste $v_c^m(N)$ puede descomponerse, entonces se tiene que $v_c^m(N) = \sum_{r=1}^{m} v_c^m(S_r)$ y que $\sum_{r=1}^{m} \sum_{j \in S_r} x_j = \sum_{j \in N} x_j$. Cualquier reparto x del núcleo debe cumplir que $\sum_{j \in N} x_j = v_c^m(N)$ y que $\sum_{j \in S_r} x_j \leq v_c^m(S_r), \quad \forall 1 \leq r \leq m$. Así:

$$v_c^m(N) = \sum_{j \in N} x_j = \sum_{r=1}^{m} \sum_{j \in S_r} x_j \leq \sum_{r=1}^{m} v_c^m(S_r) = v_c^m(N),$$

por lo que todas las desigualdades deben ser igualdades y, en particular,

$$\sum_{j \in S_r} x_j = v_c^m(S_r), \forall x \in C, 1 \leq r \leq m.$$

■

Esta proposición implica que, mientras el núcleo sea no vacío, el coste de una ruta incluida en la solución del VRP para la coalición total debe ser compartido sólo por los clientes cubiertos por esa ruta.

En el Ejemplo 3.5 se tiene que el núcleo está compuesto por todos los repartos tales que $x = 2$ y $y + z = 3{,}7$. Se puede comprobar que las dos únicas coaliciones factibles del problema son $\{1\}$ y $\{2, 3\}$ y que la ruta óptima se compone de las subrutas $(0 - 1 - 0)$ y $(0 - 2 - 3 - 0)$, por lo que la primera subruta debería costearla el cliente 1 y la segunda, los clientes 2 y 3, como así resulta en el núcleo.

Otro resultado interesante que mostraron Göthe-Lundgren et al. es una condición de suficiencia para que el núcleo sea no vacío.

Proposición 3.4 *Sea $\bar{z}$ el valor óptimo de la función objetivo de la relajación lineal del VRP asociado a la coalición total y $z = v_c^m(N)$. El núcleo del juego de rutas de vehículos es no vacío si y sólo si $\bar{z} = z$.*

Este resultado es de gran utilidad, pues nos permite saber de manera más o menos sencilla si un juego tiene o no núcleo vacío, ya que es más fácil resolver la relajación lineal del VRP que el VRP. En el caso de que dicho núcleo sea no vacío, gracias a la Proposición 3.3 puede resultarnos más sencillo calcularlo. No obstante, en [27] se presenta otra propiedad interesante del núcleo de estos juegos que facilita enormemente el cálculo de éste.

Recordemos que el núcleo viene dado por los repartos x que satisfacen:

$$\sum_{i \in N} x_i = v_c^m(N) \tag{3.4}$$

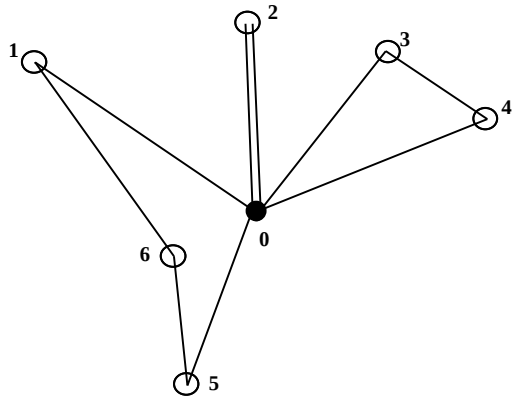


Figura 3.6: Ejemplo de VRG

$$\sum_{i \in S} x_i \leq v_c^m(S), \quad \forall S \subset N. \tag{3.5}$$

Si denominamos CDI (Core Defining Inequality) a las restricciones dadas por (3.5), podemos enunciar la siguiente proposición:

Proposición 3.5 *Cualquier CDI asociada a una coalición no factible S, $S \neq N$, es redundante.*

Demostración. Consideremos una coalición no factible $\hat{S}$, $\hat{S} \neq N$, y una ruta óptima con subrutas $\{r_1, ..., r_m\}$, cada una de ellas asociada a una coalición factible disjunta $\{S_1, ..., S_m\}$. Dado que $\sum_{j=1}^{m} \sum_{i \in S_j} x_i = \sum_{i \in \hat{S}} x_i$ y $\sum_{j=1}^{m} v_c^m(S_j) = v_c^m(\hat{S})$, se tiene que $\sum_{i \in S_j} x_i \leq v_c^m(S_j), j = 1, ..., m \Rightarrow \sum_{i \in \hat{S}} x_i \leq v_c^m(\hat{S})$. ∎ Por la proposición anterior, el núcleo de los juegos de rutas de vehículos pueden ser redefinidos de la siguiente manera:

$$C(N, v_c^m) = \left\{ x \in \mathbb{R}^n | \sum_{i \in S} x_i \leq v_c^m(S), \forall S \in \mathcal{S}; \sum_{i \in N} x_i = v_c^m(N) \right\}.$$

Este resultado es de extremada utilidad, pues disminuye el número de problemas que deben ser resueltos para calcular la función característica del juego a las coaliciones factibles. Recordemos, además, que, para el caso de las coaliciones factibles, el cálculo de $v_c^m(S)$ se reduce a la resolución de un TSP.

Veamos un ejemplo de 6 jugadores, extraído de [27], en el que se aprecia la utilidad de la proposición anterior.

Ejemplo 3.6 *Supongamos que tenemos 6 clientes ubicados según la Figura 3.6 cuyas demandas son $d = (8, 24, 22, 6, 7, 10)$ y contamos con varios vehículos de*

capacidad 30. La matriz de costes de transporte es la que sigue:

$$c = \begin{pmatrix} - & 24 & 19 & 20 & 27 & 16 & 12 \\ & - & 17 & 31 & 44 & 36 & 23 \\ & & - & 16 & 29 & 35 & 25 \\ & & & - & 15 & 34 & 28 \\ & & & & - & 40 & 37 \\ & & & & & - & 13 \\ & & & & & & - \end{pmatrix}$$

En la Figura 3.6 se muestra la ruta óptima para la gran coalición, cuyo coste es 176. Para el cálculo del núcleo podemos hacer uso del resultado de la Proposición 3.5 y, entonces, sólo será necesario calcular el TSP asociado a las coaliciones factibles, que son las siguientes:

$$\begin{array}{llll} v_c^m(1) = 48, & v_c^m(1,3) = 75, & v_c^m(3,4) = 62, & v_c^m(1,5,6) = 76, \\ v_c^m(2) = 38, & v_c^m(1,4) = 96, & v_c^m(3,5) = 70, & v_c^m(1,4,5) = 123, \\ v_c^m(3) = 40, & v_c^m(1,5) = 76, & v_c^m(4,5) = 83, & v_c^m(1,4,6) = 106, \\ v_c^m(4) = 54, & v_c^m(1,6) = 59, & v_c^m(4,6) = 76, & v_c^m(4,5,6) = 92. \\ v_c^m(5) = 32, & v_c^m(2,4) = 75, & v_c^m(5,6) = 41, & \\ v_c^m(6) = 24, & & & \end{array}$$

De esta forma, hemos reducido el número de TSPs a resolver de 62 a 20, con el consiguiente ahorro computacional. Además, por la Proposición 3.3 sabemos que cualquier reparto en el núcleo ha de cumplir que $x_2 = 38$, $x_3 + x_4 = 62$ *y* $x_1 + x_5 + x_6 = 76$. *Así, el núcleo del juego estará formado por todos los repartos* $x \in \mathbb{R}^6$ *tales que:*

$$\begin{array}{lll} x_2 = 38, & x_3 + x_4 = 62, & x_1 + x_5 + x_6 = 76. \\ 35 \leq x_1 \leq 48, & x_1 + x_3 \leq 75, & \\ 25 \leq x_3 \leq 40, & x_1 + x_5 \leq 76, & \\ 17 \leq x_5 \leq 32, & x_3 + x_5 \leq 70, & \end{array}$$

3.3.1. El juego de rutas de vehículos con ruta fija

En línea con el juego del viajante con ruta fija se puede definir el juego de rutas de vehículos con ruta fija. No existe literatura sobre este juego, que noso-

tros sepamos, por lo que a continuación se definirá formalmente y se expondrán algunos resultados sobre este nuevo juego.

Sea $G = (N_0, A)$ el grafo que define un CVRP donde $N = \{1, 2, ..., n\}$ es el conjunto de agentes ($N_0 = N \cup 0$), c es la matriz de costes, d es el vector de demandas y hay m vehículos disponibles con capacidad C. Denotemos como $r_k, k = 1, ..., m$, la ruta factible asociada al vehículo k y como R_m, el conjunto de todas esas rutas. Se define el juego de rutas de vehículos con ruta fija (N, v_{c,R_m}) como el juego que se obtiene asociando a cada coalición S el coste de la ruta que se obtiene eliminando de las subrutas r_k los elementos de $N \setminus S$ y manteniendo el orden y la configuración de r_k. Obsérvese que las rutas, así definidas, pueden no ser óptimas para cada coalición, pero siempre serán factibles.

Definir así el juego tiene la ventaja de que evita tener que resolver un VRP para cada coalición, pero tiene la desventaja de que no se asegura que el coste asociada a ellas sea mínimo. Nótese, además, que para el caso del VRP cuya solución implique la utilización de un único vehículo, el juego de rutas de vehículos con ruta fija coincide con el juego del viajante con ruta fija. Veamos un ejemplo de cómo se calcularía este juego.

Ejemplo 3.7 *Retomemos el VRP del Ejemplo 3.4 cuyo grafo se encuentra en la Figura 3.5. En ese caso, las dos rutas que configuran la solución óptima son 0-1-2-0 y 0-3-0. La principal diferencia del juego con ruta fija con el anterior es que en cada coalición no se busca la ruta óptima. En este caso, por ejemplo, si tomamos la coalición* $S = \{1, 3\}$ *la configuración de rutas es: 0-1-0 y 0-3-0 y no 0-1-3-0, que sería lo óptimo. Así, el juego de rutas de vehículos con ruta fija viene dado por la siguiente función característica:* $v_{c,R_m} = (2; 2; 2; 3{,}7; 4; 4; 5{,}7)$ *y tiene núcleo no vacío.*

En el ejemplo anterior se ha trabajado con los mismos datos que el Ejemplo 3.4, pero en dicho ejemplo el núcleo era vacío y, en este caso, no lo es. Este hecho no se reduce a este ejemplo, sino que se da para todo juego de este tipo.

Teorema 3.2 *Los juegos de rutas de vehículos con ruta fija, tales que la ruta fija sea óptima, tienen núcleo no vacío.*

Para cada ruta r_k, denotemos como S_k a la coalición visitada por esa ruta, con $k = 1, ..., n$.

Demostración. Recordemos que para que el núcleo del juego (N, v_{c,R_m}) sea no vacío debe existir algún reparto $x \in \mathbb{R}^n$ que cumpla las restricciones (3.4) y (3.5).

Se tiene, además, que los subjuegos[4] (S_k, v_{c,R_m,S_k}) coinciden con un TSG con ruta fija sobre los agentes de la coalición, por lo que tendrán núcleo no vacío. Tomemos, entonces, un reparto $x \in \mathbb{R}^n$ tal que $\forall i \in S_k, x_i \in C(S_k, v_{c,R_m,S_k})$.

Dado que el coste total de las rutas coincide con la suma de las subrutas, x cumplirá la propiedad de eficiencia (3.4):

$$\sum_{j=1}^{m} v_{c,R_m}(S_j) = v_{c,R_m}(N), \quad \sum_{i \in S_j} x_i = v_{c,R_m}(S_j) \quad \Rightarrow \quad \sum_{i \in N} x_i = v_{c,R_m}(N).$$

Nos resta, por tanto, demostrar que x cumple racionalidad coalicional. Podemos diferenciar entre las coaliciones $S \subseteq S_k$ y las coaliciones $S \nsubseteq S_k$. En el primer caso, dado que el núcleo es siempre no vacío en los subjuegos, es directo comprobar que $x(S) \leq v_{c,R_m}(S), \forall S \subseteq S_k, k = 1, ..., m$.

Para el segundo caso, debemos destacar que dichas coaliciones S siempre pueden ser descompuestas en subcoaliciones T_k cada una de las cuales esté contenida en una coalición S_k, es decir, $T_k = S \cap S_k$. Además, el coste asociado a dichas coaliciones S es la suma de las subrutas de cada subcoalición T_k $(v_{c,R_m}(S) = \sum_{k=1}^{m} v_{c,R_m}(T_k))$ y por tener núcleo no vacío los juegos (S_k, v_{c,R_m,S_k}), $\sum_{i \in T_k} x_i \leq v_{c,R_m}(T_k), \forall T_k \subseteq S_k, k = 1, ..., m$. Así,

$$\sum_{i \in S} x_i = \sum_{k=1}^{m} \sum_{i \in T_k} x_i \leq \sum_{k=1}^{m} v_{c,R_m}(T_k) = v_{c,R_m}(S).$$

Se ha demostrado que x cumple las restricciones (3.4) y (3.5), por lo que queda así demostrado que el núcleo es no vacío. ∎

Otro resultado interesante asociado a este tipo de juegos es que, a la hora de caracterizar el núcleo, las restricciones (3.5) asociadas a coaliciones $S \nsubseteq S_k$ son redundantes. Esto se debe a que $v_{c,R_m}(S) = \sum_{k=1}^{m} v_{c,R_m}(T_k)$ y que $\sum_{i \in T_k} x_i \leq$

[4]Dada una coalición $S \subseteq N$, el subjuego de (N,v) restringido a S se denota por (S, v_S) y se define por $v_S(T) = v(T), \forall T \subseteq S$.

$v_{c,R_m}(T_k), \forall T_k \subseteq S_k, k = 1, ..., m$. Por lo que:

$$\sum_{i \in T_k} x_i \leq v_{c,R_m}(T_k), \forall T_k \subseteq S_k, k = 1, ..., m \quad \Rightarrow \quad \sum_{i \in S} x_i \leq v_{c,R_m}(S).$$

Por último, tomemos el Ejemplo 3.6 y apliquemos los resultados anteriores.

Ejemplo 3.8 *En el Ejemplo 3.6, cuyo grafo se encuentra en la Figura 3.6, hay 6 agentes que desean cooperar y se sabe que la solución óptima al problema VRP que generan los agentes es: (0-2-0), (0-1-6-5-0), (0-3-4-0); con un coste de 176 unidades. Esta configuración de rutas da lugar a tres coaliciones disjuntas factibles:* $S_1 = \{2\}, S_2 = \{1, 5, 6\}$ *y* $S_3 = \{3, 4\}$. *Dado que el número de coaliciones es muy elevado, no calcularemos la función característica completa de este juego. A continuación, utilizaremos el razonamiento utilizado en la demostración del Teorema 3.2, para demostrar que el núcleo de este juego es no vacío.*

Calculemos en primer lugar las funciones características de las coaliciones de las subrutas:

$$v_{c,R_m,S_1} = (38)$$

$$v_{c,R_m,S_2} = (48, 32, 24, 76, 59, 41, 76)$$

$$v_{c,R_m,S_3} = (40, 54, 62).$$

Dado que estos subjuegos pueden verse como TSG con ruta fija, tienen núcleo no vacío. Por tanto, un reparto $x \in \mathbb{R}^6$ *tal que* $x_2 \in C(S_1, v_{c,R_m,S_1})$, $(x_1, x_5, x_6) \in C(S_2, v_{c,R_m,S_2})$ *y* $(x_3, x_4) \in C(S_3, v_{c,R_m,S_3})$ *cumplirá las siguientes restricciones:*

$$\begin{array}{lll} x_1 \leq 48, & x_1 + x_5 \leq 76, & x_2 = 38 \\ x_2 \leq 38, & x_1 + x_6 \leq 59, & x_1 + x_5 + x_6 = 76, \\ x_3 \leq 40, & x_5 + x_6 \leq 41, & x_3 + x_4 = 62. \\ x_4 \leq 54, & x_3 + x_4 \leq 62, & \\ x_5 \leq 32, & & \\ x_6 \leq 24, & & \end{array}$$

En primer lugar observamos que $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 176$, *por lo que* $x \in \mathbb{R}^6$ *cumple la propiedad de eficiencia. Además,* $x_i \leq v_{c,R_m}(i), \forall i = 1, ..., 6$, *por lo que* $x \in \mathbb{R}^6$ *también cumple racionalidad individual. Nos resta, por tanto, demostrar que dicho reparto cumple racionalidad coalicional. Para cualquier subcoalición* S *de* S_k *es directo comprobar que se cumple* $x(S) \leq v_{c,R_m}(S), \forall S \subseteq S_k, k = 1, ..., m$.

Nótese que, tal y cómo se ha definido este juego, los costes asociados a las coaliciones compuestas por agentes de distintos S_k se obtienen como suma de los costes asociados a las rutas de los agentes de cada S_k. Por ejemplo, tomemos la coalición $\{1,2,3,4,6\}$. Si eliminamos el agente 5 de la configuración de rutas, obtendremos las rutas (0-2-0), (0-1-6-0), (0-3-4-0) por lo que el coste asociado a esta coalición será:

$$v_{c,R_m}(\{1,2,3,4,6\}) = v_{c,R_m}(\{2\})+v_{c,R_m}(\{1,6\})+v_{c,R_m}(\{3,4\}) = 38+59+62 = 159.$$

Además, se tiene que $x_2 \leq 38$, $x_1 + x_6 \leq 59$ y que $x_3 + x_4 \leq 62$, por lo que

$$x_1+x_2+x_3+x_4+x_6 \leq v_{c,R_m}(\{2\})+v_{c,R_m}(\{1,6\})+v_{c,R_m}(\{3,4\}) = v_{c,R_m}(\{1,2,3,4,6\}).$$

Este razonamiento puede seguirse para cualquier coalición que pueda dividirse en subcoaliciones de S_k, por lo que utilizando el mismpo procedimiento sobre las coaliciones restantes, quedaría demostrado que el núcleo de este juego es no vacío. A continuación, caracterizamos dicho núcleo teniendo en cuenta que las restricciones asociadas a coaliciones no contenidas en S_k son redundantes. Para ilustrar este resultado, tomemos de nuevo la coalición $\{1,2,3,4,6\}$. Su restricción asociada es: $x_1 + x_2 + x_3 + x_4 + x_6 \leq 159$. Como $x_2 \leq 38$, $x_1 + x_6 \leq 59$ y $x_3 + x_4 \leq 62$, siempre se tendrá que $x_1 + x_2 + x_3 + x_4 + x_6 \leq 159$, por lo que esta restricción será redundante. Por lo tanto, el núcleo de este juego viene dado por los repartos $x \in \mathbb{R}^6$ que cumplen las siguientes restricciones:

$$\begin{array}{ccc} x_2 = 38, & x_3 + x_4 = 62, & x_1 + x_5 + x_6 = 76, \\ 35 \leq x_1 \leq 48, & 8 \leq x_3 \leq 40, & 17 \leq x_5 \leq 32, \\ & 0 \leq x_6 \leq 24. & \end{array}$$

3.3.2. Líneas de trabajo futuras

Como ocurre con el juego del viajante de comercio, no existen publicaciones, que nosotros sepamos, en las que se estudien propiedades de distintas reglas de reparto para el juego de rutas de vehículos. De nuevo, es particularmente interesante definir estas reglas cuando el núcleo es no vacío. Por tanto, pueden plantearse varias líneas de investigación:

- En línea con las propiedades de Monge y de Kalmanson, sería interesante

encontrar casos particulares del VRP para los que sí se pueda asegurar que el núcleo es no vacío, por ejemplo, cuando la matriz de distancias cumpla alguna propiedad concreta. Para ello, podría utilizarse la Proposición 3.4 e intentar estudiar en qué casos $\bar{z}$ coincide con z.

- Estudiar las propiedades de las reglas de reparto de costes definidas en el Capítulo 2 (valor de Shapley, nucleolo, etc.), o de otras reglas planteadas en la literatura de la teoría cooperativa de juegos, en el caso del juego de rutas de vehículos con ruta fija (recordemos que se trata de un juego que siempre tiene núcleo no vacío).

- Extender el método propuesto por Göthe-Lundgren et al. [27], para calcular el nucleolo del VRG básico, a variantes del VRG más complejas, como se hizo en el artículo de Tae et al. [56] con el VRP con ventanas de tiempo.

Bibliografía

[1] Balinski, M., Quandt, R. (1964).*On an integer program for a delivery problem*. Operations Research 12, 300-304.

[2] Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, USA.

[3] Borm, P., Hamers, H., Hendrickx, R. (2001). *Operations Research Games: A survey*. TOP, 9(2), 139-216.

[4] Bullnheimer, B., Hartl, R.F., Strauss, C. (1999). *An improved ant system for the vehicle routing problem*. Annals of Operation Research 89, 319-328.

[5] Chardaire, P. (2001). *The core and nucleolus of games: A note on a paper by Göthe-Lundgren et al.* Mathematical Programming 90, 147-151.

[6] Christofides, N. (1973). *The Optimum Traversal of a Graph*. Omega 1, 719-732.

[7] Christofides, N. (1976). *Worst-case analysis of a New Heuristic for the Traveling Salesman Problem*. Report 388, Graduate school of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA.

[8] Clarke G., Wright, J.W. (1964). *Scheduling of vehicles from a central depot to a number of delivery points*. Operations Research 12, 568-581.

[9] Clay Mathematics Institute. (2000). *Millenium problems*. www.claymath.org/millenium/.

[10] Climer, S., Zhang, W. (2004). *Take a walk and cluster genes: A TSP-based approach to optimal rearrangement clustering*. 21st International Conference on Machine Learning (ICML'04), Banff, Alberta, Canada, 169-176.

[11] Colorni, A., Dorigo, M., Maniezzo, V. (1991). *Distributed optimization by ant colonies*. In F. Varela and P. Bourgine, editors, Proceedings of the European Conference on Artificial Life, Elsevier, Amsterdam.

[12] Cook S.A. (1971) *The complexity of theorem-proving procedures*. ACM Press, New York, USA. 151-158.

[13] Dantzig, G.B., Ramser, J.H. (1959). *The truck dispatching problem*. Management Science, 6-80.

[14] Dantzig, G., Fulkerson, R., Johnson, S. (1954) *Solution of a large-scale traveling-salesman problem*. Operations Research 2, 393-410.

[15] Edmonds J. (1965). *Paths, trees and flowers*. Canadian Journal of Mathematics 17, 449-467.

[16] ein alter Commis-Voyageur. (1832). *Der Handlungsreisende-wie er sein soll und was er zu thun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiss zu sein*. B. Fr. Voigt, Ilmenau.

[17] Eastman, W.L. (1958) *Linear Programming with Pattern Constrainst*. Ph.D. Thesis. Department of Economics, Harvard University, Cambridge, Massachusetts, USA.

[18] Euler, L. (1736). *Solutio Problematics ad Geometrica Situs Pertinentis*. Commentarii academiae scientarum Petropolitanae 8, 128-140.

[19] Fishburn, P.C., Pollack, H.O. (1983). *Fixed route cost allocation*. American Mathematical Monthly 90, 366-378.

[20] Fisher, M.L., Jaikumar, R. (1981). *A generalized assignment heuristic for vehicle routing*. Networks 11, 109-124.

[21] Flood, M.M. (1956). *The traveling-salesman problem*. Operations Research 4, 61-75.

[22] Flood, M.M. (1984). *Merrill Flood (with Albert Tucker), Interview of Merrill Flood in San Francisco on 14 May 1984*. The princeton Mathematics Community in the 1930s, Transcript number 11 (PMC11). Princeton University.

[23] Fonlupt, J., Naddef, D. (1985). *The traveling salesman problem in graphs with some excluded minors*. Technical Report 557, Université Scientifique et Medicale de Grenoble, France.

[24] Gately, D. (1974). *Sharing the gains from regional cooperation: a game theoretic aplication to planning investment in electric power*. International Economy Review 15, 195-208.

[25] Gilmore, P.C., Gomory, R.E. (1964). *Sequencing a one state-variable machine: A solvable case of the traveling salesman problem*. Operations Research 12, 655-679.

[26] Golden, B.L., Wong, R.T. (1981). *Capacitated Arc Routing Problems*. Networks 11, 305-315.

[27] Göthe-Lundgren, M., Jörnsten, K., Värbrand, P. (1996). *On the nucleolus of the basic vehicle routing game*. Mathematical Programming 72, 83-100.

[28] Guan, M. (1962). *Graphic programming using odd and even points*. Chinese Math. 1: 273-277.

[29] Held, M., Karp, R.M. (1962). *A dynamic programming approach to sequencing problems*. Journal of the Society of Industrial an Applied Mathematics 10, 196-210.

[30] Hierholzer, C. (1873). *Über die endlichen und unendlichen Graphen*. Akademische Verlagsgesellschaft, Leipzig.

[31] Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI.

[32] Kawamura, H., Yamamoto, M., Suzuki, K., Ohuchi, A. (1998). *Cooperative search on pheromone communication for vehicle routing problems*. IEEE Transactions on Fundamentals, E81-A: 1089-1096.

[33] Kindervater, G.A.P., Savelsbergh, M.W.P. (1997). *Vehicle routing: Handling edge exchanges. In E.H.L. Aarts and J.K. Lenstra, editors. Local search in Combinatorial Optimization*. Wiley, Chichester, UK, 337-360.

[34] Kuipers, J. (1993). *A note on the 5-person traveling salesman game*. Mathematical Methods of Operation Research (ZOR) 38, 131-140.

[35] Kuipers, J., Solymosi, T., Aarts, H. (2000). *Computing the nucleolus of some combinatorially-structured games*. Mathematical Programming 88, 541-563.

[36] Laporte, G. (1992). *The Vehicle Routing Problem: An overview of exact and aproimate algorithms*. European Journal of Operations Research 59: 345-358.

[37] Laporte, G., Osman, I.H. (1995). *Routing Problems: A bibliography*. Annals of Operations Research 61: 227-262.

[38] Lenstra, J.K., Rinnooy Kan, A.H.G. (1976). *On General Routing Problems*. Networks 6, 273-280.

[39] Lin, S., Kernighan, B.W. (1973). *An effective heuristic algorithm for the traveling-salesman problem*. Operation Research 21, 498-516.

[40] Little, J.D.C., Murty, K.G., Sweeny, D.W., Karel, C.. (1963). *An algorithm for the traveling salesman problem*. Operations Research 11, 972-989.

[41] Menger, K. (1931). *Bericht über ein mathematisches Kolloquium*. Monatshefte für Mathematik und Physik 38, 17-38.

[42] Mirás Calvo, M.A., Sánchez Rodríguez, E. (2008). *Juegos cooperativos con utilidad transferible usando Matlab: TUGlab*. Colección Monografías de la Universidad de Vigo. Serie de tecnología y ciencias experimentales, número 16.

[43] Morton, G., Land, A.H. (1955). *A contribution to the "traveling-salesman"problem*. Journal of the Royal Statistical Society, Series B, 17, 185-194.

[44] Okamoto, Y. (2004). *Traveling sakesman games with the Monge property*. Discrete Applied Mathematics 138, 349-369.

[45] Oliver, I.M., Smith, D.J., Holland, J.R.C. (1987). *A study of permutation crossover operators on the traveling salesman problem.*In the J.J. Grefenstette, editor, Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum, Hillsdale, NJ, 224-230.

[46] Orloff, C.S. (1974). *A Fundamental Problem in Vehicle Routing*. Networks 4, 35-64.

[47] Osman, I.H. (1993). *Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem*. Annals of Operations Research 41, 421-451.

[48] Potters, J.A.M., Curiel, I.J., Tijs, S.H. (1987). *Traveling salesman games*. Technical Report, Catholic University Nijmegen, The Netherlands.

[49] Potters, J.A.M., Curiel, I.J., Tijs, S.H. (1992). *Traveling salesman games*. Mathematical Programming 53, 199-211.

[50] Robinson, J. (1949). *On the Hamiltonian game (a traveling salesman problem)*. RAND Research Memorandum RM-303. RAND Corporation, Santa Monica, California, USA.

[51] Ryan, D. M., Hjorring, C., Glover, F. (1993). *Extensions of the Petal Method for Vehicle Routing*. Journal of the Operational Research Society 44, 289-296.

[52] Schmeidler, D. (1969). *The nucleolus of a characteristic function game*. SIAM J. Appl. Math. 17, 1163-1170.

[53] Shapley, L.S. (1953). *Additive and Non-Additive Set Functions*. Ph. D. Thesis, Princeton University.

[54] Shapley, L.S. (1971). *Cores of convex games*. International Journal of Game Theory 1, 11-26.

[55] Straffin, P., Heaney, J.P. (1981). *Game Theory and the Tenessee Valley Authority*. International Journal of Game Theory 10, 35-43.

[56] Tae, H., Jun, Y., Kim, B. (2010). *The Vehicle Routing Cost Allocation Problem with Time Windows*. The 11th Asia Pacific Industrial Engineering and Management Systems Conference. Melaka, Malasia.

[57] Taillard, E.D. (1993). *Parallel iterartive search methods for vehicle routing problems*. Networks 23, 661-673.

[58] Tamir, A. (1989). *On the core of a traveling salesman cost allocation game*. Operation Research Letters 8, 31-34.

[59] Thomson, P.M., Psaraftis, H.N. (1993). *Cyclic transfer algorithms for multivehicle routing and scheduling problem*. Operations Research 41, 935-946.

[60] Toth, P., Vigo, D. (2002). *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications.

[61] Toth, P., Vigo, D. (2003). *The granular tabu search and its application to the vehicle routing problem*. INFORMS Journal on Computing 15, 333-346.

[62] Van Breedam, A. (1994). *An analysis of the behavior of heuristics for the vehicle routing problem for a selection of problems with vehicle-related, customer-related, and time-related constraints*. Ph.D. dissertation, University of Antwerp.

[63] Willard, J.A.G. (1989). *Vehicle routing using r-optimal tabu search*. M.sc. dissertation, The Management School, Imperial College, London.

[64] Wren, A. (1971). *Computers in Transport Planning and Operation*. Ian Allan, London.

[65] Young, H.P. (1994). *Handbook of Game Theory, Volume 2, Edited by R.J. Aumman y S. Hart*. Capítulo 34: 1193-1235.