Trabajo Fin de Máster

Cooperación en los problemas de arborescencias de mínimo coste

María Cristina Fraga Canosa

Máster en Técnicas Estadísticas

Curso 2021-2022

Propuesta de Trabajo Fin de Máster

Título en galego: Cooperación nos problemas de arborescencias de custo mínimo
Título en español: Cooperación en los problemas de arborescencias de mínimo coste
English title: Cooperation on minimum cost arborescence problems
Modalidad: Modalidad A
Autor/a: María Cristina Fraga Canosa, Universidade da Coruña
Director/a: Silvia María Lorenzo Freire, Universidade da Coruña
Breve resumen del trabajo: Este trabajo se centra en el estudio de los problemas de arborescencias de mínimo coste. Además, se introducen los problemas de arborescencias de mínimo coste con grupos, definiendo e implementando una regla de reparto.

Don/doña Silvia María Lorenzo Freire, Profesora Titular del Departamento de Matemáticas de la Universidade da Coruña, informa que el Trabajo Fin de Máster titulado

Cooperación en los problemas de arborescencias de mínimo coste

fue realizado bajo su dirección por don/doña María Cristina Fraga Canosa para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, da su conformidad para su presentación y defensa ante un tribunal.

En A Coruña, a 13 de junio de 2022.

LORENZO FREIRE SILVIA MARIA - 32680954R

Firmado digitalmente por LORENZO FREIRE SILVIA MARIA - 32680954R
Fecha: 2022.06.13 11:43:29 +02'00'

El/la director/a:
Don/doña Silvia María Lorenzo Freire

El/la autor/a:
Don/doña María Cristina Fraga Canosa

Declaración responsable. Para dar cumplimiento a la Ley 3/2022, de 24 de febrero, de convivencia universitaria, referente al plagio en el Trabajo Fin de Máster (Artículo 11, Disposición 2978 del BOE núm. 48 de 2022), **el/la autor/a declara** que el Trabajo Fin de Máster presentado es un documento original en el que se han tenido en cuenta las siguientes consideraciones relativas al uso de material de apoyo desarrollado por otros/as autores/as:

- Todas las fuentes usadas para la elaboración de este trabajo han sido citadas convenientemente (libros, artículos, apuntes de profesorado, páginas web, programas,...)
- Cualquier contenido copiado o traducido textualmente se ha puesto entre comillas, citando su procedencia.
- Se ha hecho constar explícitamente cuando un capítulo, sección, demostración,... sea una adaptación casi literal de alguna fuente existente.

Y, acepta que, si se demostrara lo contrario, se le apliquen las medidas disciplinarias que correspondan.

Índice general

Resumen

Resumen en español

En este trabajo se hace una revisión de los problemas de arborescencias de mínimo coste. En este tipo de problemas varios agentes pretenden conectarse directa o indirectamente a algún tipo de servicio mediante un mínimo coste. Las conexiones entre ellos mismos y el propio servicio pueden variar en función de las características del problema. En algunos casos las conexiones serán idénticas en ambos sentidos y en otros diferirán. También existirán situaciones en las que los agentes tiendan a agruparse debido a diversos motivos, mientras que en otros momentos decidirán permanecer aislados. Estas dos características básicas darán lugar a los cuatro tipos de problemas a tratar en este trabajo; los de árboles de mínimo coste, las arborescencias de mínimo coste y estos dos mismos problemas teniendo en cuenta una estructura grupal de asociación entre agentes. La finalidad de cualquiera de los mencionados problemas será la de obtener la red de conexión para todos los agentes al menor coste posible. La parte cooperativa aplicable a estos problemas se encargará de estudiar el reparto del coste total de conexión entre los agentes.

Además, se introducirán los problemas de arborescencias de mínimo coste con grupos, tratando de generalizar las definiciones existentes para los problemas de árboles de mínimo coste con grupos. Por último, se introducirá e implementará una regla de reparto para los problemas de arborescencias de mínimo coste con grupos.

English abstract

In this paper, a review of the minimum cost arborescence problems is made. In these kind of problems, several agents intend to connect directly or indirectly to some type of service at a minimum cost. The connections between themselves and the service itself may be different depending on the characteristics of the problem. In some cases the connections will be identical in both directions and in others they will differ. There will also be situations in which agents tend to group together for various reasons, while at other times they decide to remain isolated. These two basic characteristics will give rise to the four types of problems to be treated in this work; minimum cost tree probles, minimum cost arborescence problems and these two same problems taking into account a group structure of association between agents. The purpose of any of the aforementioned problems will be to obtain the connection network for all the agents at the lowest possible cost. The cooperative part applicable to these problems will be in charge of studying the distribution of the total connection cost between the agents.

In addition, minimum cost arborescence problems with groups will be introduced, trying to generalize the existing definitions for the minimum cost tree problems with groups. Finally, a sharing rule will be introduced and implemented for minimum cost arborescence problems with groups.

Prefacio

Un sinfín de actividades que realizamos cotidianamente y muchos de los servicios de los que hacemos uso a diario pueden ser estudiados mediante la teoría de grafos. Desde el paquete que es recibido a través de un servicio de mensajería, hasta cada vez que se enciende una lámpara o se abre el grifo del agua, todas y cada una de esas acciones pueden estar asociadas a un problema de grafos.

El origen de la teoría de grafos se remonta al siglo XVIII, cuando el matemático y físico suizo Leonhard Euler publica su trabajo acerca del "problema de los siete puentes de Königsberg". El problema consistía en descubrir si era posible recorrer la ciudad y cruzar todos sus puentes una sola vez regresando al punto de partida. El método que utilizó para plantearlo fue el de transformar el mapa de la ciudad en un gráfico esquemático y equivalente (grafo). Sin embargo, el término grafo tal y como hoy lo entendemos no es acuñado hasta el siglo XIX por el químico inglés Edward Frankland.

Un siglo después del trabajo de Euler, en 1847, Gustav Kirchhoff se sirvió de la teoría de grafos al publicar sus leyes acerca de las corrientes en los circuitos eléctricos, siendo esta la primera aplicación en el campo de la ingeniería. Unos pocos años mas tarde, Francis Guthrie plantea el "problema de los cuatro colores", que afirma que es posible rellenar cualquier mapa con cuatro colores de forma que dos países vecinos nunca coincidan en el color.

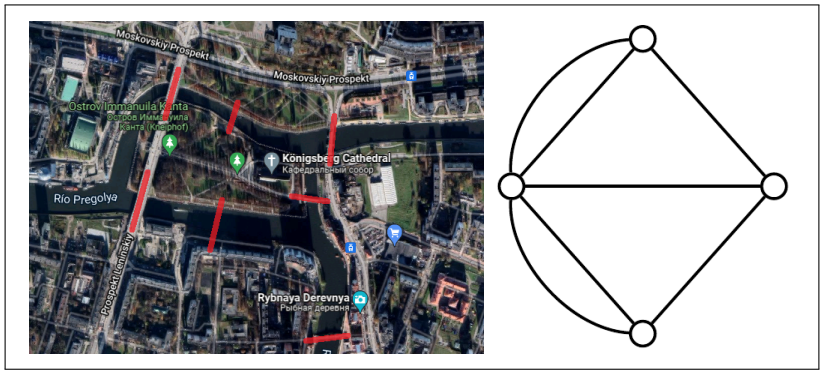


Figura 1: Plano de Königsberg con los siete puentes y grafo que representa el problema. [1]

[1] Obtenido de Google Maps.

Con el paso de los años la teoría de grafos se ha ido extendiendo y aplicando a diferentes ramas del conocimiento como la química, la biología o las ciencias sociales. En las últimas décadas y debido al auge de la globalización y principalmente al desarrollo vertiginoso de las redes de comunicaciones y transporte, la teoría de grafos ha tomado un gran impulso y su estudio ha ido en aumento.

Para que un problema pueda ser resuelto mediante la teoría de grafos, éste ha de ser representable mediante un grafo, que no es más que un conjunto de nodos o vértices que representan a los elementos

que conforman el problema, ya sean personas, ciudades, dispositivos; y aristas o arcos que simbolizan el método de unión entre los nodos. A través de la unión de nodos y arcos se genera una red que puede llegar a ser representativa de cualquier problema de este tipo.

A la hora de hacernos una idea de cómo se puede trasladar esta clase de problemas a la realidad, encontramos en las redes de telecomunicaciones un buen ejemplo. En esta categoría de redes se intercambia información de todo tipo entre dispositivos mediante protocolos de comunicación. Cada uno de los elementos que conforman la red han de transmitir y recibir una cantidad de datos diferente, de forma que el flujo de la información es distinto en un sentido y en otro. Esta asimetría en la capacidad de la red puede plasmarse mediante arcos dirigidos y, por tanto, a través de un problema de arborescencias.

En la figura 2 aparece representada la que podría ser la red LAN de una oficina, en la que se aprecia cómo los diferentes aparatos propios de un entorno de trabajo se conectan entre sí y con servidores. En este caso, resulta pertinente contar con conexiones diferentes teniendo en cuenta el sentido de las mismas, ya que el volumen de información y datos que se envía y recibe difiere en función de cada dispositivo. Resulta evidente que la información que puede llegar a recibir un ordenador de una impresora o un fax es mucho menor que la que se envía a éstos. De igual modo, una petición desde un ordenador a cualquier servidor web requiere menos información que todos los datos que devuelve el servidor web al ordenador. Teniendo esto en cuenta, podría resultar de interés un problema en el que el objetivo sea detectar cuál sería la red que soporta mayor capacidad.

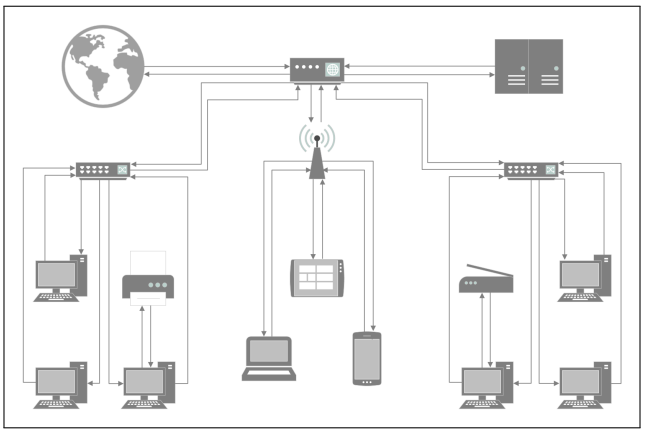


Figura 2: Diagrama de red LAN.

De las diferentes categorías de problemas relativos a la teoría de grafos, este trabajo se centra fundamentalmente en dos tipos de ellos, en los problemas de arborescencias de mínimo coste y en los problemas de árboles de mínimo coste. Si bien en la estructura del trabajo son tratados como problemas independientes, la intención es que quede bien claro desde el comienzo que los segundos son una forma específica de los primeros, es decir, los problemas de árboles de mínimo coste son una categoría de problemas que se encuentran englobados dentro de los problemas de arborescencias de mínimo coste. El hecho de que los problemas de árboles de mínimo coste cobren tanta importancia en este trabajo se debe principalmente a que existe un mayor recorrido en su estudio, ya que resulta mucho más sencillo trabajar con este tipo de problemas que con las arborescencias de mínimo coste.

Otro de los aspectos a tener en cuenta en este trabajo es que la motivación de ambos problemas es la de minimizar costes y no otra. Para lograr este fin, es necesario en primer lugar encontrar la forma más eficaz de conectar la red para después discernir el reparto de los costes de conexión de la red de la forma más eficiente posible para las partes implicadas. Con respecto al primer objetivo,

existen diferentes algoritmos de búsqueda que facilitan enormemente el proceso de obtención de una red de conexión óptima, presentándose en este trabajo algunos de los más utilizados. En cuanto a la asignación de costes, también se pueden encontrar diversas reglas que proponen repartos dispares. La idea que se esconde detrás de estos problemas es que exista la posibilidad de cooperación entre los agentes, de forma que se pueda llegar a un acuerdo o solución entre todos los implicados.

En cuanto a la diferencia existente entre ambos problemas, la más fundamental reside en la orientación de los arcos o aristas. En los problemas de árboles de mínimo coste éstos no son orientados, lo cual implica que el valor de la conexión en ambos sentidos es el mismo. Por el contrario, en los problemas de arborescencias de mínimo coste las aristas son orientadas, lo que se traduce en que el valor de la conexión en un sentido y en otro no ha de ser necesariamente el mismo. Este matiz que puede resultar sutil e incluso insignificante se puede entender más claramente y comprender su importancia con ejemplos de la vida real. Si pensamos en el tráfico aéreo, somos conscientes de que las rutas que unen dos poblaciones no son las mismas en un sentido y en el otro. Por diversos motivos de seguridad y eficiencia los aviones recorren diferentes trayectorias, teniendo como resultado tiempos, gastos de combustible y costes distintos en función del sentido del vuelo. Sin embargo, si pensamos en una carretera de calzada única nos damos cuenta de que el viaje es análogo en ambos sentidos, puesto que tendremos que recorrer los mismo kilómetros, atravesar los mismos túneles, salvar las mismas montañas, etc.

Otro ejemplo real y específico en el que es posible aplicar esta clase de problemas es en la construcción de la red de suministro de agua de los pueblos del entorno del municipio de Pereiro de Aguiar (Ourense). Lo que sucede en este caso, estudiado en Bergantiños y Lorenzo (2004), es que la única fuente de abastecimiento de agua de estas viviendas eran sus propios pozos de agua, que en ocasiones, principalmente en la época de verano, resultaban insuficientes para satisfacer las necesidades de agua de los vecinos. Este hecho lleva a las autoridades de la zona a tomar la decisión de construir un embalse de agua junto con la red de tuberías y depósitos necesarios para abastecer a los vecinos.

Figura 3: Embalse de Cachamuíña en Pereiro de Aguiar. [2]

[2] Obtenido de la Web de Turismo de Pereiro de Aguiar.

Para ello fue preciso, en primer lugar, construir el propio embalse además de la red de tuberías para conectarlo con los diferentes depósitos que serían exclusivos para cada una de las poblaciones. Los costes del embalse y de su conexión a los depósitos fueron cubiertos en su totalidad por el ayuntamiento. Posteriormente hubo que conectar dichos depósitos con cada una de las casas, para lo cual cada vecino tuvo que decidir cómo conectar su casa a la red de suministro, ya que eran los propietarios de las

viviendas quienes debían de pagar dicho coste. Así podían decidir conectarse directamente al depósito o a otra casa vecina que se fuera a conectar. Una vez planeada la red, el ayuntamiento llevaría a cabo las obras y pasaría a ser el propietario de la misma.

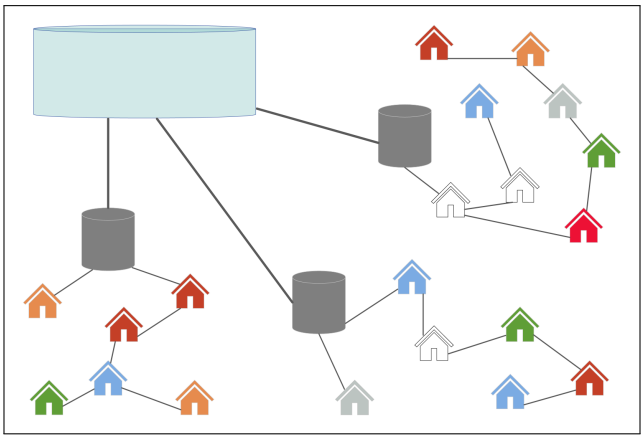


Figura 4: Esquema asociado al problema del embalse.

El esquema de la figura 4 podría servir para hacerse una idea más clara acerca del caso descrito previamente; en una primera etapa se construye y conecta el embalse con los depósitos para cada pueblo y posteriormente se conectan las casas con los depósitos. La falta de previsión hizo que los vecinos se conectaran en distintas etapas, de manera que algunos vecinos conectados en etapas anteriores se quejaban de pagar costes más altos que otros vecinos que se conectaron posteriormente.

La primera etapa, de unión del embalse con los depósitos, al correr a cargo del ayuntamiento en su totalidad, no supone ningún problema. Sin embargo, en la segunda etapa, en la que las viviendas de cada pueblo se conectan al depósito correspondiente de manera aislada, sin que existan conexiones entre casas o depósitos de poblaciones diferentes, sí que hay un problema que resolver. Lo que resulta de interés en este caso es ver cómo construir las canalizaciones de la forma más barata posible y comprobar el reparto de esos costes entre las casas. En el caso de que todas las casas se hubiesen conectado al mismo tiempo, para obtener una red de tuberías lo más eficiente posible (en este caso, lo más económica), se podría haber planteado un problema de árboles de mínimo coste. Una vez discernidos los costes de conexión entre casas y con el depósito, simplemente sería necesario obtener un árbol de mínimo coste para hallar la red de tuberías óptima y después repartir los costes entre los propietarios de las viviendas.

El grafo que se muestra en la figura 5 sería una opción viable para plasmar una situación como la planteada en la construcción de la red de suministro de agua de cada uno de los pueblos de Pereiro de Aguiar. Nos encontramos ante un grafo no dirigido, debido a que los costes de conexión entre las casas y el embalse se presuponen iguales en ambas direcciones. Si por ejemplo, debido a la orografía del terreno los costes de conexión fuesen distintos en función del sentido de la conexión, entonces sería necesario trabajar con arcos dirigidos. Generalmente aquellas conexiones con costes demasiado elevados no interesan, de forma que no se tienen en cuenta a la hora de realizar el grafo y se prescinde de ellas. En el caso de que existiera algún depósito compartido entre pueblos o que todos ellos se conectaran directamente al embalse sin necesidad de depósitos intermedios nos encontraríamos, como veremos a continuación, ante un problema de árboles con grupos.

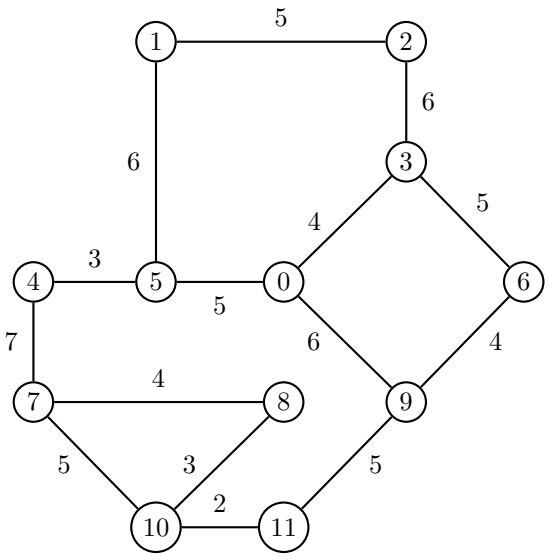


Figura 5: Grafo asociado al problema de conexión de las casas al depósito.

La segunda parte del trabajo contempla la posibilidad de que los agentes que componen la red formen distintas agrupaciones. Resulta interesante tener en cuenta esta circunstancia en aquellas ocasiones en las que por diferentes motivos los agentes tienen algún tipo de afinidad y deciden agruparse. Esta asociación puede venir motivada por proximidad geográfica, afinidad política o ideológica, etc. Lo que sucede en estos casos es que, si bien los grupos no afectan a la hora de realizar la conexión de la red, sí es necesario tener en cuenta la estructura generada por dichas asociaciones a la hora de llevar a cabo el reparto de los costes, ya que esa distribución puede diferir de un escenario a otro.

Los objetivos que se persiguen a lo largo del trabajo son los siguientes:

- Realizar una revisión de la cooperación en los problemas de arborescencias de mínimo coste y por extensión, en los problemas de árboles de mínimo coste.
- Estudio de los problemas de arborescencias de mínimo coste con grupos. Para ello se llevará a cabo una revisión de los problemas de árboles de mínimo coste con grupos, intentando trasladar los resultados obtenidos en éstos a los problemas de arborescencias de mínimo coste con grupos.
- Proporcionar en lenguaje de programación R una herramienta que permita obtener un reparto del coste mínimo de conexión entre los agentes en los problemas de arborescencias con y sin grupos. Esta herramienta complementará a los paquetes ya existentes optrees y cooptrees.

El trabajo está planteado de forma que el primer capítulo trata sobre árboles de mínimo coste y el segundo atañe a las arborescencias de mínimo coste. En ambos capítulos se introduce cada uno de los problemas para a continuación exponer los algoritmos de búsqueda que se han seleccionado. Se prosigue explicando los juegos cooperativos y la forma irreducible de estos problemas para finalmente exponer algunas reglas de reparto de costes.

En los capítulos tercero y cuarto nos ocuparemos de los problemas con grupos tanto en el caso de los árboles de mínimo coste como en el de las arborescencias de mínimo coste. Estos capítulos serán más breves para evitar la repetición y el solapamiento, ya que parte de la información aportada en los dos primeros capítulos es común a este tipo de problemas. Su contenido se divide en un primer apartado introductorio de presentación del problema seguido del planteamiento de la regla utilizada para la obtención de un reparto de los costes.

Se contempla un último capítulo dedicado a la programación en R de la regla de reparto de costes para problemas con arborescencias basada en las etapas del algoritmo de Edmonds. Se propone una herramienta para la resolución de dicha clase de problemas tanto para los casos en que no existen uniones entre los agentes como para aquellos que sí las consideran.

Capítulo 1

Optimización y cooperación en problemas de árboles de mínimo coste

En un problema de árboles de mínimo coste (abreviadamente, *mcstp* por sus siglas en inglés) varios agentes situados en diferentes zonas geográficas están interesados en un servicio determinado, que será proporcionado por un proveedor común llamado fuente. Los agentes serán servidos a través de conexiones que conllevan algún coste. Sin embargo, no les importa si están conectados directa o indirectamente a la fuente. El objetivo es encontrar un árbol de mínimo coste de forma que todos los agentes estén conectados directa o indirectamente a la fuente al menor coste posible. Además resultará de gran interés estudiar posibles repartos entre los agentes del coste total asociado al árbol obtenido como solución al problema.

En este primer capítulo se hará una revisión de las principales definiciones y resultados relacionados con la optimización y la cooperación en los problemas de árboles de mínimo coste. Para ello, se hará uso de la notación habitual utilizada en la literatura relacionada con este tipo de problemas. En particular, se tendrá en cuenta el trabajo de Bergantiños y Vidal-Puga (2021), donde se hace una excelente recopilación de los resultados relacionados con el enfoque cooperativo de los problemas de árboles de mínimo coste.

1.1. Introducción

Para comenzar se presentan una serie de definiciones propias de la teoría de grafos necesarias para contextualizar el problema de los árboles de mínimo coste.

Un grafo se define a partir de un conjunto de nodos y un conjunto de arcos que los unen. En nuestro caso, los nodos o vértices simbolizan a los agentes. Los arcos o aristas son las líneas presentes entre aquellos nodos en que existe alguna conexión. Los arcos pueden ser orientados si solamente permiten ir de un nodo origen a otro nodo final, o no orientados si permiten ir de un nodo cualquiera a otro indistintamente. A continuación, introduciremos la definición formal de grafo.

Definición 1.1. *Un grafo no dirigido G es un par (V, A) donde:*

- *V es el conjunto de nodos del grafo.*
- *A es el conjunto de arcos no orientados del grafo, es decir, $A = \{(i, j) : i, j \in V\}$.*

Nótese que en el caso de los grafos no dirigidos se tiene que $(i,j) = (j,i)$.

Un grafo tiene pesos si a cada arco se le asigna un valor que puede representar distancias, longitudes, costes, beneficios, flujos, etc. En este trabajo nos ocuparemos exclusivamente de grafos conexos y con pesos que representarán el coste de conexión entre nodos.

G^V indica el conjunto de todos los grafos de V. Como más adelante se estudiará la cooperación en los problemas de árboles y arborescencias de mínimo coste, sustituiremos la notación habitual de teoría de grafos V por N_0, en donde N es el conjunto de agentes que se conectan a la fuente y 0 la fuente o proveedor.

Un camino en un grafo es una sucesión de arcos de manera que, a excepción del primero, el nodo de entrada de un arco es el nodo de salida del siguiente arco.

Definición 1.2. *Dados* G *e* $i,j \in N_0$ *de tal modo que* $i \neq j$*, un camino de* i *a* j *en* G, G_{ij}*, es una secuencia de arcos* $\{(i_{k-1},i_k)\}_{k=1}^{K}$ *tales que* $(i_{k-1},i_k) \in G$ *para todo* $k \in \{1,2,\ldots,K\}$*,* $i_0 = i$ *y* $i_K = j$*. En caso de que* $i_0 = i_K$ *se tiene un ciclo.*

Definición 1.3. *Un árbol* t *es un grafo donde existe un único camino desde* i *a la fuente para todo* $i \in N$*.*

Si t es un árbol, normalmente se escribe $t = \{(i_0,i)\}_{i\in N}$, donde i_0 representa el primer nodo en el camino único en t de i hasta la fuente. El conjunto de todos los árboles será G_0^N.

Definición 1.4. *Un problema de árboles de mínimo coste es un par* (N_0, C)*, donde:*

- $N_0 = N \cup \{0\}$*, donde* $N = \{1,\ldots,n\}$ *el conjunto de agentes que se conectan a la fuente y* 0 *es la fuente o proveedor.*
- $C = (c_{ij})_{i,j\in N_0}$*, con* $c_{ij} = c_{ji} \geq 0$ *y* $c_{ii} = 0$*.*

Una matriz de costes $C = (c_{ij})_{i,j\in N_0}$ proporciona el coste de conexión directo entre dos nodos cualesquiera. En los problemas de árboles de mínimo coste asumimos que $c_{ii} = 0$ para todo $i \in N_0$ y $c_{ij} = c_{ji}$ para todo $i,j \in N_0$. Esto implica que se trabaja con arcos no dirigidos y que la matriz es simétrica. El conjunto de todas las matrices de coste sobre N_0 se denota como C^N. Dados $C, C' \in C^N$ se dice que $C \leq C'$ si $c_{ij} \leq c'_{ij}$ para todo $i,j \in N_0$.

La representación gráfica de un *mcstp* se realiza mediante un grafo, es decir, una red de nodos y arcos donde el nodo 0 representa a la fuente y los nodos restantes a los agentes y los arcos llevan asociado el coste de conexión entre agentes.

Un árbol de mínimo coste será aquel árbol cuyo coste asociado sea el menor de todos.

Definición 1.5. *Un árbol de mínimo coste (abreviadamente, mt por sus siglas en inglés) para* (N_0, C) *es un árbol* t *para* N_0 *tal que* $c(t) = \min_{G\in G_0^N} \sum_{(i,j)\in G} c_{ij}$*.*

En un problema de árboles de mínimo coste no tiene por qué existir un único árbol de mínimo coste.

Dado un *mcstp* (N_0, C), se denota el coste asociado a cualquier *mt* como $m(N_0, C)$. Por tanto, la finalidad al resolver un *mcstp* es encontrar un *mt* de forma que todos los agentes se conecten a la fuente directa o indirectamente de manera que el coste total de las conexiones sea el mínimo posible.

A continuación se presenta el ejemplo que sirve de base y que se irá modificando para poder adaptarse y ser utilizado en las diferentes aplicaciones prácticas a lo largo del capítulo.

Ejemplo 1.1. *Para (N_0, C) con $N = \{1, 2, 3\}$ y matriz de costes C, se presenta el grafo asociado y el árbol de mínimo coste:*

$$C = \begin{pmatrix} 0 & 4 & 9 & 8 \\ 4 & 0 & 7 & 10 \\ 9 & 7 & 0 & 1 \\ 8 & 10 & 1 & 0 \end{pmatrix}$$

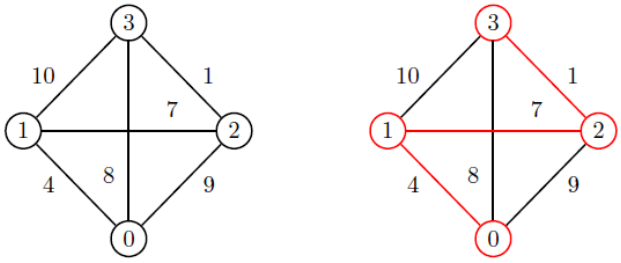


Figura 1.1: Matriz de costes, grafo asociado y árbol de mínimo coste.

Como se ha comentado previamente, en el caso de que haya arcos con el mismo coste, puede suceder que no exista una única solución y haya más de un árbol de mínimo coste. A continuación se muestra un caso con más de un árbol de mínimo coste como solución al problema.

Ejemplo 1.2. *Problema con más de un árbol de mínimo coste:*

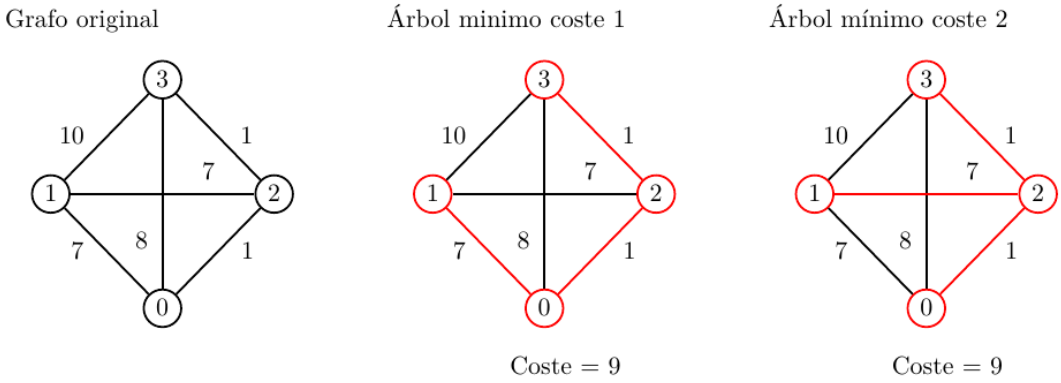


Figura 1.2: Problema con más de un árbol de mínimo coste.

1.2. Algoritmos de búsqueda

Como quedó expuesto en el apartado anterior, el primer objetivo de los *mcstp* es el de obtener un árbol de mínimo coste que conecte todos los agentes a la fuente. Para ello existen multiplicidad de algoritmos, pero en este trabajo nos centraremos en los tres más extendidos y utilizados, el algoritmo de Prim, el de Kruskal y el de Borůvka.

Algoritmo de Prim

El algoritmo de Prim fue dado a conocer por Robert C. Prim en el año 1957. Lo que propone el algoritmo esencialmente es lo siguiente:

- Se marca un nodo cualquiera, que será el nodo de partida.
- Se selecciona el arco de menor valor que contenga un único nodo marcado, en caso de empate se elige uno cualquiera. Se marca el otro nodo del arco.

- Se repite el paso 2 siempre que el arco elegido enlace un nodo marcado y otro que no lo esté.
- Se finaliza cuando todos los nodos estén marcados.

El algoritmo de Prim se define de la siguiente manera (Prim 1957):

Comenzamos con $S^0 = \{0\}$ y $G^0 = \emptyset$.

- **Etapa 1** Se toma un arco $(0, i)$ tal que $c_{0i} = \min_{j \in N} \{c_{0j}\}$. Si existen varios arcos que cumplen esta condición, se selecciona uno al azar. Ahora $S^1 = \{0, i\}$ y $G^1 = \{(0, i)\}$.
- **Etapa p+1 :** Se ha definido $S^p \subset N_0$ y $G^p \in G_0^N$. Ahora se define S^{p+1} y G^{p+1}. Se selecciona un arco (j, i), con $j \in S^p$ e $i \in N_0 \backslash S^p$, tal que $c_{ji} = \min_{k \in S^p, l \in N_0 \backslash S^p} \{c_{kl}\}$. Si hay varios arcos que cumplen esta condición, se selecciona uno al azar. Ahora $S^{p+1} = S^p \cup \{i\}$ y $G^p \cup \{(j, i)\}$.

Este proceso se completa en n etapas.

Ejemplo 1.3. *Aplicación etapa a etapa del algoritmo de Prim en el problema del ejemplo 1.1:*

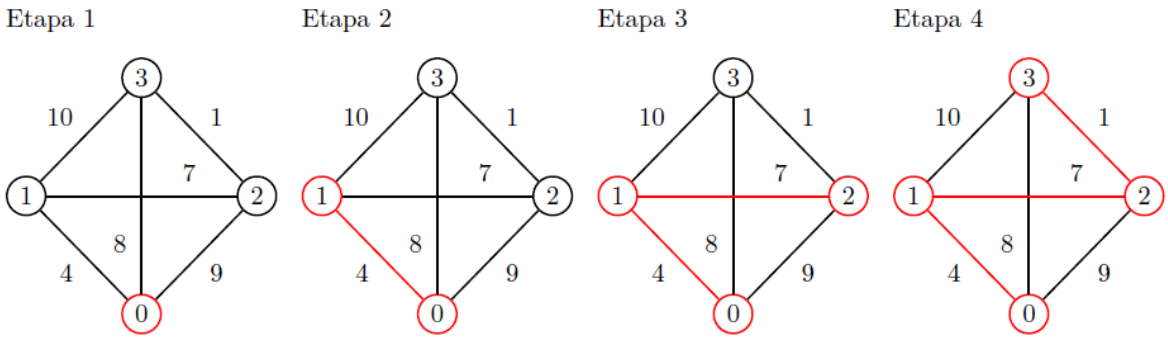


Figura 1.3: Etapas del algoritmo de Prim.

En la cuarta etapa del algoritmo de Prim se obtiene un árbol de mínimo coste formado por la consecución de los arcos (0,1)-(1,2)-(2,3).

Seguidamente se expone el algoritmo de Kruskal como otro de los posibles métodos para la obtención de un árbol de mínimo coste.

Algoritmo de Kruskal

La solución propuesta por Kruskal (1956) plantea que si un grafo conectado (finito) tiene un número real positivo adjunto a cada arco (la longitud del arco), y si estas longitudes son todas distintas, entonces entre los árboles de expansión del grafo solo habrá uno cuya suma de arcos es un mínimo, es decir, el árbol de expansión más corto del grafo es único. De manera general el algoritmo es el siguiente:

- Se marca el arco de menor valor, en caso de empate se elige uno al azar.
- De los arcos restantes se marca el de menor valor, en caso de empate se elige uno al azar.
- Se repite el paso 2 siempre que el arco elegido no forme un ciclo con los ya marcados.

- El proceso termina cuando tenemos todos los nodos de la red en alguno de los arcos marcados, es decir, si la red tiene n nodos, cuando se han marcado $n-1$ arcos.

Las etapas del algoritmo son las que siguen:

Comenzamos con $A^0(C) = \{(i,j) \mid i,j \in N_0, i \neq j\}$ y $G^0(C) = \emptyset$.

- **Etapa 1:** Sea $(i,j) \in A^0(C)$ un arco tal que $c_{ij} = \min\limits_{(k,l)\in A^0(C)} \{c_{kl}\}$. (Si hay varios arcos que satisfacen esta condición, se selecciona uno al azar). Tenemos que

  $\left(i^1(C), j^1(C)\right) = (i,j)$, $A^1(C) = A^0(C) \setminus \{(i,j)\}$, y $G^1(C) = \left\{\left(i^1(C), j^1(C)\right)\right\}$.

- **Etapa p+1:** Tenemos definidos los conjuntos $A^p(C)$ y $G^p(C)$. Sea $(i,j) \in A^p(C)$ un arco tal que $c_{ij} = \min\limits_{(k,l)\in A^p(C)} \{c_{kl}\}$. (Si hay varios arcos que satisfacen esta condición, se selecciona uno al azar). Hay dos posibles casos:

  1. $G^p(C) \cup \{(i,j)\}$ contiene un ciclo. Ir al principio de la etapa p+1. con $A^p(C) = A^p(C) \setminus \{(i,j)\}$ y $G^p(C)$ igual.
  2. $G^p(C) \cup \{(i,j)\}$ no contiene ciclos. Poner $\left(i^{p+1}(C), j^{p+1}(C)\right) = (i,j)$, $A^{p+1}(C) = A^p(C) \setminus \{(i,j)\}$, y $G^{p+1}(C) = G^p(C) \cup \left\{\left(i^{p+1}(C), j^{p+1}(C)\right)\right\}$. Ir a la etapa p+2.

La idea general de este algoritmo es semejante a la del algoritmo de Prim, pero en este caso se seleccionan arcos en lugar de nodos.

Ejemplo 1.4. *El algoritmo de Kruskal por etapas aplicado al problema del ejemplo 1.1:*

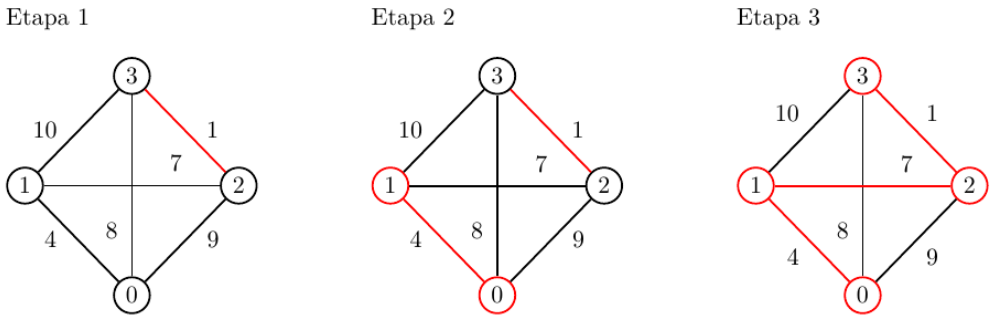


Figura 1.4: Etapas del algoritmo de Kruskal.

Siguiendo los pasos del algoritmo de Kruskal en la tercera etapa se obtiene el árbol de mínimo coste formado por los arcos (0,1)-(1,2)-(2,3).

Para finalizar esta sección dedicada a los algoritmos de búsqueda de árboles de mínimo coste se expondrá el algoritmo de Borůvka.

Algoritmo de Borůvka

El último de los algoritmos de búsqueda de árboles de mínimo coste expuesto en este trabajo es el de Borůvka (1926), que además de ser el primero en modelizar este tipo de problemas también propuso el primer algoritmo para la obtención de una solución. La idea general del algoritmo es la siguiente:

- Partimos de los nodos y consideramos que cada nodo es una única componente.

- Incluimos los arcos más baratos que salen de cada una de las componentes hacia otra componente, teniendo en cuenta que no deben formar ciclos.

- Repetimos el paso anterior hasta que todos los nodos pertenecen a una única componente.

A continuación se describen las diferentes etapas del algoritmo de Borůvka:

Dado un grafo $G \in G_0^N$, $\mathcal{P}(G) = \{\mathcal{P}_k(G)\}_{k=1}^{n(G)}$ denota la partición de N_0 en componentes conexas inducida por G. Formalmente, $\mathcal{P}(G)$ es la única partición de N_0 que satisface:

- Si $i, j \in \mathcal{P}(G)$, i y j están conectados en G.

- Si $i \in \mathcal{P}_k(G)$ y $j \in \mathcal{P}_l(G)$ con $k \neq l$, entonces i y j no están conectados en G.

Sea π una ordenación sobre el conjunto de todos los posibles arcos.

$$\pi : \{(i,j) : i,j \in N_0, i \neq j\} \longrightarrow \left\{1, 2, \ldots, \binom{|N|}{2}\right\}.$$

Y sea $\mathcal{P}(G^{\pi,s})$ el conjunto de componentes conexas en la etapa $s+1$, $s = 0, 1, ...$

- **Etapa 1:** Sea $G^{\pi,0} = \emptyset$. Tener en cuenta que $\mathcal{P}(G^{\pi,0}) = \{\{0\}, \{1\}, \ldots, \{|N|\}\}$. Se asume que se ha alcanzado la etapa s (s=1,2,...) y que se ha definido $G^{\pi,s-1}$.

- **Etapa *s*:** Para cada $T \in \mathcal{P}(G^{\pi,s-1})$, $0 \notin T$, sea $(i^{\pi,T}, j^{\pi,T}) \in T \times (N_0 \backslash T)$ tal que $c_{i^{\pi,T} j^{\pi,T}} = \text{mín}\{c_{ij} : i \in T, j \in N_0 \backslash T\}$. En caso de que haya más de un posible arco, se selecciona el que ocupe la posición más baja en el orden π.

$$G^{\pi,s} = G^{\pi,s-1} \cup \left\{\left(i^{\pi,T}, j^{\pi,T}\right) : T \in \mathcal{P}\left(G^{\pi,s-1}\right), 0 \notin T\right\}.$$

Se sabe que si para cada $T \in \mathcal{P}\left(G^{\pi,s-1}\right)$ el arco $\left(i^{\pi,T}, j^{\pi,T}\right)$ se elige a través de π, entonces $G^{\pi,s}$ es un grafo sin ciclos.

Si $\mathcal{P}(G^s) = \{N_0\}$, entonces $G^{\pi,s}$ es un árbol y el proceso termina. Si $\mathcal{P}(G^{\pi,s}) \neq \{N_0\}$, entonces hay que pasar a la etapa s+1.

El proceso termina en un número finito de etapas. El árbol obtenido mediante este procedimiento se denota por t^π. Es sabido que para cada orden π, t^π es un árbol de mínimo coste. Es más, dado un árbol de mínimo coste t, existe un orden π' tal que $t^{\pi'}$ coincide con t. Es posible que $t^\pi = t^{\pi'}$, incluso si π y π' son de diferente orden. Por ejemplo, si todos los costes son diferentes, $t^\pi = t^{\pi'}$ para todo π y π'.

Ejemplo 1.5. *Pasos a seguir para obtener una solución al problema del ejemplo 1.1 según el algoritmo de Borůvka:*

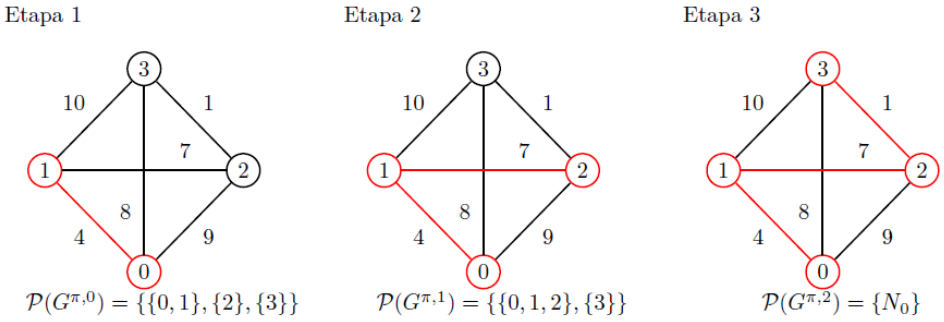


Figura 1.5: Etapas del algoritmo de Borůvka.

Al aplicar el algoritmo de Borůvka se obtiene el árbol de mínimo coste en la tercera etapa, de forma que el resultado obtenido coincide con los otros algoritmos. Esto es así porque en este caso existe un único árbol de mínimo coste, de existir más de uno, la solución aportada por cada algoritmo podría diferir en función del mismo.

1.3. Los juegos cooperativos y la forma irreducible

La Teoría de Juegos estudia problemas de decisión en los que interaccionan varios agentes decisores. El punto de partida para la Teoría de Juegos fue la publicación del tratado Theory of Games and Economic Behaviour de von Neumann y Morgenstern (1944). Todo el trabajo posterior que se ha llevado a cabo en Teoría de Juegos está fuertemente influenciado por esta obra, en la que se definen las bases de lo que hoy en día es conocida como Teoría de Juegos clásica. En los juegos cooperativos, los jugadores disponen de mecanismos que les permiten tomar acuerdos vinculantes previos al juego. Los jugadores pueden cooperar formando coaliciones con el fin de obtener mayores beneficios. El problema central es el reparto de beneficios entre los jugadores que forman la coalición. El objetivo principal de la Teoría de Juegos Cooperativos es analizar la importancia o influencia que ha tenido cada jugador en la obtención de ese beneficio, para proponer un reparto de beneficios adecuado.

Los juegos cooperativos se caracterizan por el hecho de que los jugadores pueden cooperar entre ellos para buscar un beneficio común. En el momento en que varios jugadores deciden cooperar en algún sentido, debe formarse una coalición entre estos jugadores. Los jugadores de esta coalición actuarán buscando el máximo beneficio posible para la coalición. Una coalición puede estar formada por cualquier grupo de jugadores de cualquier tamaño. Los beneficios que la coalición obtendrá serán en función de la propia coalición y deberán ser repartidos al finalizar el juego. Cuando cualquier reparto del pago entre los jugadores es posible, hablamos de un juego con utilidad transferible (juego TU por sus siglas en inglés). En este trabajo nos limitaremos a esta clase de juegos.

Definición 1.6. *Un juego cooperativo con utilidad transferible, juego TU, es un par* (N, v) *donde:*

- N=*{1,. . . , n} es el conjunto de jugadores.*
- v *es la función característica, que vincula con cada coalición* $S \subset N$ *un número real v(S) que representa el valor de la coalición S. Además, se satisface que* $v(\emptyset) = 0$.

Lo habitual es asociar un juego TU a cada problema para a continuación obtener una solución para dicho juego TU y por tanto para el problema original asociado. Bird (1976) asocia un juego TU a (N, v_C) con cada *mcstp* (N_0, C). Para cada coalición $S \subset N : v_C(S) = m(S_0, C)$.

En el caso de los *mcstp* se pueden considerar dos tipos de juegos, el juego pesimista (Bird 1976) y el juego optimista (Bergantiños y Vidal-Puga 2007a).

Juego Pesimista

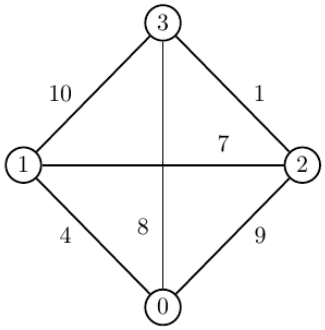
El juego pesimista fue formulado por Bird (1976) y en él cada coalición obtiene el mínimo coste de conexión asumiendo que los agentes que no pertenecen a la coalición no están presentes.

Definición 1.7. *El juego cooeprativo pesimista para los problemas de árboles de mínimo coste se define como:*

$$(N, v_C) \quad tal\ que \quad v_C(S) = m\,(S_0, C) \quad para\ toda\ coalición \quad S \subset N.$$

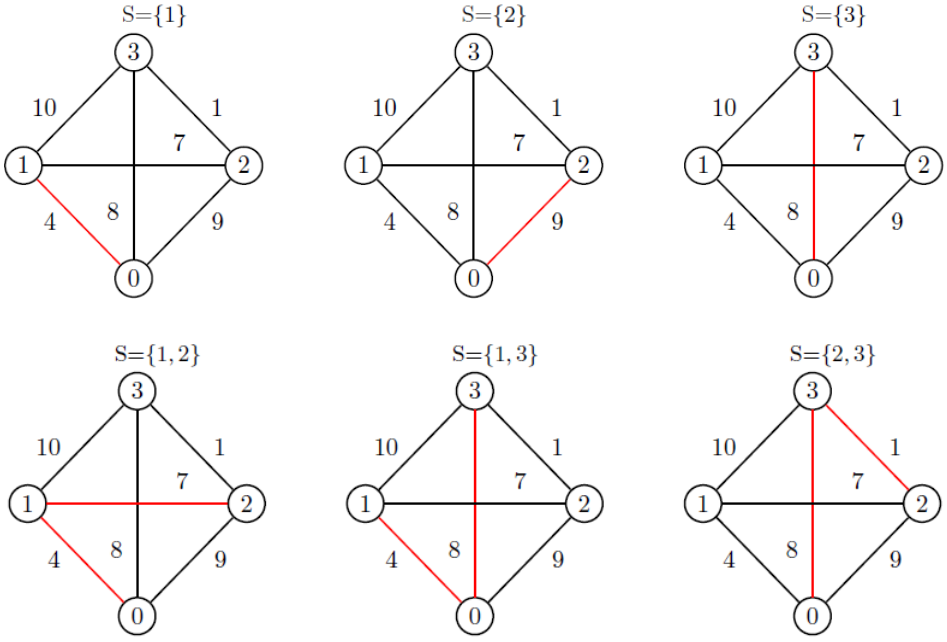
En este caso el valor de una coalición se limita al mejor valor que puedan lograr por sí mismos, sin la contribución de otros jugadores ajenos a la propia coalición.

Ejemplo 1.6. *Vamos a obtener el juego pesimista asociado al problema del ejemplo 1.1:*



Coalición	Juego pesimista
{1}	4
{2}	9
{3}	8
{1,2}	11
{1,3}	12
{2,3}	9
{N}	12

Además se presentan a continuación los grafos asociados al juego pesimista recién calculado para el ejemplo 1.1:



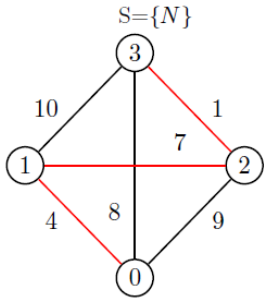


Figura 1.6: Juego pesimista para cada una de las coaliciones.

Juego Optimista

En el juego optimista definido por Bergantiños y Vidal-Puga (2007a), el coste de conexión de cada coalición se obtiene asumiendo que el resto de agentes no pertenecientes a la coalición ya están conectados a la fuente y que la conexión a través de ellos es posible sin que implique coste alguno.

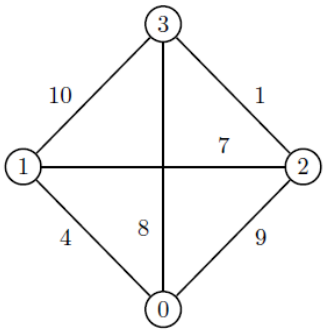
Definición 1.8. *El juego cooperativo optimista para los problemas de árboles de mínimo coste se define como:*

$$(N, v_C^+) \quad tal\ que \quad v_C^+(S) = m\left(S_0, C^{+(N\setminus S)}\right) \quad para\ todo \quad S \subset N$$

$$donde \quad c_{ij}^{+(N\setminus S)} = c_{ij} \quad para\ todos\ los\ agentes \quad i,j \in S$$

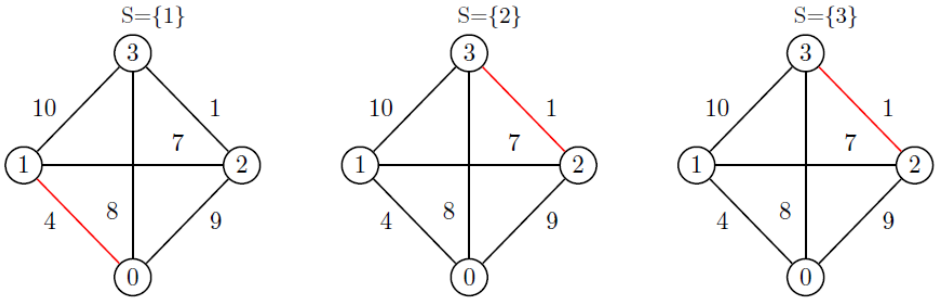
$$y \quad c_{0i}^{+(N\setminus S)} = min_{j\in(N\setminus S)_0} c_{ji} \quad para\ todo \quad i \in S.$$

Ejemplo 1.7. *Cálculo del juego optimista asociado al problema del ejemplo 1.1:*



Coalición	Juego optimista
{1}	4
{2}	1
{3}	1
{1,2}	5
{1,3}	5
{2,3}	8
{N}	12

En este caso también se muestra el grafo para cada una de las posibles coaliciones del juego optimista recién calculado.



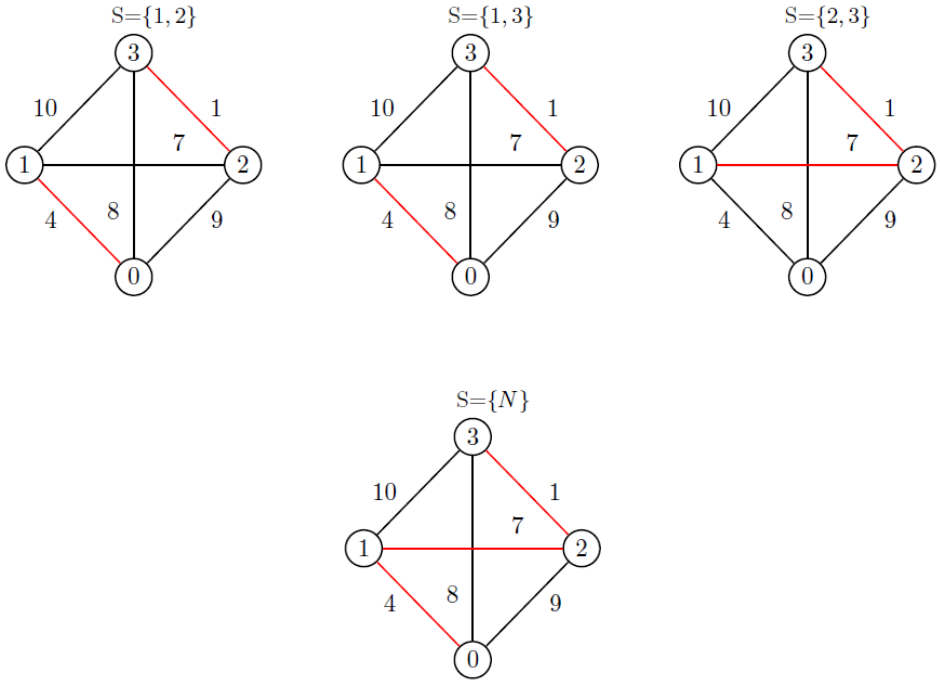


Figura 1.7: Juego optimista para cada una de las coaliciones.

A la hora de calcular el juego optimista hay que recordar que los agentes que se quieren conectar a la fuente no están obligados a conectarse directamente a la misma o mediante agentes de su propia coalición, dado que se asume que los agentes restantes ya están conectados a la fuente. Por tanto, si por ejemplo nos centramos en la coalición $S = \{2\}$, el agente 2 no ha de limitarse necesariamente a conectarse a la fuente mediante el arco (2,0) de coste 9; le sale más rentable conectarse al agente 3 a través del arco (2,3) de coste 1, al presuponerse que el agente 3 ya está conectado a la fuente. Si ahora observamos la coalición $S = \{1, 2\}$ vemos que al agente 1 le compensa conectarse directamente a la fuente, ya que el arco (0,1) es el de menor coste para dicho agente, mientras que para el agente 2 sigue resultando ventajoso el conectarse por medio del agente 3.

Forma irreducible de un problema de árboles de mínimo coste

En Bird (1976) se puede encontrar la definición de grafo minimal, noción previa y necesaria para la formulación de la forma irreducible. Dado un problema de árboles de mínimo coste (N_0, C) y un árbol de mínimo coste t, el grafo minimal (N_0, C^t) se obtiene reduciendo el coste de un arco (i, j) al máximo coste del arco que hay en el único camino de t que une los dos nodos del arco (i, j).

La forma irreducible de un problema de árboles de mínimo coste (N_0, C) se define como el grafo minimal $(N_0, C^*) = (N_0, C^t)$ asociado con uno de sus árboles de mínimo coste t (Bergantiños y Vidal-Puga, 2007a), de tal modo que no depende del árbol de mínimo coste que se esté considerando.

Si (N_0, C^*) es una forma irreducible, decimos que C^* es una matriz irreducible. Es más, $C^* \leq C$. Una matriz es irreducible si al reducir el coste de cualquier arco, el coste de conectar los agentes a la fuente también se reduce.

(N_0, C^*) es irreducible si y sólo si existe un árbol t en (N_0, C^*) que satisface las siguientes condiciones:

1. $t = \{(\pi_{s-1}, \pi_s)\}$ donde $\pi_0 = 0$ (la fuente).

2. Dado $\pi_p, \pi_q \in N_0$ con $p < q, c^*_{\pi_p \pi_q} = \text{máx}_{s|p<s\leq q} \left\{c_{\pi_{s-1}\pi_s}\right\}$.

Además t es un árbol minimal.

Ejemplo 1.8. *Forma y matriz irreducibles para el problema del árbol de mínimo coste del ejemplo 1.1:*

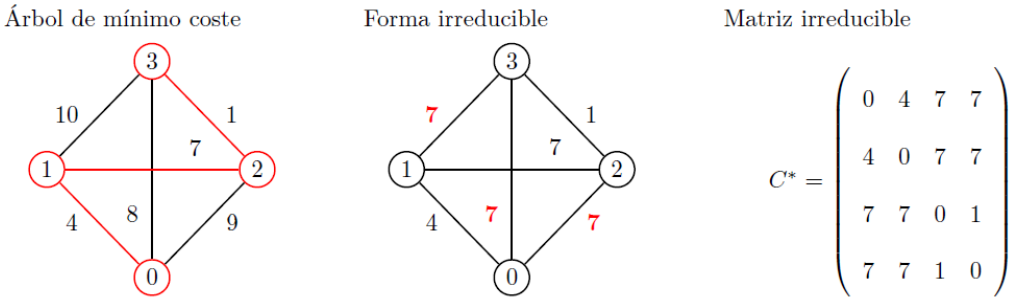


Figura 1.8: Árbol de mínimo coste y su forma y matriz irreducibles.

1.4. Reglas de reparto

Una vez que se conoce cuál es el coste correspondiente al árbol de mínimo coste $m(N_0, C)$, resulta interesante conocer de qué modo se reparte dicho coste entre los agentes implicados. En este trabajo se expondrán dos de las reglas de reparto de costes más utilizadas y extendidas, la regla de Bird y la regla Folk.

Definición 1.9. *Una regla de asignación de costes es una función ψ que asigna a cada problema de árboles de mínimo coste (N_0, C) el vector $\psi(N_0, C) \in \mathbb{R}^N$ tal que*

$$\sum_{i \in N} \psi_i(N_0, C) = m(N_0, C),$$

donde $\psi_i(N_0, C)$ es el coste asignado al agente i.

1.4.1. La regla de Bird

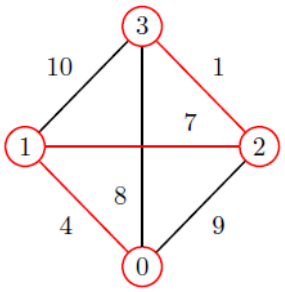
La regla de Bird (1976) se calcula siguiendo el algoritmo de Prim, por tanto el reparto de costes se realizará en función del árbol de mínimo coste seleccionado por dicho algoritmo. De acuerdo a esta regla, los agentes se conectan secuencialmente a la fuente según las etapas del algoritmo de Prim y cada agente paga su correspondiente coste de conexión al predecesor inmediato.

Definición 1.10. *Dado un problema de árboles de mínimo coste (N_0, C), la regla de Bird para la asignación de costes:*

$$B_i(N_0, C) = c_{i^0 i},$$

donde i^0 representa el predecesor inmediato de i.

Ejemplo 1.9. *Siguiendo la regla de Bird, el reparto de los costes asociados al mcstp expuesto en el ejemplo 1.1 sería el siguiente:*



$$B_1(N_0, C) = c_{01} = 4$$
$$B_2(N_0, C) = c_{12} = 7$$
$$B_3(N_0, C) = c_{23} = 1$$
$$B(N_0, C) = (4, 7, 1)$$

Figura 1.9: Árbol de mínimo coste y reparto según la regla de Bird.

1.4.2. La regla Folk

En esta sección se expondrá la regla Folk estudiada por Bergantiños y Vidal-Puga (2007a, 2007b, 2021). Esta regla coincide con la conocida regla ERO por sus siglas en inglés de *Equal Remaining Obligation*, definida por primera vez en Feltkamp et al. (1994). Dicha regla se puede calcular de distintas maneras, aquí se exponen cuatro de ellas. Las dos primeras están basadas en los juegos cooperativos y las dos últimas se apoyan en algoritmos polinomiales. Antes de plantear los dos métodos fundamentados en los juegos cooperativos, resulta pertinente introducir el valor de Shapley para facilitar su futura exposición.

Valor de Shapley

El valor de Shapley (Shapley 1953) propone un reparto imparcial de los costes generados por la cooperación entre los jugadores. El pago que asigna a cada jugador es una medida ponderada de las contribuciones marginales de dicho jugador a las coaliciones a las que pertenece.

Definición 1.11. *Dado un juego TU (N, v), el valor de Shapley viene dado por:*

$$Sh(N, v) = \frac{1}{\mid \Pi^N \mid} \sum_{\pi \in \Pi^N} m^\pi(N, v), \text{ donde}$$

$m_i^\pi(N, v) = v(Pre(i, \pi) \cup \{i\}) - v(Pre(i, \pi)) \forall i \in N.$

Definición 1.12. *Dado un problema de árboles de mínimo coste, la regla Folk para la asignación de costes se define como:*

$$Folk(N_0, C) = Sh(N, v_C^+) = Sh(N, v_{C^*}).$$

La regla Folk coincide con el valor de Shapley para el juego optimista del problema y con el juego pesimista asociado a la forma irreducible del problema. A continuación veremos cómo se aplica el valor de Shapley a dichos juegos cooperativos mediante su aplicación a un problema de ejemplo.

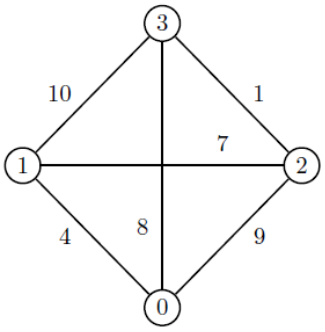
Valor de Shapley para el juego optimista

La primera de las opciones posibles para la obtención de la regla Folk es mediante el cómputo del valor de Shapley del juego optimista.

Definición 1.13. *La regla Folk para un problema de árboles de mínimo coste se define como el valor de Shapley para el juego optimista:*

$$Folk(N_0, C) = Sh(N, v_C^+).$$

Ejemplo 1.10. *En el ejemplo 1.7 expuesto previamente ya está calculado el juego optimista, de modo que nos basaremos en dichos resultados para calcular la regla Folk mediante el valor de Shapley para ese juego:*



Orden	Jugador 1	Jugador 2	Jugador 3
123	4	1	7
132	4	7	1
213	4	1	7
231	4	1	7
312	4	7	1
321	4	7	1
Total	24	24	24

$$Folk(N_0, C) = (4, 4, 4)$$

Valor de Shapley para el juego pesimista irreducible

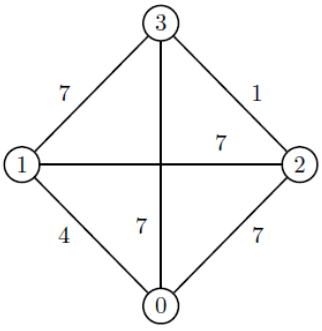
Otro modo de obtener la regla Folk es mediante el cálculo del valor de Shapley del juego pesimista para la forma irreducible.

Definición 1.14. *La regla Folk para un problema de árboles de mínimo coste se define como el valor de Shapley del juego pesimista para la forma irreducible:*

$$Folk(N_0, C) = Sh(N, v_{C^*}).$$

Es posible aplicar el juego pesimista definido por Bird a la forma irreducible del problema en lugar de aplicarla al problema original.

Ejemplo 1.11. *Como la forma irreducible ya está calculada en el ejemplo 1.8, se procede directamente a calcular el valor de Shapley para el juego pesimista de dicha forma irreducible, de forma que se obtiene el siguiente reparto:*



Coalición	Juego pesimista
{1}	4
{2}	7
{3}	7
{1,2}	11
{1,3}	11
{2,3}	8
{N}	12

Seguidamente se muestran los grafos correspondientes a cada una de las posibles coaliciones del ejemplo 1.8 siguiendo el juego pesimista para la forma irreducible.

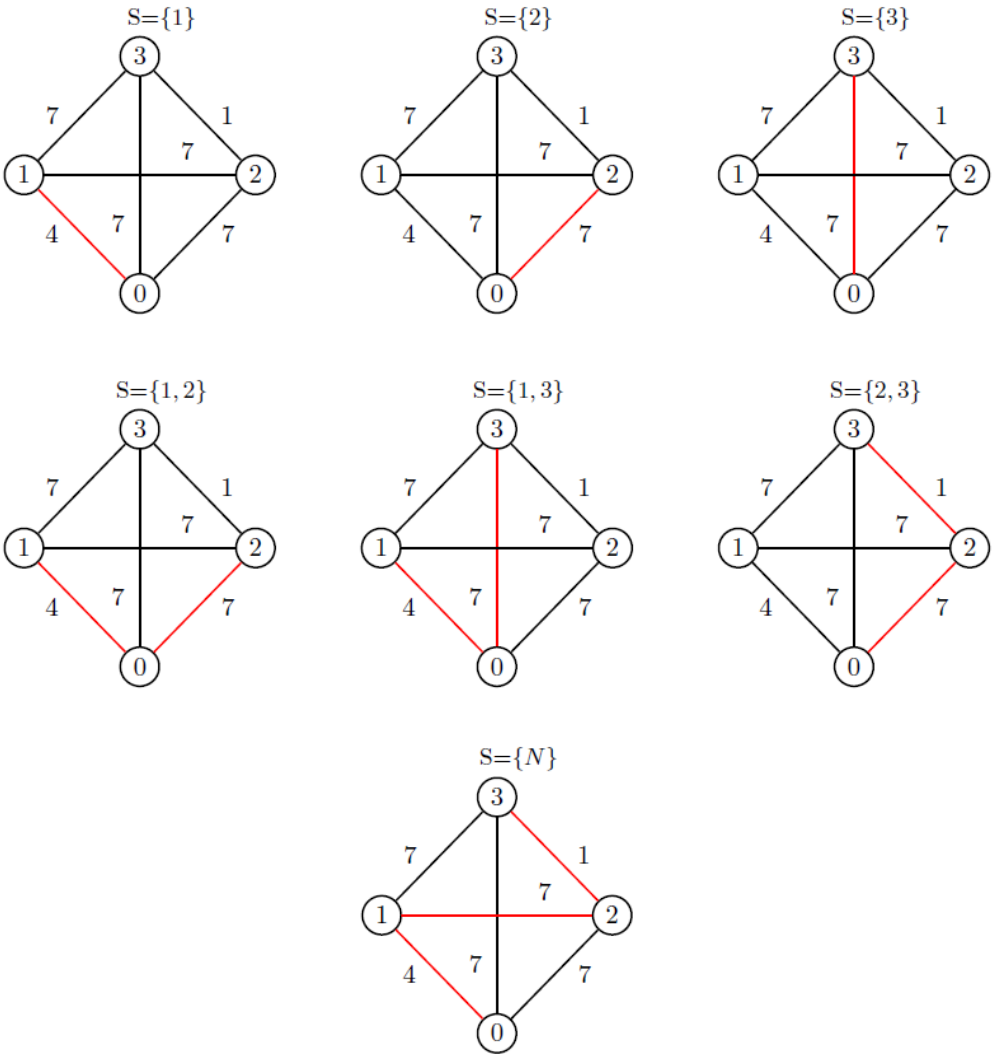


Figura 1.10: Juego pesimista de la forma irreducible para cada coalición.

Hay que tener en cuenta que la obtención del reparto de costes mediante el valor de Shapley resulta engorroso en cuanto aumenta el número de jugadores. Para salvar esta limitación será aconsejable calcular la regla Folk haciendo uso de algoritmos polinomiales que se enfrenten con mayor solvencia a conjuntos grandes de jugadores. Se expondrán dos procedimientos, el primero de ellos se auxiliará en el algoritmo de búsqueda de árboles de mínimo coste de Kruskal y el segundo será el método conocido como "pintar tramos de carretera".

Folk con las etapas de algoritmo de Kruskal

Con este procedimiento, introducido por primera por Feltkamp et al. (1994), se siguen las etapas del algoritmo de Kruskal para el reparto de los costes. Los costes de conexión en las diferentes etapas se reparten en función de las obligaciones de cada uno de los agentes. Los pasos a seguir son los siguientes:

- El coste del arco seleccionado en cada etapa del algoritmo de Kruskal se reparte entre los agentes que se benefician de la construcción del arco.

- En cada etapa del algoritmo el coste de un arco se reparte teniendo en cuenta la obligación de cada agente en la etapa anterior menos la obligación del agente en esta etapa.

- En la etapa inicial (cuando los agentes están aislados) tienen una obligación de 1 unidad.

- En las siguientes etapas se descuenta de la obligación de la etapa anterior la obligación de la etapa actual, teniendo en cuenta que la obligación de 1 unidad se reparte a partes iguales entre los agentes conectados.

- Una vez que los agentes están conectados a la fuente su obligación es siempre 0.

Definición 1.15. *La regla Folk siguiendo las etapas del algoritmo de Kruskal:*

$$Folk_i(N_0, C) = \sum_{p=1}^{n} c_{i^p j^p} \left[O_i\left(S^{p-1}(i)\right) - O_i\left(S^p(i)\right) \right] \forall i \in N, donde$$

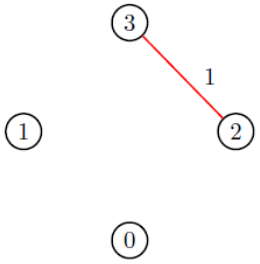
$S^P(i)$ es la componente conexa a la que pertenece el agente i en la etapa p del algoritmo,

y $O_i(S) = \begin{cases} \frac{1}{|S|}, 0 \notin S \\ 0 \in S \end{cases} \quad \forall i \in S \subset N_0.$

Ejemplo 1.12. *Para calcular la regla Folk nos serviremos del algoritmo de Kruskal ya aplicado en el ejemplo 1.4, de forma que aquí se mostrará únicamente el reparto de los costes siguiendo dicho algoritmo.*

Etapa 1

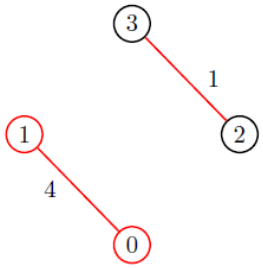
En la primera etapa se selecciona el arco (2,3) que beneficia a los agentes 2 y 3, por tanto el coste del arcos se divide entre ambos ($\frac{1}{2}$).



Agente	Obligaciones
1	1-1
2	1-$\frac{1}{2}$
3	1-$\frac{1}{2}$

Coste etapa $1 = 1 \cdot \left(0, \frac{1}{2}, \frac{1}{2}\right)$

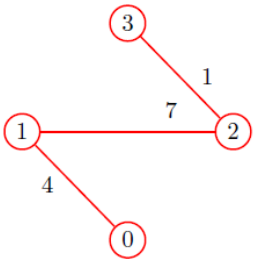
Etapa 2
En la segunda etapa del algoritmo se seleccionó el arco (0,1) que beneficia al agente 1, de modo que el coste del arco es asumido en su totalidad por dicho agente.



Agente	Obligaciones
1	1-0
2	$\frac{1}{2} - \frac{1}{2}$
3	$\frac{1}{2} - \frac{1}{2}$

Coste etapa 2 $= 4 \cdot (1, 0, 0)$

Etapa 3
En la tercera etapa finaliza el algoritmo con el arco (1,2) cuya construcción beneficia a los agentes 2 y 3, teniendo que repartir el coste del mismo entre ambos agentes ($\frac{7}{2}$).



Agente	Obligaciones
1	0-0
2	$\frac{1}{2}$-0
3	$\frac{1}{2}$-0

Coste etapa 3 $= 7 \cdot \left(0, \frac{1}{2}, \frac{1}{2}\right)$

$$Folk(N_0, C) = 1 \cdot \left(0, \frac{1}{2}, \frac{1}{2}\right) + 4 \cdot (1, 0, 0) + 7 \cdot \left(0, \frac{1}{2}, \frac{1}{2}\right) = (4, 4, 4)$$

Folk pintando tramos de carretera

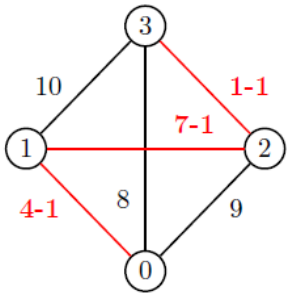
La idea en que se basa este método, definida de forma más general para problemas de árboles de coste fijo en Bergantiños et al. (2014), es la de gestionar el pintado de una carretera dividida en distintos tramos, cada uno de esos tramos representa los arcos del grafo. Para ello se escoge un árbol de mínimo coste y se asume que cada arco se corresponde con un tramo de carretera que hay que pintar. Los pasos a seguir son los siguientes:

- En cada arco hay una máquina que solo va a trabajar en ese arco.
- Todas las máquinas trabajan a la misma velocidad.
- Los agentes se dedican a pintar los tramos de carretera tan cerca de su residencia como sea posible.
- El agente termina de pintar en el momento en que el camino entre el agente y el punto central está pintado completamente.
- Los agentes asignados al mismo arco en la misma etapa se reparten el coste a partes iguales.
- Cada etapa termina en el momento que alguno de los arcos está completamente pintado.

- El algoritmo termina cuando todos los arcos del árbol están pintados.

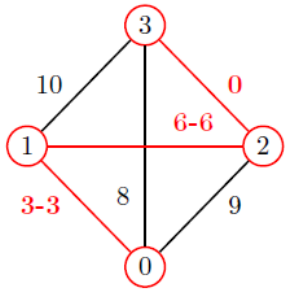
Ejemplo 1.13. *A continuación se muestra paso a paso cómo se resuelve el reparto de costes para el ejemplo 1.1 mediante el método de pintar tramos de carretera.*

Etapa 1
En la primera etapa el agente 1 pinta en el arco (0,1), el agente 2 se ocupa del arco (1,2) y el agente 3 del arco (2,3). El arco de menor coste es el (2,3) que tiene un valor de 1 y por tanto es el primero en completarse. Se resta ese coste de 1 a todos los arcos en los que se comenzó a trabajar.



Agente	Arco 1	Coste
1	(0,1)	1
2	(1,2)	1
3	(2,3)	1

Etapa 2
Al comenzar la etapa 2 los arcos que quedan por completar son el (0,1) con un coste de 3 y el arco (1,2) con un coste de 6. Los agentes 2 y 3 trabajan en el arco (1,2) y el agente 1 continúa en el (0,1). Ambos arcos se completan simultáneamente y se termina el proceso. El coste del arco (0,1) lo asume completamente el agente 1 y el coste del arco (1,2) se reparte equitativamente entre loso agentes 2 y 3.



Agente	Arco 1	Coste
1	(0,1)	3
2	(1,2)	3
3	(1,2)	3

$$Folk(N_0, C) = (1,1,1) + (3,3,3) = (4,4,4)$$

Después de haber aplicado diferentes métodos sobre el mismo ejemplo para calcular le regla Folk, se comprueba que independientemente del procedimiento a seguir, el resultado no varía y es siempre el mismo. Sin embargo sí que existe diferencia entre el reparto de costes propuesto por la regla de Bird y la regla Folk.

Capítulo 2

Optimización y cooperación en problemas de arborescencias de mínimo coste

En los problemas de arborescencias de mínimo coste, al igual que en los problemas de árboles de mínimo coste planteados en el capítulo anterior, varios agentes situados en diferentes zonas geográficas están interesados en un servicio determinado, que será proporcionado por un proveedor común, llamado fuente. A diferencia de los problemas de árboles de mínimo coste, puede suceder que los costes de conexión directa entre agentes no sean simétricos, es decir, que el coste de conexión de un agente i a otro agente j no tiene por qué ser el mismo que el coste de conexión del agente j al agente i. Esta situación de asimetría entre agentes puede estar motivada por distintas causas, como la diferencia de altitud, el sentido de la corriente fluvial, la capacidad asimétrica de la red, o incluso la intervención de reglamentos o leyes que restringen la circulación en alguno de los sentidos. Se comprueba de manera inmediata que los problemas de árboles de mínimo coste son problemas de arborescencias de mínimo coste en los que los costes de conexión directa entre agentes son simétricos.

Al igual que en el capítulo anterior, en este capítulo se hará una revisión de las principales definiciones y resultados relacionados con la optimización y la cooperación en los problemas de arborescencias de mínimo coste. Para ello, se hará uso de la notación habitual utilizada en la literatura relacionada con este tipo de problemas. En particular, se tendrán en cuenta los trabajos de Dutta y Mishra (2012) y Bahel y Trudeau (2017).

2.1. Introducción

A continuación procederemos a exponer una serie de términos relativos a la teoría de grafos que serán necesarios para complementar los expuestos en el capítulo anterior, de forma que sirvan de aclaración previa a los problemas de arborescencias de mínimo coste.

A diferencia de los problemas de árboles de mínimo coste, en los problemas de arborescencias de mínimo coste los arcos son orientados, por lo que nos encontramos ante grafos dirigidos. Otra característica que suele diferenciarlos es que los costes de conexión entre agentes no siempre van a ser simétricos, por lo que la matriz de costes C no tendría que ser simétrica, de forma que podría haber pares de nodos i y j tales que $c_{ij} \neq c_{ji}$.

Definición 2.1. *Un grafo dirigido D es un par (V, A) donde:*

- *V es el conjunto de nodos del grafo.*

- *A es el conjunto de arcos orientados del grafo, es decir, $A = \{(i,j) : i,j \in V\}$.*

Nótese que, aunque la definición de los grafos dirigidos es similar a la de los grafos no dirigidos, en este caso los arcos son orientados, lo que significa que $(i,j) \neq (j,i)$. El conjunto de todos los grafos dirigidos de V se denota por D^V.

Al igual que en los problemas de árboles de mínimo coste, dado que los nodos van a representar a la fuente y a un conjunto de agentes, a partir de este momento en lugar de V se va a utilizar N_0.

En un grafo dirigido un camino es una sucesión de arcos de forma que, exceptuando el primero de ellos que ha de ser la fuente 0, el nodo de entrada de un arco es el nodo de salida del siguiente arco.

Definición 2.2. *Dados D e $i,j \in N_0$ de tal modo que $i \leq j$, un camino de i a j en D, D_{ij}, es una secuencia de arcos $\{(i_{k-1}, i_k)\}_{k=1}^{K}$ tales que $(i_{k-1}, i_k) \in D$ para todo $k \in \{1,2,\ldots,K\}$, $i_0 = i$ y $i_K = j$.*

Se puede comprobar que todas estas definiciones son comunes y muy similares a las proporcionadas en el anterior capítulo dedicado a los problemas de árboles de mínimo coste.

Definición 2.3. *Una arborescencia a es un grafo dirigido y conexo que no forma ciclos, en el que exclusivamente existe un camino desde la fuente hacia el resto de nodos de tal forma que hay un único arco incidente en cada nodo, a excepción de la fuente que no tiene arcos incidentes. El conjunto de todas las arborescencias se denota por D_0^N.*

Definición 2.4. *Un problema de arborescencias de mínimo coste es un par (N_0, C), donde:*

- *$N_0 = N \cup \{0\}$, donde $N = \{1,\ldots,n\}$ el conjunto de agentes que se conectan a la fuente y 0 es la fuente o proveedor.*

- *$C = (c_{ij})_{i,j \in N_0}$, con $c_{ij} \geq 0$ y $c_{ii} = 0$.*

La matriz de costes $C = (c_{ij})_{ij \in N_0}$ proporciona el coste de conexión directo entre dos nodos cualesquiera. En los problemas de arborescencias de mínimo coste se asume que $c_{ii} = 0$ para todo $i \in N_0$ y c_{ij} no tiene por qué coincidir con c_{ji}. Esto hace referencia al hecho de trabajar con arcos dirigidos, por lo tanto la matriz será asimétrica. El conjunto de todas las matrices de coste sobre N_0 se denota como C^N. Dados $C, C' \in C^N$ se dice que $C \leq C'$ si $c_{ij} \leq c'_{ij}$ para todo $i,j \in N_0$.

Este tipo de problemas se puede representar por medio de un grafo dirigido, conexo y con pesos. El objetivo vuelve a ser nuevamente el mismo que en los problemas de árboles de mínimo coste, conectar todos los agentes directa o indirectamente a la fuente al menor coste posible. Por tanto, una arborescencia de mínimo coste será aquella arborescencia cuyo coste asociado sea el menor de todos los posibles.

Definición 2.5. *Una arborescencia de mínimo coste (abreviadamente, ma, por sus siglas en inglés) es una arborescencia a para N_0 tal que $c(a) = \min_{D \in D_0^N} \sum_{(i,j) \in D} c_{ij}$.*

Un problema de arborescencias de mínimo coste se puede representar mediante un grafo dirigido en el que el nodo 0 representa la fuente y el resto de nodos a los agentes. Los costes de conexión entre agentes se señalan en cada uno de los arcos, de forma que aquellos arcos no presentes en el grafo se asume que tienen un coste muy elevado. Modificamos el ejemplo expuesto hasta ahora para adaptarlo a un problema de arborescencias.

Ejemplo 2.1. *Para (N_0, C) donde $N = \{1, 2, 3\}$ es el conjunto de agentes y C la matriz de costes, se presenta el grafo asociado y la arborescencia de mínimo coste:*

$$C = \begin{pmatrix} 0 & 4 & 9 & 8 \\ - & 0 & 7 & 10 \\ - & 9 & 0 & 1 \\ - & 8 & 3 & 0 \end{pmatrix}$$

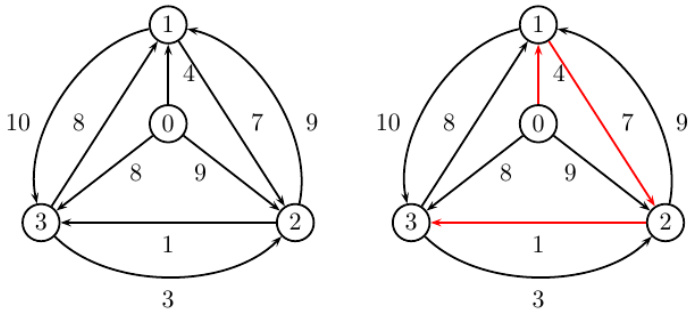


Figura 2.1: Matriz de costes, grafo asociado y arborescencia de mínimo coste.

Al igual que sucedía en los problemas de árboles de mínimo coste, es posible que haya más de una arborescencia en un mismo grafo.

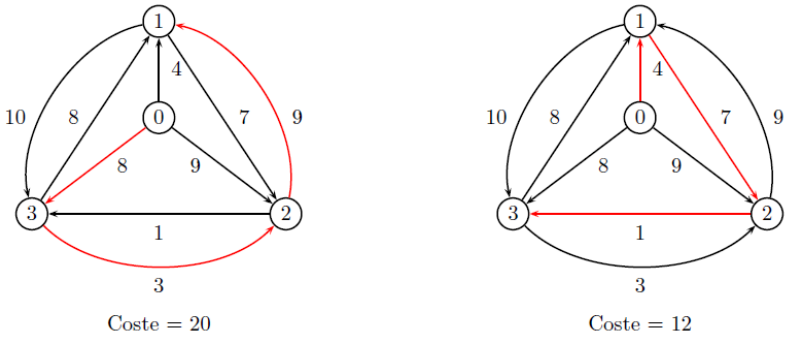


Figura 2.2: Arborescencia y arborescencia de mínimo coste.

De igual modo puede suceder que la arborescencia minimal no sea única.

Ejemplo 2.2. *Problema con más de una arborescencia de mínimo coste:*

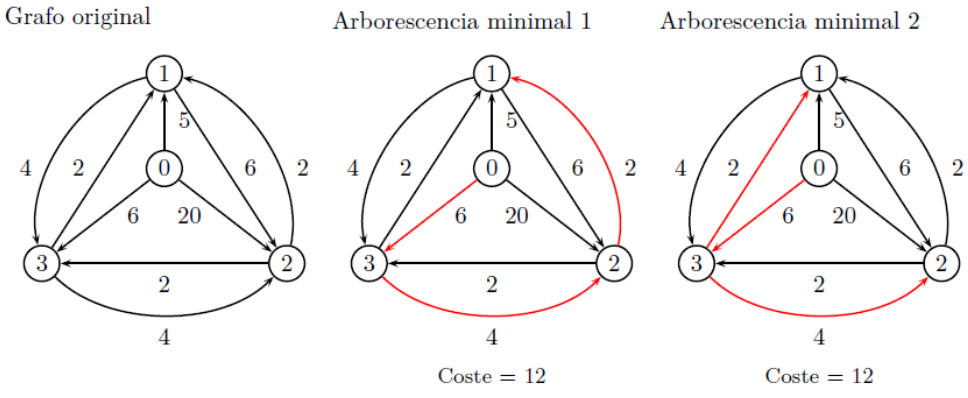


Figura 2.3: Problema con más de una arborescencia de mínimo coste.

Cuando existen pocos agentes implicados es factible resolver el problema de forma manual, pero a medida que este número aumenta resulta cada vez más engorroso, de manera que la aplicación de un algoritmo polinómico facilitará enormemente la tarea.

2.2. Algoritmo de búsqueda

Para la resolución de los problemas de arborescencias de mínimo coste se expondrá únicamente el conocido como algoritmo de Edmonds (1967), si bien los primeros en desarrollarlo fueron Chu y Liu (1965). El algoritmo se divide en dos fases diferenciadas, la primera de ellas procura encontrar una arborescencia de coste cero y la segunda traslada dicha arborescencia sobre el grafo original dando como resultado una arborescencia de mínimo coste. A continuación introducimos algunas definiciones necesarias para la correcta interpretación del algoritmo (Dutta y Mishra 2012):

Una matriz C es simple si el grafo obtenido al encontrar el arco incidente de menor coste para cada nodo da lugar a una arborescencia minimal.

Definición 2.6. *Una matriz C es simple si dado $p(i) \in \arg\min_{j\in N_0} c_{ji} \forall i \in N, \{p(i), i\}_i \in N$ es una arborescencia minimal.*

Definición 2.7. *Dada una matriz C, diremos que el conjunto de nodos $I = \{1, ..., K\}$ forman un C-ciclo si $c_{ii+1} = 0$ para todo $i = 1, ..., K-1$ y $c_{K1} = 0$.*

El algoritmo de Edmonds se desarrolla en las siguientes etapas según Dutta y Mishra (2012):

- **Etapa 0**. Para cada $j \in N$, sea $\delta_j^0 = \min_{i\in N_0} c_{ij}$. Si C es una matriz simple, sea $T = 0$ y terminar. Sea $N^0 \equiv \{N_1^0, \ldots, N_{n+1}^0\} \equiv \{\{0\}, \{1\}, \{2\}, \ldots, \{n\}$.

- **Etapa 1**. Se define C^1 de forma que $C_{ij}^1 \forall i \in N_0, \forall j \in N = c_{ij} - \delta_j^0$. Se construye una partición $\{N_1^1, \ldots, N_{K^1}^1\}$ de N_0 tal que cada $N_{k\in 1,\ldots,k^1}^1$ es bien un $C^1 - ciclo$ de elementos de N_0 o está aislado, con la restricción de que ningún conjunto de elementos aislados forme un $C^1 - ciclo$. Por definición, 0 está siempre aislado; y se escribe $N_1^1 \equiv \{0\}$. Para $k, l \in \{1, \ldots, K^1\}$ y $l \neq 1$, sea

$$\tilde{C}_{kl}^1 = \min_{i\in N_k^1, j\in N_l^1} C_{ij}^1.$$

$$\text{Para cada} \quad k = 2, \ldots, K^1, \quad \text{sea} \quad \delta_{N_k^1}^1 = \min_{j\in\{1,\ldots,K^1\}\setminus k} \tilde{C}_{N_j^1 N_k^1}^1.$$

$$\text{Si} \quad \tilde{C}^1 \quad \text{es una matriz simple, sea} \quad T = 1 \quad \text{y se termina.}$$

- **Etapa t**. Se define C^t de manera que $C_{N_k^{t-1} N_l^{t-1}}^t = \tilde{C}_{N_k^{t-1} N_l^{t-1}}^{t-1} - \delta_{N_l^{t-1}}^{t-1}$ para todo $k, l \in \{1, \ldots, K^{t-1}\}$. Se construye la partición $\{N_1^t, \ldots, N_{K^t}^t\}$ de N^{t-1} tal que cada N_k^t es bien un $C^t - ciclo$ de elementos de N^t o está aislado, con la restricción de que ningún terminal aislado forme un $C^t - ciclo$. Para $k, l \in \{1, \ldots, K^t\}$ y $l \neq 1$, sea

$$\tilde{C}_{kl}^t = \min_{i\in N_k^t, j\in N_l^t} C_{ij}^t.$$

$$\text{Para cada} \quad k = 2, \ldots, K^t, \quad \text{sea} \quad \delta_{N_k^t}^t = \min_{j\in\{1,\ldots,K^t\}\setminus k} \tilde{C}_{N_j^t N_k^t}^t.$$

$$\text{Si} \quad \tilde{C}^t \quad \text{es una matriz simple, sea} \quad T = t \quad \text{y se termina. Se pasa al paso} \quad t+1.$$

El procedimiento finaliza en n pasos, teniendo en cuenta que en cada etapa t bien se forma un C^t-ciclo o bien se finaliza. Al final, uno tiene que volver sobre sus pasos para obtener la arborescencia de mínimo coste asociado a la matriz c.

Este mismo algoritmo se puede explicar también de forma más sencilla e intuitiva de la manera que sigue:

- Para cada nodo distinto de la fuente se elige el arco incidente de menor coste. Se resta el coste de dicho arco a todos los arcos incidentes en dicho nodo. Para cada nodo distinto de la fuente escogemos un arco incidente de coste cero. Si los arcos seleccionados dan lugar a una arborescencia, se termina la primera fase y se pasa a la segunda fase. Si no dan lugar a una arborescencia, eso implica que se ha formado al menos un ciclo y es necesario continuar con el siguiente paso.

- Si existe algún ciclo se escoge uno al azar. Los nodos que dan lugar a ese ciclo se unen en un único nodo. Obtenemos así un nuevo grafo donde los costes entre el nuevo nodo y el resto de los nodos vienen dados por el coste mínimo entre los nodos del ciclo y cada uno de los nodos restantes.

- Repetimos los pasos anteriores tantas veces como sea necesario hasta encontrar una arborescencia.

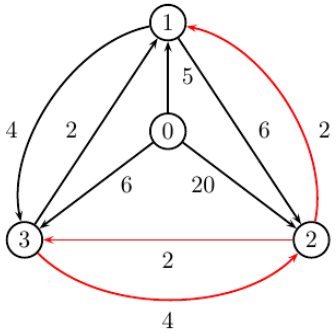
- Una vez se llega a la arborescencia comienza la segunda fase, de traslado al grafo original. Aquí habrá que volver sobre los pasos que nos llevaron a la arborescencia, teniendo en cuenta los arcos seleccionados y deshaciendo los ciclos. El resultado será una arborescencia de expansión de mínimo coste.

Ejemplo 2.3. *Aplicación del algoritmo de Edmonds para la obtención de una arborescencia de mínimo coste al ejemplo 2.2 expuesto previamente:*

Etapa 1

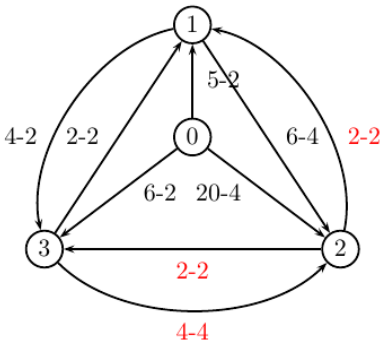
Se selecciona el arco de menor coste incidente en cada uno de los nodos, a excepción de la fuente. En caso de empate se selecciona uno de ellos evitando que se formen ciclos. En este caso los arcos $(2,1)$ y $(3,1)$ tienen coste de 2, se selecciona el arco $(2,1)$ de manera aleatoria.

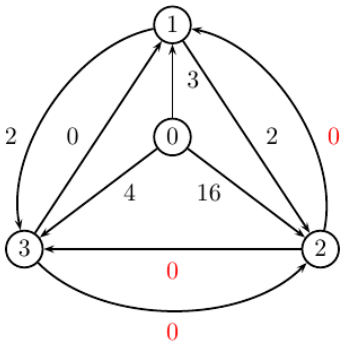
$$\begin{aligned}
\text{mín}\{c_{01}, c_{21}, c_{31}\} &= \text{mín}\{5, 2, 2\} = c_{21} = 2.\\
\text{mín}\{c_{02}, c_{12}, c_{32}\} &= \text{mín}\{20, 6, 4\} = c_{32} = 4.\\
\text{mín}\{c_{03}, c_{13}, c_{23}\} &= \text{mín}\{6, 4, 2\} = c_{23} = 2.
\end{aligned}$$



Etapa 2

Se resta el coste correspondiente a cada uno de los arcos seleccionados a todos aquellos arcos que compartan nodo de incidencia, incluyendo ellos mismos.

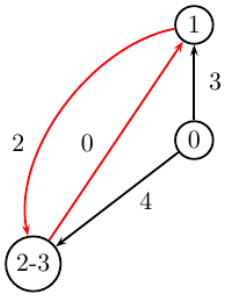




Etapa 3

Se ha creado un ciclo entre los nodos 2 y 3, formado por los arcos $\{(2,3)-(3,2)\}$. Ahora es necesario unir esos dos nodos en un supernodo y reformular el grafo y sus conexiones. Para ello es preciso localizar en el grafo anterior el arco de menor coste que una a los agentes localizados en distintos nodos. En este caso los arcos $(1,2)$ y $(1,3)$ comparten el mismo coste de 2, de modo que se selecciona el arco $(1,2)$ al azar. Por último se seleccionan aquellos arcos de menor coste que incidan en cada uno de los nodos.

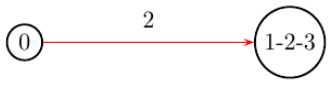
$$\begin{aligned}\text{mín}\{c_{02},c_{03}\}=\text{mín}\{16,4\}=c_{03}=4.\\ \text{mín}\{c_{12},c_{13}\}=\text{mín}\{2,2\}=c_{13}=2.\\ \text{mín}\{c_{21},c_{31}\}=\text{mín}\{0,0\}=c_{21}=0.\end{aligned}$$



Etapa 4

Se ha vuelto a crear un ciclo entre el nodo 1 y el supernodo $2-3$, formado por los arcos $\{(1,2-3),(2-3,1)\}$. Ahora es necesario unir esos dos nodos en otro supernodo $(1-2-3)$ y reformular el grafo y sus conexiones. Para ello es preciso localizar en el grafo anterior el arco de menor coste que una a los agentes localizados en distintos nodos. Por último se seleccionan aquellos arcos de menor coste que incidan en cada uno de los nodos.

$$\text{mín}\{c_{01},c_{023}\}=\text{mín}\{3,2\}=c_{023}=2.$$

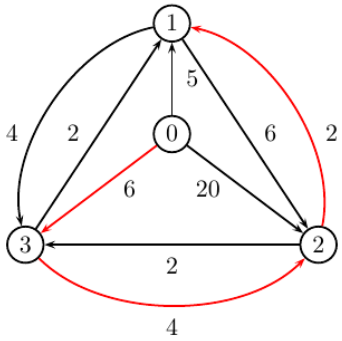


Etapa 5

De nuevo se restan los costes de los arcos seleccionados en la etapa anterior a todos los arcos que incidan en el mismo nodo. El resultado es la arborescencia $\{(0, 1-2-3)\}$ de coste 0. Aquí termina la primera fase del algoritmo de Edmonds.

Etapa 6

Para trasladar el resultado de la etapa anterior al grafo original hay que ir desandando los pasos llevados a cabo anteriormente para poder deshacer los supernodos. Para ello hay que comprobar qué arcos proporcionaron el menor coste de conexión para ambos supernodos, en este caso fueron los arcos $(0, 3)$ y $(2, 1)$.



Como se pudo comprobar en el ejemplo 2.2, existe más de una arborescencia minimal para este problema específico, pero con el algoritmo solamente se alcanza una de ellas, en función de los arcos seleccionados aleatoriamente en las diferentes etapas del algoritmo.

2.3. Los juegos cooperativos y las formas irreducibles

En el caso de los problemas de arborescencias de mínimo coste también es posible formular los juegos cooperativos explicados en el capítulo anterior relativo a los problemas de árboles de mínimo coste. En el desarrollo de este capítulo se expondrán los juegos cooperativos pesimista y optimista además de introducir las formas irreducibles del problema, conceptos necesarios posteriormente para presentar las reglas de reparto.

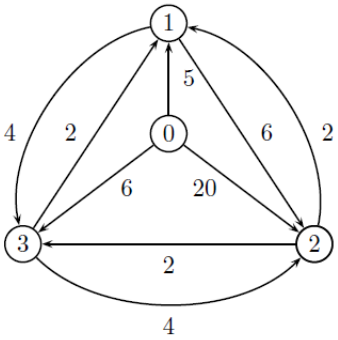
Juego Pesimista

Definición 2.8. *El juego cooperativo pesimista para los problemas de arborescencias de mínimo coste se define como el par* (N, v_C)*, donde*

$$v_c(S) = m(S_0, C) \quad \textit{para toda coalición} \quad S \subset N.$$

Cada coalición obtiene el mínimo coste de construcción de una arborescencia de coste mínimo para el problema (S_0, C), en la que hay un único camino de la fuente a cada nodo, asumiendo que los agentes externos a su propia coalición no están conectados a la fuente. Si los arcos no fueran dirigidos, este juego coincidiría con el juego pesimista de los problemas de árboles de mínimo coste.

Ejemplo 2.4. *El juego pesimista asociado al problema del ejemplo 2.2 es el siguiente:*



Coalición	Juego pesimista
{1}	5
{2}	20
{3}	6
{1,2}	11
{1,3}	8
{2,3}	10
{N}	12

Seguidamente se presenta el juego pesimista para cada una de las coaliciones posible siguiendo el problema del ejemplo 2.2:

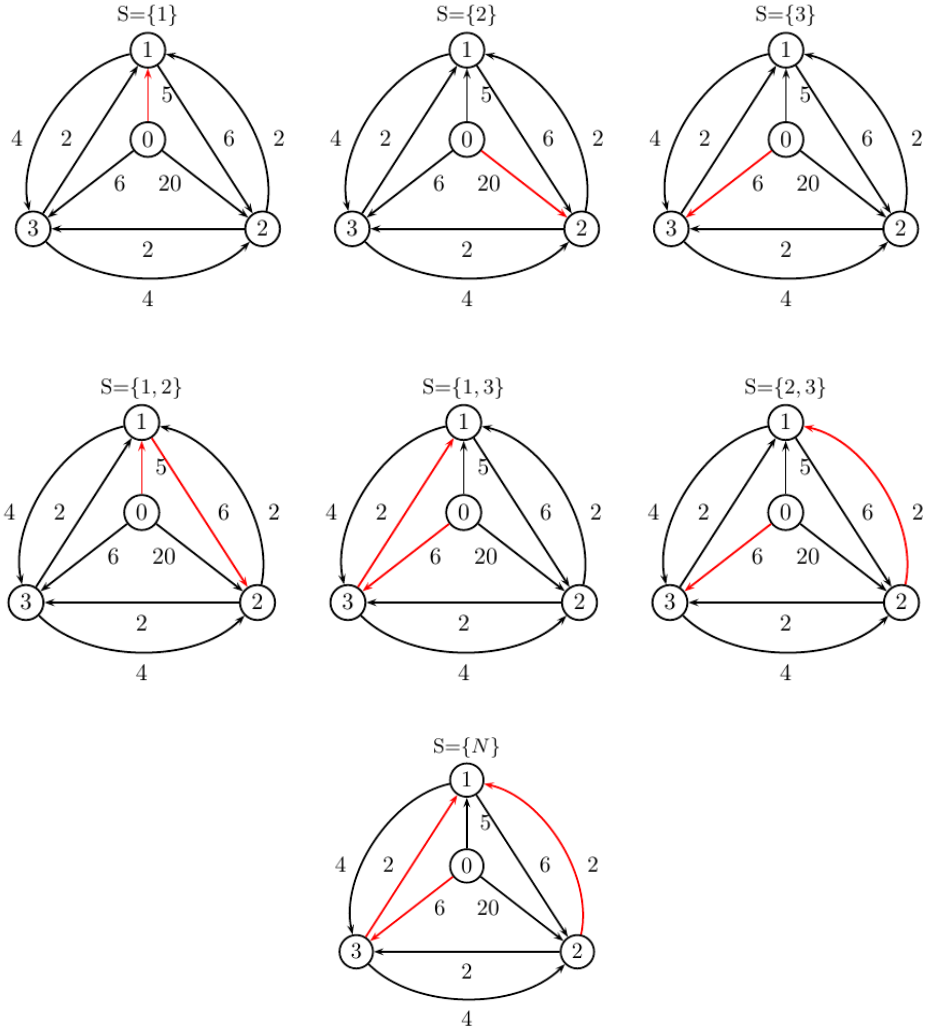


Figura 2.4: Juego pesimista para cada una de las coaliciones.

Juego Optimista

Definición 2.9. *El juego cooperativo optimista para los problemas de arborescencias de mínimo coste se define como:*

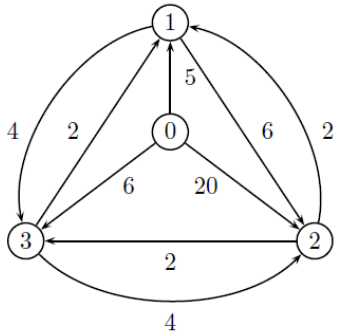
$$(N, v_C^+) \quad tal\ que \quad v_C^+(S) = m\left(S_0, C^{+(N\setminus S)}\right) \quad para\ todo \quad S \subset N$$

$$donde \quad c_{ij}^{+(N\setminus S)} = c_{ij} \quad para\ todos\ los\ agentes \quad i,j \in S$$

$$y \quad c_{0i}^{+(N\setminus S)} = min_{j\in(N\setminus S)_0} c_{ji} \quad para\ todo \quad i \in S.$$

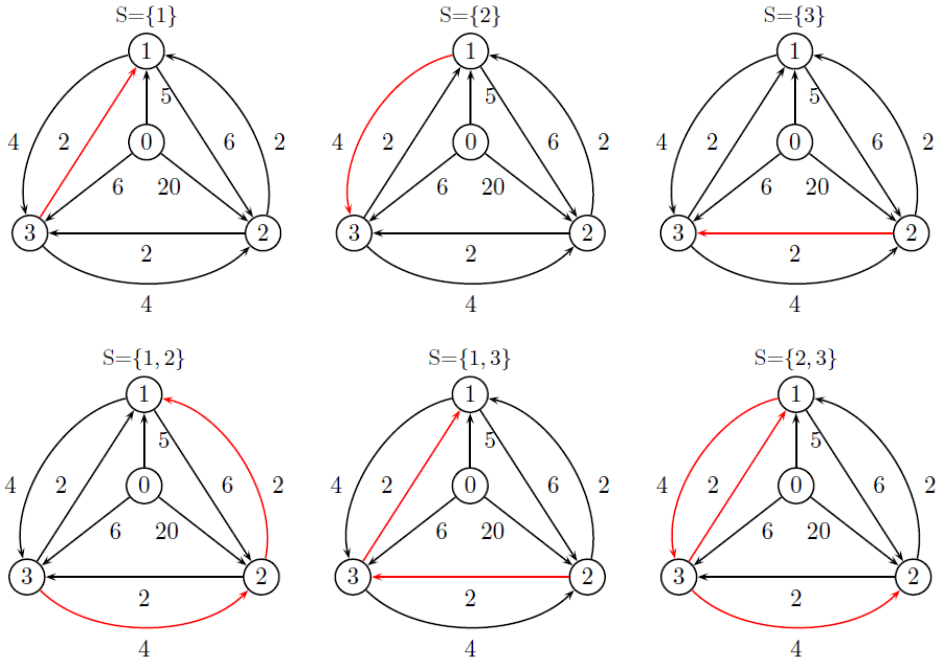
En el juego optimista para arborescencias de mínimo coste los agentes de cada coalición pueden conectarse a la red también a través de agentes pertenecientes a otras coaliciones, ya que se da por hecho que dichos agentes ya están conectados a la red.

Ejemplo 2.5. *A continuación se calcula el juego optimista para el ejemplo 2.2 que venimos viendo a lo largo del capítulo:*



Coalición	Juego optimista
{1}	2
{2}	4
{3}	2
{1,2}	6
{1,3}	4
{2,3}	8
{N}	12

Ahora se muestran todas las coaliciones posibles para el ejemplo 2.2 siguiendo el juego optimista:



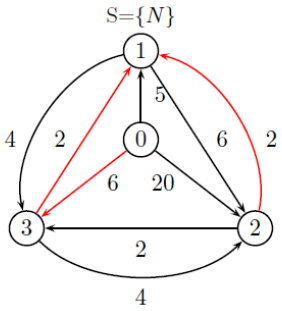


Figura 2.5: Juego optimista para cada una de las coaliciones.

Como sucedía en el caso de los problemas de árboles de mínimo coste, a la hora de calcular el juego optimista, los agentes no han de limitarse a aquellos agentes en su propia coalición o a la fuente para lograr estar conectados. Para unirse a la red será suficiente con que busquen el arco de menor coste, sin importar con quien lo conecte, ya que se asume que el resto de agentes ya están conectados a la fuente.

Forma irreducible de un problema de arborescencias de mínimo coste

En los problemas de arborescencias de mínimo coste también es posible calcular la forma irreducible. Pero al contrario de lo que sucede en los problemas de árboles de mínimo coste, en este tipo de problemas puede haber más de una forma irreducible.

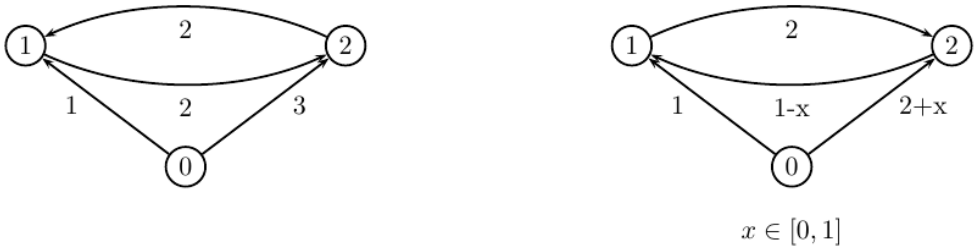


Figura 2.6: Arborescencia con más de una forma irreducible.

Una forma irreducible de un problema de arborescencias se puede definir en función del algoritmo de Edmonds. Dada una matriz de costes C, el algoritmo de búsqueda de una arborescencia minimal finaliza en T etapas. Si $T > 0$, entonces para todo $i \in V$ y $j \in V \backslash \{i\}$, se dice que i y j son $t-hermanos$ si $i, j \in N_k^t$ para algunos $N_k^t \in N^t$, y no existe $t' < t$ tal que $i, j \in N_l^{t'}$ para algún $N_l^{t'} \in N^t$ (Dutta y Mishra 2012).

Definición 2.10. *Para una matriz C con nodos en V, se define la matriz de costes irreducible C^* de la siguiente manera:*
Para cada $i \in V$ y $j \in N \setminus \{i\}$,

$$C_{ij}^* := \sum_{t'=0}^{t-1} \delta_j^{t'},$$

donde i y j son $t-hermanos$.

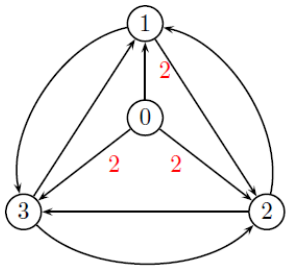
Para poder hallar esta forma irreducible será necesario apoyarse en resultados obtenidos a lo largo de las diferentes etapas del algoritmo de Edmonds. Los pasos a seguir para obtener la forma irreducible en este tipo de problemas son los que siguen:

- Partimos del último grafo obtenido en el algoritmo de Edmonds.
- En dicho grafo a cada arco incidente en un nodo que no sea la fuente, se le asigna el menor coste de entre todos los arcos que incidan en dicho nodo.
- Siguiendo de forma inversa las etapas llevadas a cabo en el algoritmo de Edmonds se suma a cada arco el correspondiente coste que se fue restando etapa a etapa.
- La suma total de esos costes proporciona los nuevos costes para la forma irreducible.

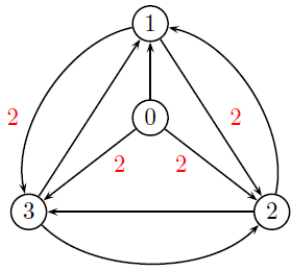
Como el algoritmo de Edmonds ya se ha aplicado para el ejemplo 2.2, nos serviremos de los resultados obtenidos en el ejemplo 2.3 para continuar con el cálculo de la forma irreducible:

Ejemplo 2.6. *Aplicación del algoritmo de Edmonds para el cómputo de la forma irreducible según las etapas del ejemplo 2.3:*

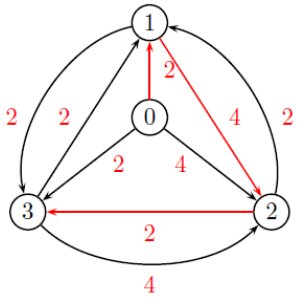
Etapa 1



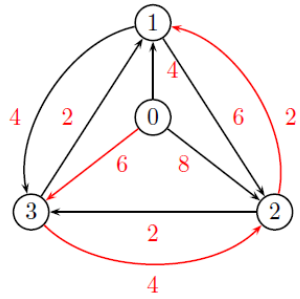
Etapa 2



Etapa 3



Forma irreducible



El resultado es una forma irreducible del problema original de arborescencias de mínimo coste, que mantiene el resultado de la arborescencia de mínimo coste obtenida con el algoritmo de Edmonds previamente y cuyos arcos han visto reducidos sus costes al mínimo posible sin que se vea alterado el problema original. Además, si la matriz de costes es simétrica, la matriz irreducible obtenida con este procedimiento coincide con la matriz irreducible definida en el caso de los problemas de árboles de mínimo coste.

2.4. Reglas de reparto

Una vez obtenida la arborescencia de mínimo coste y, por tanto, el mínimo coste total para conectar a todos los agentes a la fuente, resulta interesante determinar cómo se ha de repartir dicho coste entre

los agentes implicados, para lo que será necesario aplicar alguna regla de reparto. Del mismo modo que previamente ocurría con los problemas de árboles de mínimo coste, nos limitaremos a exponer dos de esas reglas de reparto, la regla de Bird y la regla Folk.

2.4.1. La regla de Bird

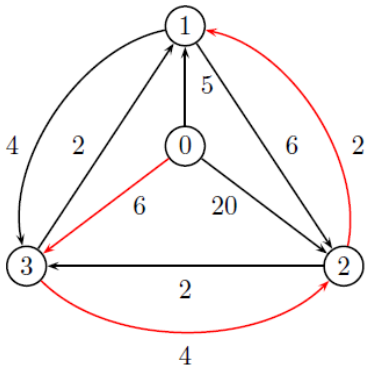
La idea general vuelve a ser la misma que en los problemas de árboles de mínimo coste, es decir, a cada agente se le asigna el coste del arco que lo precede inmediatamente. Así cada agente paga el coste del arco que incide en él de acuerdo a la arborescencia minimal.

Definición 2.11. *Dado un problema de arborescencias de mínimo coste (N_0, C) y una arborescencia de mínimo coste ma asociada a dicho problema, la regla de Bird para la asignación de costes se define como:*

$$B_i(N_0, C, a) = c_{i^0 i},$$

donde i^0 representa el predecesor inmediato de i en la arborescencia a.

Ejemplo 2.7. *Siguiendo la regla de Bird para el reparto de costes, su aplicación al ejemplo 2.2 es el que sigue:*



$$B_1(N_0, C, a) = c_{01} = 2$$
$$B_2(N_0, C, a) = c_{12} = 4$$
$$B_3(N_0, C, a) = c_{23} = 6$$
$$B(N_0, C, a) = (2, 4, 6)$$

En este tipo de problemas también sucede que el reparto según la regla de Bird variará en función de la arborescencia elegida para aquellos casos en que exista más de una arborescencia de mínimo coste para un mismo problema.

2.4.2. La regla Folk

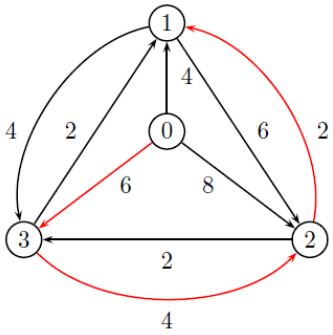
Nuevamente recurriremos a los juegos cooperativos para plantear la regla Folk aplicada a los problemas de arborescencias de mínimo coste. Esta vez será calculada como el valor de Shapley para el juego optimista y también como el juego pesimista asociado a una forma irreducible del problema de arborescencias de mínimo coste. A mayores se expondrá un método diferente para calcular la regla Folk mediante la aplicación de la fórmula obtenida a partir del algoritmo de Edmonds.

Valor de Shapley del juego pesimista de la forma irreducible

Definición 2.12. *La regla Folk se define como el valor de Shapley para el juego pesimista de la forma irreducible:*

$$Folk(N_0, C) = Sh(N, v_{C^*}).$$

Ejemplo 2.8. *Aplicando el juego pesimista visto previamente a la forma irreducible obtenida en el ejemplo 2.6 se obtiene el siguiente resultado:*



Coalición	Juego pesimista
{1}	4
{2}	8
{3}	6
{1,2}	10
{1,3}	8
{2,3}	10
{N}	12

Para este ejemplo también se presentan las diferentes posibles coaliciones siguiendo el juego pesimista:

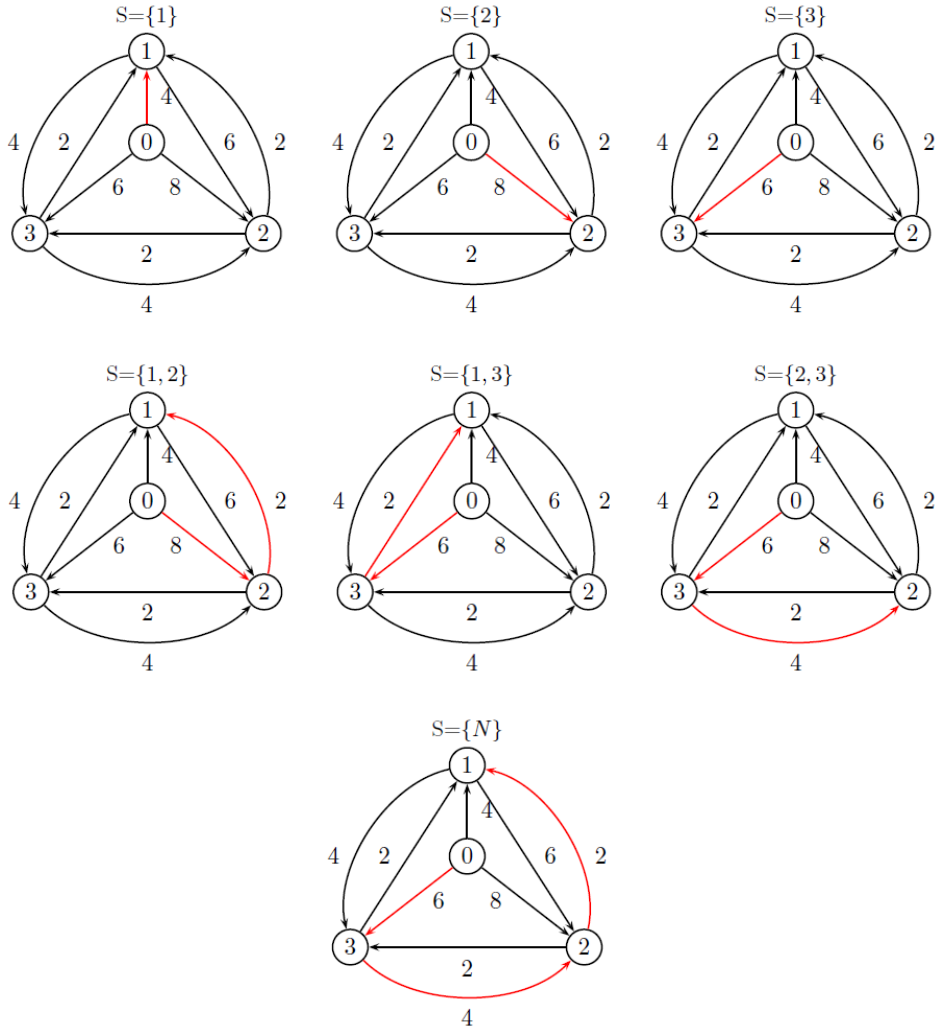
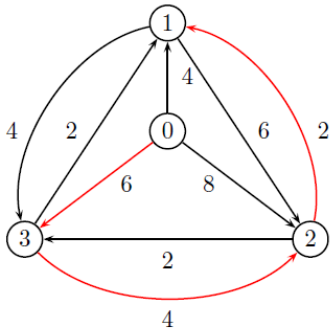


Figura 2.7: Juego pesimista de la forma irreducible para cada una de las coaliciones.

Como acabamos de calcular el juego cooperativo pesimista para la forma irreducible ya podemos continuar con la presentación de la regla Folk mediante el cómputo del valor de Shapley para dicho juego aplicado al ejemplo 2.3:



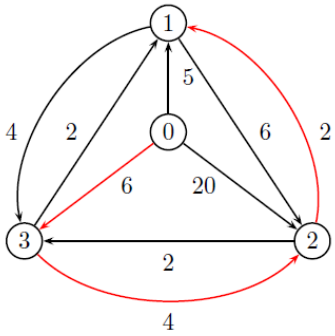
Orden	Jugador 1	Jugador 2	Jugador 3
123	4	6	2
132	4	4	4
213	2	8	2
231	2	8	2
312	2	4	6
321	2	4	6
Total	16	34	22

$$Folk(N_0, C) = \left(\tfrac{8}{2}, \tfrac{17}{2}, \tfrac{11}{2}\right)$$

De manera análoga al procedimiento seguido en los problemas de árboles de mínimo coste se puede comprobar que la regla Folk también coincide con el valor de Shapley del juego optimista para los problemas de arborescencias de mínimo coste.

Valor de Shapley del juego optimista

Ejemplo 2.9. *Al haberse definido previamente el juego optimista para problemas de arborescencias, se procederá directamente a la obtención de la regla Folk mediante el cálculo del valor de Shapley del juego optimista:*



Orden	Jugador 1	Jugador 2	Jugador 3
123	2	4	6
132	2	8	2
213	2	4	6
231	4	4	4
312	2	8	2
321	4	6	2
Total	16	34	22

$$Folk(N_0, C) = \left(\tfrac{8}{2}, \tfrac{17}{2}, \tfrac{11}{2}\right)$$

Al obtener el resultado se comprueba que coincide con el obtenido mediante el cómputo del valor de Shapley para el juego pesimista asociado a la forma irreducible del problema del ejemplo 2.2.

Fórmula obtenida a partir del algoritmo de Edmonds

Otro método posible para el cálculo de la regla Folk es mediante la fórmula obtenida a partir del algoritmo de Edmonds, introducida en Bahel y Trudeau (2017). Este procedimiento está basado en el propio algoritmo de Edmonds utilizado para encontrar arborescencias de mínimo coste y su funcionamiento es el que sigue:

- En cada una de las etapas utilizadas en el algoritmo de Edmonds se tienen en cuenta las cantidades descontadas por los arcos incidentes en los nodos o supernodos.

- Se divide equitativamente cada una de esas cantidades entre los agentes que forman parte del nodo o supernodo al que apunta el arco incidente correspondiente a esa cantidad.

Definición 2.13. *La fórmula obtenida a partir del algoritmo de Edmonds para la consecución de un reparto de los costes para cada agente $i \in N$, viene dada por:*

$$Folk_i(N_0, C) = \sum_{t=0}^{T} \frac{\delta^t_{N^t(i)}}{\mid N^t(i) \mid}.$$

A continuación se muestra de manera práctica, siguiendo con el ejemplo utilizado a lo largo del capítulo, cómo se aplica el algoritmo de Edmonds para obtener un reparto de los costes entre los agentes.

Ejemplo 2.10. *Como el algoritmo de Edmonds ya está aplicado paso a paso en el ejemplo 2.3, a continuación se muestran unicamente los costes de cada etapa necesarios para el cálculo del reparto:*

Cuadro 2.1: La regla Folk mediante la fórmula de Edmonds.

Etapa 1.	Etapa 2.	Etapa 3.
$N^1 = \{\{1\}, \{2\}, \{3\}\}$.	$N^2 = \{\{1\}, \{23\}\}$.	$N^3 = \{\{123\}\}$.
$\delta^1_1 = 2$.	$\delta^2_1 = 0$.	$\delta^3_{123} = 2$.
$\delta^1_2 = 4$.	$\delta^2_{23} = 2$.	
$\delta^1_3 = 2$.		
$Folk(N_0, C) = \left(2 + 0 + \frac{2}{3}, 4 + \frac{2}{2} + \frac{2}{3}, 2 + \frac{2}{2} + \frac{2}{3}\right) = (\frac{8}{3}, \frac{17}{3}, \frac{11}{3})$.		

Para seguir este método es suficiente con repartir los costes de los arcos implicados en cada una de las etapas del algoritmo de Edmonds entre aquellos agentes que se vean beneficiados por la selección de esos arcos. En este caso, por ejemplo, el arco (0,1-2-3) de la etapa 3 es un arco que favorece la conexión de los agentes 1,2 y 3, por este motivo su coste se divide a partes iguales entre todos los agentes ($\frac{2}{3}$).

Capítulo 3

Optimización y cooperación en problemas de árboles de mínimo coste con grupos

A continuación se presenta un tipo de problemas de árboles de mínimo coste en los que los agentes se encuentran agrupados. Estas agrupaciones o coaliciones se deben a diferentes motivos, desde acuerdo o afinidad sobre cierto tema hasta proximidad geográfica, como sucedería con agentes pertenecientes a distintos territorios.

Para la elaboración de este capítulo hemos hecho una revisión de los principales resultados (excluyendo las caracterizaciones de la regla Folk) de los trabajos de Bergantiños y Gómez-Rúa (2010, 2015).

3.1. Introducción

El modelo clásico de problemas de árboles de mínimo coste planteado hasta el momento serviría igualmente para modelizar esta situación que se menciona, pero no estaría contemplando el hecho de que los agentes están agrupados en distintas zonas geográficas. Para tener este hecho en cuenta, Bergantiños y Gómez-Rúa (2010) proponen la partición $P = \{P^1, ..., P^m\}$ del conjunto de agentes N. Para cada $k = 1, ..., m$, donde P^k representa a cada uno de los grupos de agentes. Una ordenación $\pi \in \Pi^N$ con respecto a P será admisible si es una ordenación de N tal que los elementos de las coaliciones aparecen ordenados de forma consecutiva. Π_P^N denota el conjunto de ordenaciones admisibles con respecto a P.

Definición 3.1. *Un problema de árboles de mínimo coste con grupos es una terna (N_0, C, P) donde (N_0, C) es un problema de árboles de mínimo coste, y $P = \{P^1, ..., P^m\}$ es una partición de N.*

En los trabajos de Bergantiños y Gómez-Rúa (2010, 2015) se añade además la condición de que para cada $k = 1, ..., m$,

$$\max_{i,j \in P^k} \{c_{ij}\} \leq \min_{i \in P^k, j \in N_0 \setminus P^k} \{c_{ij}\}.$$

Esta condición hace referencia a que el coste máximo entre grupos sea menor o igual que el mínimo coste entre cada grupo y el resto de grupos. Si bien es cierto que puede resultar interesante tener en cuenta esta condición, no será necesaria para los objetivos de este trabajo, por lo que se prescindirá de la misma.

Podemos adaptar el grafo utilizado en el capítulo 1 para ejemplificar los problemas de árboles de mínimo coste de forma que se puedan diferenciar distintos grupos.

Ejemplo 3.1. *Como estamos trabajando con un ejemplo sencillo, con pocos arcos y nodos para poder operar comodamente de forma manual, en este capítulo se presentarán las particiones posibles para 3 agentes y así veremos qué sucede con los distintos casos con los que se trabajará.*

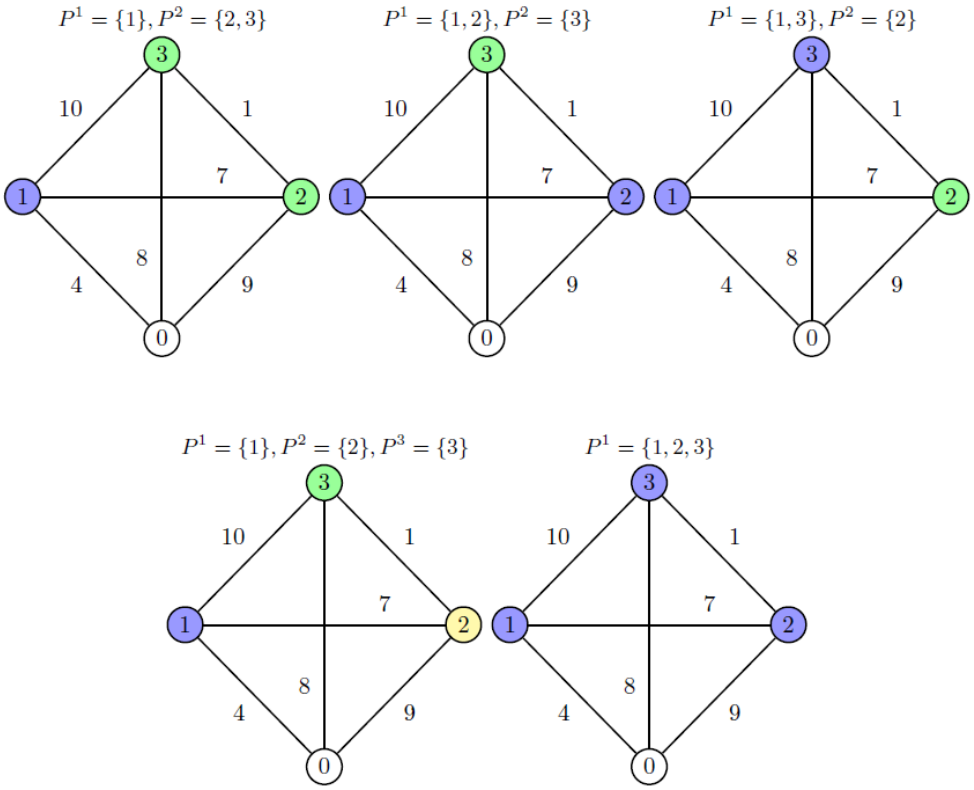


Figura 3.1: Todas las particiones posibles con tres agentes.

La existencia de particiones entre los agentes no altera el resultado a la hora de encontrar un árbol de mínimo coste, por lo que la solución obtenida en el capítulo 1 servirá para continuar con la exposición de los problemas con grupos. Asimismo, los juegos cooperativos y las formas irreducibles también se mantienen, de forma que será suficiente con ver cómo se lleva a cabo el reparto de los costes teniendo en cuenta la nueva estructura grupal.

3.2. La regla Folk con grupos

Para este tipo de problemas se planteará como única regla de reparto la regla Folk. Para calcularla habrá que hacerlo sobre la forma irreducible del problema y también será necesario tener en cuenta la estructura grupal de los agentes.

Comenzamos definiendo una regla de reparto para problemas de árboles de mínimo coste con grupos (Bergantiños y Gómez-Rúa 2010) como una función f tal que $f(N_0, C, P) \in \Re^N$ y $\sum_{i \in N} f_i(N_0, C, P) = m(N_0, C)$ para cada problema de árboles de mínimo coste (N_0, C).

Del mismo modo que fue necesario formular el valor de Shapley al introducir la regla Folk en los problemas de árboles de mínimo coste, para los problemas con grupos resulta imprescindible presentar el valor coalicional o valor de Owen. El valor de Owen (Owen 1977) es una generalización del valor

de Shapley adaptada para repartir los costes de la gran coalición de un juego TU entre los agentes que la forman, cuando existe una estructura coalicional previa, es decir, cuando los jugadores están organizados en grupos por motivos de afinidad, proximidad, etc.

Una permutación $\pi \in \Pi^N$ es admisible con respecto a P si, dados $i, i' \in P^k$ y $j \in N$ tales que $\pi(i) < \pi(j) < \pi(i')$, entonces $j \in P^k$. Denotamos por Π_N^P al conjunto de todas las permutaciones de N admisibles con respecto a P. En este caso, el pago de cada jugador es una media ponderada, sobre las permutaciones admisibles de las contribuciones marginales del jugador a las coaliciones a las que pertenece.

Definición 3.2. *Dados (N, v, P) e $i \in P^k \in P$, el valor de Owen se define como*

$$Ow(N, v, P) = \frac{1}{\mid \Pi_P^N \mid} \sum_{\pi \in \Pi_P^N} m^\pi(N, v).$$

En el trabajo de Bergantiños y Gómez-Rúa (2015) se define la regla Folk para problemas de árboles de mínimo coste con grupos como el valor de Owen aplicado al juego pesimista de la forma irreducible del problema de árboles de mínimo coste respetando los grupos establecidos.

La forma irreducible de un problema de árboles de mínimo coste con grupos se calcula de igual modo que si no existieran los grupos, ya que éstos no se tienen en cuenta a la hora de calcularla.

Definición 3.3. *Dado un problema de árboles de mínimo coste con grupos (N_0, C, P), se define la regla Folk con grupos como*

$$Folk(N_0, C, P) = Ow(N, v_{c^*}, P).$$

Además, se tiene que $Folk(N_0, C, P^N) = Folk(N_0, C, P^n) = Folk(N_0, C)$, donde $P^N = \{N\}$ y $P^n = \{\{i\} : i \in N\}$.

Ejemplo 3.2. *Como la forma irreducible ya está calculada en el ejemplo 1.8 será suficiente con aplicar el valor de Owen al juego pesimista para así obtener la regla Folk:*

Cuadro 3.1: La regla Folk mediante el valor de Owen para el juego pesimista de la forma irreducible.

$P = \{\{1\},\{2,3\}\}$				$P = \{\{1,2\},\{3\}\}$				$P = \{\{1,3\},\{2\}\}$			
Orden	Jugador 1	Jugador 2	Jugador 3	Orden	Jugador 1	Jugador 2	Jugador 3	Orden	Jugador 1	Jugador 2	Jugador 3
123	4	7	1	123	4	7	1	132	4	1	7
132	4	1	7	213	4	7	1	213	4	7	1
231	4	7	1	312	4	1	7	231	4	7	1
321	4	1	7	321	4	1	7	312	4	1	7
Total	16	16	16	Total	16	16	16	Total	16	16	16
$Folk(N_0, C, P) = (4, 4, 4)$				$Folk(N_0, C, P) = (4, 4, 4)$				$Folk(N_0, C, P) = (4, 4, 4)$			

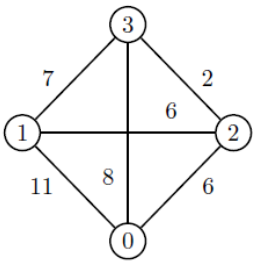
Como se ha explicado previamente, en el cálculo de la regla Folk para problemas de árboles de mínimo coste con grupos, las particiones P^N y P^n siempre generan el mismo resultado que en este mismo tipo de problemas sin grupos. Este es el motivo por el cual se ha prescindido de estas particiones a la hora de calcular la regla Folk en el cuadro 3.1.

De todas formas, después de calcular la regla Folk para las particiones seleccionadas, se comprueba que en este caso el reparto de los costes entre agentes es siempre el mismo, independientemente de

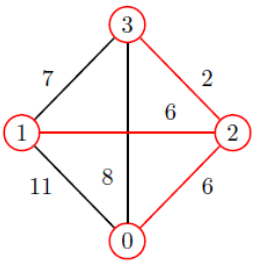
los grupos que se hayan formado. Incluso llega a coincidir con el reparto de los costes del capítulo 1 en el que no había grupos. Para este ejemplo concreto no supone diferencia alguna el hecho de que existan grupos entre los agentes a la hora de llevar a cabo el reparto de los costes. Esto no siempre tiene porque ser así y dependerá de los arcos implicados y de los costes asociados a los mismos.

Ejemplo 3.3. *Se introduce un nuevo problema de árboles de mínimo coste con el fin de aplicar la regla Folk a un caso práctico en el que se obtengan repartos distintos en función de las particiones. Se continuará calculando la regla Folk únicamente para las tres particiones tenidas en cuenta en el ejemplo previo.*

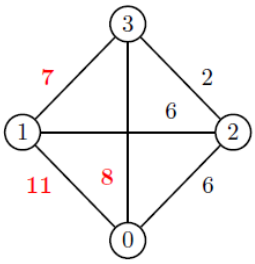
Grafo original



Árbol de mínimo coste



Forma irreducible



Coalición	Juego pesimista
{1}	6
{2}	6
{3}	6
{1,2}	12
{1,3}	12
{2,3}	8
{N}	14

Figura 3.2: Grafo original, árbol de mínimo coste, su forma irreducible y juego pesimista.

Al haberse presentado ya el juego pesimista para la forma irreducible del problema, nos encontramos en disposición de poder calcular los repartos de costes que se obtienen mediante la regla Folk con las distintas particiones.

Cuadro 3.2: La regla Folk mediante el valor de Owen para el juego pesimista de la forma irreducible.

$P = \{\{1\},\{2,3\}\}$				$P = \{\{1,2\},\{3\}\}$				$P = \{\{1,3\},\{2\}\}$			
Orden	Jugador 1	Jugador 2	Jugador 3	Orden	Jugador 1	Jugador 2	Jugador 3	Orden	Jugador 1	Jugador 2	Jugador 3
123	6	6	2	123	6	6	2	132	6	2	6
132	6	2	6	213	6	6	2	213	6	6	2
231	2	6	6	312	6	2	6	231	2	6	6
321	2	6	6	321	2	6	6	312	6	2	6
Total	16	20	20	Total	20	20	16	Total	20	16	20
$Folk(N_0,C,P) = (4,5,5)$				$Folk(N_0,C,P) = (5,5,4)$				$Folk(N_0,C,P) = (5,4,5)$			

Se comprueba que con este nuevo problema de árboles de mínimo coste con grupos se obtienen diferentes repartos en función de las particiones que formen los agentes. De este modo se han visto dos ejemplos con diferentes resultados; el primero de ellos manteniendo el reparto de los costes independientemente de las particiones e incluso de la existencia o no de grupos. El segundo de ellos, con repartos diferenciados para cada partición.

Capítulo 4

Optimización y cooperación en problemas de arborescencias de mínimo coste con grupos

De manera análoga a lo que sucede en los problemas de árboles de mínimo coste con grupos, en los problemas de arborescencias también puede resultar útil tener en cuenta la predisposición de los agentes a formar grupos o coaliciones, en función de sus afinidades, proximidad geográfica, etc. La existencia de estas particiones atañe principalmente al reparto de los costes entre los agentes, dado que para la resolución del problema y, por tanto, la obtención de la arborescencia minimal no se tiene en cuenta la existencia de grupos.

Aunque los problemas de arborescencias de mínimo coste con grupos se introducen por primera vez en este trabajo, hemos tratado de respetar la notación utilizada en los problemas de arborescencias de mínimo coste y en los problemas de árboles de mínimo coste con grupos. Nótese que, además, los problemas de arborescencias de mínimo coste con grupos englobarán a la clase de problemas de árboles de mínimo coste con grupos por lo que trataremos, en la medida de lo posible, de extender las definiciones y resultados de los problemas de árboles de mínimo coste con grupos.

4.1. Introducción

A la hora de trabajar con problemas de arborescencias con grupos será necesario mantener ciertas premisas planteadas en los problemas de árboles con grupos. Para definir estos problemas se conserva el concepto de partición $P = \{P^1, ..., P^m\}$ del conjunto de agentes N, con $k = 1, ..., m$, donde P^k representa a cada uno de los grupos de agentes. También se conserva el concepto de ordenación $\pi \in \Pi^N$ con respecto a P, que será admisible si es una ordenación de N tal que los elementos de las coaliciones aparecen ordenados de forma consecutiva. Π_P^N seguirá denotando el conjunto de ordenaciones admisibles con respecto a P.

Definición 4.1. *Un problema de arborescencias de mínimo coste con grupos es una terna (N_0, C, P) donde (N_0, C) es un problema de arborescencias de mínimo coste y $P = \{P^1, ..., P^m\}$ es una partición de N.*

Ejemplo 4.1. *Teniendo esto en cuenta y siguiendo con el patrón llevado a cabo en el capítulo 3, continuaremos con el problema de la arborescencia de mínimo coste del ejemplo 2.2 y se le aplicarán las posibles agrupaciones teniendo en cuenta que existen tres agentes.*

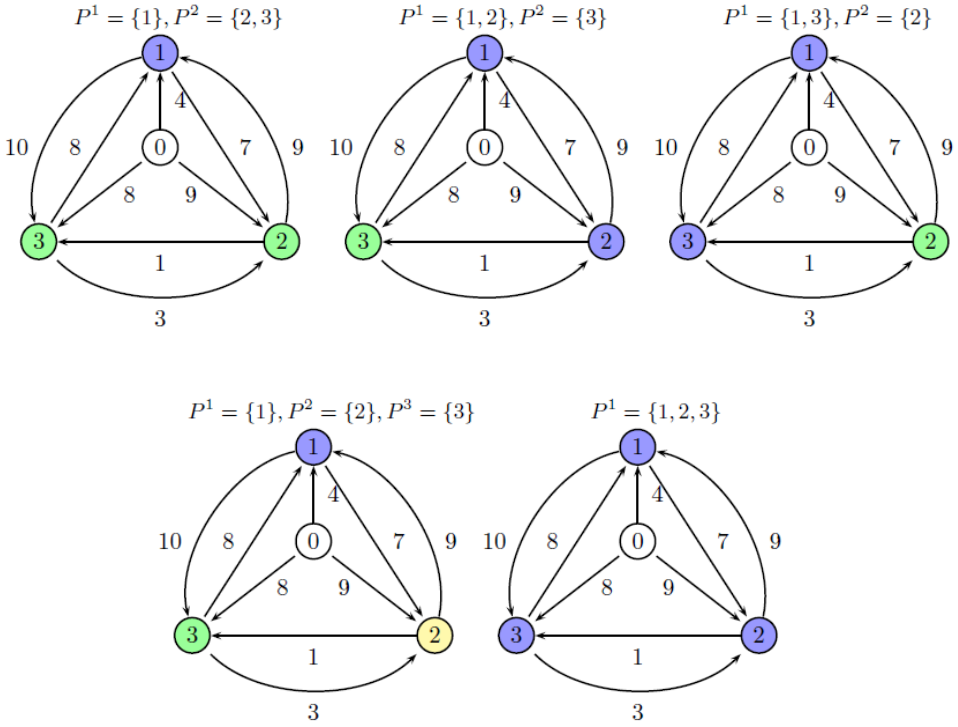


Figura 4.1: Todas las particiones posibles con tres agentes.

Al trabajar con problemas de arborescencias con grupos, también sucede que tanto las arborescencias de mínimo coste como las formas irreducibles se mantienen con respecto a los problemas sin grupos. Por tanto, seguiremos trabajando con el ejemplo 2.2 para así echar mano de algunos de los resultados obtenidos a lo largo del capítulo 2. Al igual que sucedía en el capítulo de problemas de árboles de mínimo coste con grupos, simplemente será necesario exponer una nueva regla de reparto que tenga en cuenta la existencia de las particiones.

4.2. La regla Folk con grupos

A la hora de buscar un reparto de los costes entre los agentes cuando nos encontramos con problemas de arborescencias con grupos, la regla Folk vuelve a surgir como una posible solución.

Así, una regla de reparto para problemas de arborescencias de mínimo coste con grupos es una función f tal que $f(N_0, C, P) \in \Re^N$ y $\sum_{i \in N} f_i(N_0, C, P) = m(N_0, C)$ para cada problema de arborescencias de mínimo coste (N_0, C).

Al estar tratando de nuevo con problemas con grupos tendremos que volver a echar mano del valor de Owen, definido en el capítulo anterior, como método para adaptar la regla Folk. Del mismo modo que sucedía en los problemas de árboles de mínimo coste con grupos, la regla Folk se calcula para problemas de arborescencias de mínimo coste con grupos como el valor de Owen de la forma irreducible obtenida a partir de las etapas del algoritmo de Edmonds. En los problemas de arborescencias de

mínimo coste con grupos también sucede que la forma irreducible se calcula de forma idéntica a si no existieran grupos. Esto es así porque las particiones no intervienen en ningún momento a la hora de obtener la forma irreducible de un problema.

Definición 4.2. *Dado un problema de arborescencias de mínimo coste con grupos (N_0, C, P), se define la regla Folk con grupos como*

$$Folk(N_0, C, P) = Ow(N, v_{C^*}, P).$$

Además, se tiene que $Folk(N_0, C, P^N) = Folk(N_0, C, P^n) = Folk(N_0, C)$, donde $P^N = \{N\}$ y $P^n = \{\{i\} : i \in N\}$.

Como acabamos de ver que el reparto de los costes para P^N y P^n coinciden con los mostrados en el capítulo sin grupos, también se prescindirá en los problemas de arborescencias de mínimo costes con grupos de calcular la regla Folk para las mencionadas particiones.

Nótese que, además, dado que en problemas con grafos no dirigidos la matriz irreducible en los problemas de arborescencias coincide con la matriz irreducible definida para los problemas de árboles de mínimo coste, la definición de la regla Folk con grupos también coincidirá con la definición para problemas de árboles de mínimo coste con grupos si los grafos son no dirigidos.

Ejemplo 4.2. *La regla Folk calculada mediante el valor de Owen para el juego pesimista de la forma irreducible del ejemplo 2.6 se aplica al problema de arborescencias para los diferentes grupos seleccionados de la siguiente manera:*

Cuadro 4.1: La regla Folk mediante el valor de Owen para el juego pesimista de la forma irreducible.

$P = \{\{1\},\{2,3\}\}$				$P = \{\{1,2\},\{3\}\}$				$P = \{\{1,3\},\{2\}\}$			
Orden	Jugador 1	Jugador 2	Jugador 3	Orden	Jugador 1	Jugador 2	Jugador 3	Orden	Jugador 1	Jugador 2	Jugador 3
123	2	8	2	123	2	4	6	132	2	4	6
132	2	4	6	213	2	8	2	213	2	4	6
231	4	4	4	312	4	4	4	231	2	8	2
321	2	8	2	321	4	6	8	312	4	6	8
Total	12	22	14	Total	10	22	16	Total	10	24	14
$Folk(N_0,C,P) = \left(3, \frac{11}{2}, \frac{7}{2}\right).$				$Folk(N_0,C,P) = \left(\frac{5}{2}, \frac{11}{2}, 4\right).$				$Folk(N_0,C,P) = \left(\frac{5}{2}, 6, \frac{7}{2}\right).$			

La regla Folk para problemas de arborescencias con grupos también se puede obtener de forma directa mediante el uso de los costes obtenidos en cada una de las etapas del algoritmo de Edmonds. De esta forma, es posible obtener el reparto de costes de la regla Folk a partir de un algoritmo polinomial.

Teorema 1. *Dado un problema de arborescencias con grupos (N_0, C, P) y dado $i \in P_k \in P$, tenemos que*

$$Folk_i(N_0, C, P) = \sum_{t=0}^{T} \frac{\delta_{N^t(i)}}{|j \in \{1, ..., m\} : P_j \cap N^t(i) \neq \emptyset| \cdot |P_k \cap N^t(i)|}.$$

Demostración:
Tenemos que

$$\begin{aligned} Folk_i(N_0, C, P) &= Ow_i(N, v_{C^*}, P) \\ &= \frac{1}{\mid \Pi_P^N \mid} \sum_{\pi \in \Pi_P^N} m_i^{\pi}(N, v_{C^*}) \\ &= \frac{1}{\mid \Pi_P^N \mid} \sum_{\pi \in \Pi_P^N} v_{C^*}(Pre(i,\pi) \cup \{i\}) - v_{C^*}(Pre(i,\pi)). \end{aligned}$$

En el trabajo de Dutta y Kar (2004), se demuestra que

$$v_{C^*}(S \cup \{i\}) - v_{C^*}(S) = min_{k \in S \cup \{0\}} c^*_{ki}.$$

De esta forma, se tiene que

$$Folk_i(N_0, C, P) = \frac{1}{\mid \Pi_P^N \mid} \sum_{\pi \in \Pi_P^N} min_{l \in Pre(i,\pi) \cup \{0\}} c^*_{li}.$$

Por definición, sabemos que la matriz irreducible se obtiene a partir del algoritmo de Edmonds como $c^*_{ji} := \sum_{t'=0}^{t-1} \delta_i^{t'}$ para todo $i \in N$ y $j \in N \cup \{0\}$ tales que i y j son t-hermanos, con $t \in \{0, \ldots, T\}$.
Esto significa que si para cada $t \in \{0, \ldots, T\}$ definimos las matrices $\bar{C}^t$ como $\bar{c}^t_{li} = 0$ si $l \in N^t(i)$ y $\bar{c}^t_{li} = \delta_{N^t(i)}$ en otro caso (para todo $l \in N \cup \{0\}$ y todo $i \in N$), entonces $C^* = \sum_{t=0}^{T} \bar{C}^t$.

De esta forma,

$$Folk_i(N_0, C, P) = \frac{1}{\mid \Pi_P^N \mid} \sum_{\pi \in \Pi_P^N} min_{l \in Pre(i,\pi) \cup \{0\}} \sum_{t=0}^{T} \bar{c}^t_{li}.$$

Por definición, sabemos que si $c^*_{li} \leq c^*_{ji}$ para $l, i \in N \cup \{0\}$ entonces $\bar{c}^t_{li} \leq \bar{c}^t_{ji}$ para todo $t \in \{0, \ldots, T\}$. Por tanto

$$\begin{aligned} Folk_i(N_0, C, P) &= \frac{1}{\mid \Pi_P^N \mid} \sum_{\pi \in \Pi_P^N} min_{l \in Pre(i,\pi) \cup \{0\}} \sum_{t=0}^{T} \bar{c}^t_{li} \\ &= \sum_{t=0}^{T} \frac{1}{\mid \Pi_P^N \mid} \sum_{\pi \in \Pi_P^N} min_{l \in Pre(i,\pi) \cup \{0\}} \bar{c}^t_{li}. \end{aligned}$$

Además, para todo $t \in \{0, \ldots, T\}$ tenemos que dada una permutación $\pi \in \Pi_P^N$, si i es el primer agente de $N^t(i)$ en ser elegido entonces $min_{l \in Pre(i,\pi) \cup \{0\}} \bar{c}^t_{li} = \delta_{N^t(i)}$ y si esto no ocurre entonces $min_{l \in Pre(i,\pi) \cup \{0\}} \bar{c}^t_{li} = 0$.
Dado el conjunto de posibles órdenes Π^P, sabemos que la probabilidad de que i sea el primer agente de $N^t(i)$ en ser elegido es $\frac{1}{|j \in \{1,\ldots,m\} : P_j \cap N^t(i) \neq \emptyset| \cdot |P_k \cap N^t(i)|}$.
Esto significa que

$$Folk_i(N_0, C, P) = \sum_{t=0}^{T} \frac{\delta_{N^t(i)}}{|j \in M : P_j \cap N^t(i) \neq \emptyset| \cdot |P_k \cap N^t(i)|}.$$

Como en el ejemplo 2.3 ya se puede encontrar el algoritmo de Edmonds etapa por etapa aplicado al ejemplo 2.2, en este capítulo nos serviremos de esos resultados para calcular el reparto de los costes con grupos.

Ejemplo 4.3. *A continuación se muestran los diferentes resultados en función de cómo estén formadas las particiones.*

Cuadro 4.2: La regla Folk mediante las etapas del algoritmo de Edmonds.

Etapa 1	Etapa 2	Etapa 3
$N^1 = \{\{1\},\{2\},\{3\}\}.$	$N^2 = \{\{1\},\{23\}\}.$	$N^3 = \{\{123\}\}.$
$\delta_1^1 = 2.$	$\delta_1^2 = 0.$	$\delta_{123}^3 = 2.$
$\delta_2^1 = 4.$	$\delta_{23}^2 = 2.$	
$\delta_3^1 = 2.$		
$P = \{\{1\},\{2,3\}\}, Folk(N_0, C, P) = \left(2+0+1, 4+1+\frac{1}{2}, 2+1+\frac{1}{2}\right) = (3, \frac{11}{2}, \frac{7}{2}).$		
$P = \{\{1,2\},\{3\}\}, Folk(N_0, C, P) = \left(2+0+\frac{1}{2}, 4+1+\frac{1}{2}, 2+1+1\right) = (\frac{5}{2}, \frac{11}{2}, 4).$		
$P = \{\{1,3\},\{2\}\}, Folk(N_0, C, P) = \left(2+0+\frac{1}{2}, 4+1+1, 2+1+\frac{1}{2}\right) = (\frac{5}{2}, 6, \frac{7}{2}).$		

Para alcanzar los distintos repartos obtenidos en la tabla anterior, es necesario únicamente tener en cuenta los costes de los arcos seleccionados en las diferentes etapas del algoritmo de Edmonds y repartirlos entre aquellos agentes que se vean beneficiados por dichos arcos. Para lograr el reparto final solo hay que sumar los costes en cada etapa para cada uno de los agentes.

Después de haber calculado la regla Folk para un problema de arborescencias con grupos aplicando dos métodos diferentes, se comprueba que el resultado es siempre el mismo para cada una de las posibles particiones. Lo que se desprende de los repartos obtenidos es que los tres jugadores obtienen el reparto más favorable para ellos mismos en aquella particiones en las que se asocian con otro jugador y no están solos. También cabe destacar que los repartos que obtienen los agentes cuando están agrupados son mejores que aquellos logrados sin particiones de ningún tipo. De igual modo, los repartos empeoran con respecto al problema sin particiones cuando en el problema con grupos están aislados. A diferencia de lo que sucedió con el primer ejemplo utilizado en los problemas de árboles de mínimo coste con grupos, aquí sí se alcanzan diferentes repartos según haya grupos o no y en función de cuales sean esos grupos. Los resultados aquí comentados son para un ejemplo específico y no tiene por qué ser así en todos los casos.

Capítulo 5

Programación en R.

Actualmente, al trabajar con problemas reales, el tamaño de los mismos hace poco viable su resolución manualmente, por lo que el uso de programas informáticos que los resuelvan en un mínimo tiempo y de manera fiable resulta de gran utilidad. Por este motivo nos hemos propuesto aportar una nueva manera de obtener en lenguaje R la regla Folk de reparto de costes para problemas de arborescencias sin grupos. Con el propósito de lograr dicha regla de reparto se seguirán las etapas llevadas a cabo durante la aplicación del algoritmo de Edmonds, ya que éste resulta más eficaz computacionalmente que aquellos métodos en que el reparto se obtiene mediante el cálculo del valor de Shapley. Además, dicha función también contemplará la opción de resolver problemas de arborescencias con grupos. Para llevar a cabo la mencionada programación echaremos mano de los paquetes de R optrees (v1.0; Fontenla 2014a) y cooptrees (v1.0; Fontenla 2014b), en los que ya se encuentran algunos resultados útiles para la obtención de los repartos de los costes entre los agentes.

5.1. Estado del arte.

Como ya se ha adelantado previamente, a la hora de programar la regla Folk no será necesario partir de cero, dado que ya hay creados dos paquetes de R que engloban ciertos resultados en los que basarse para llevar a cabo la implementación de dicha regla mediante las etapas del algoritmo de Edmonds. Los paquetes de los que partiremos son optrees y cooptrees desarollados como parte del TFM de Fontenla (2014c). El paquete optrees se centra principalmente en la búsqueda de árboles óptimos y abarca los principales algoritmos utilizados para la resolución de los diferentes tipos de problemas relativos a los árboles óptimos. Por su parte, el paquete cooptrees está enfocado en la cooperación tanto en los problemas de árboles de coste mínimo como en los de arborescencias de coste mínimo.

De este modo, la gran mayoría de resultados obtenidos de forma manual a lo largo del presente trabajo se pueden conseguir de manera inmediata con los paquetes de R optrees y cooptrees. Por ello, se aplicarán a continuación los mencionados paquetes al problema de arborescencias contemplado a lo largo del trabajo para obtener computacionalmente algunos resultados que pueden resultar útiles o estar relacionados con la regla Folk y así revisar las bases en las que nos hemos apoyado para desempeñar nuestro trabajo.

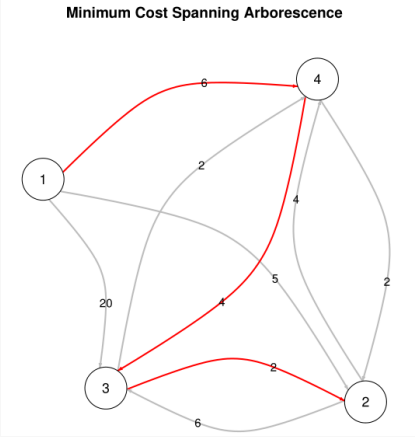
Para comenzar será necesario introducir los datos que definen el problema, indicando los nodos y los arcos que lo conforman. Los nodos se incluyen mediante un vector que contenga el número total de agentes del problema incluida la fuente, desde 1 hasta n. Será necesario tener en cuenta que el nodo fuente generalmente representado por el 0, aquí será siempre el nodo 1. En cuanto a los arcos del problema, se han de introducir sucesivamente en una matriz en la que la primera columna contiene los nodos de salida, la segunda los nodos de llegada y la tercera el coste asignado al arco.

```
## Nodos y arcos
nodes<-1:4
arcs<-matrix(c(1,2,5, 1,3,20, 1,4,6, 2,3,6, 2,4,4, 3,2,2,
3,4,2, 4,2,2, 4,3,4), byrow=T, ncol = 3)
```

Una vez introducidos los datos del problema con el que se quiere trabajar, se puede proceder a resolver distintas cuestiones. En este caso parece natural comenzar por la obtención de la arborescencia de mínimo coste empleando el algoritmo de Edmonds, con este fin se puede utilizar la función getMinimumArborescence. Comprobamos que la salida nos indica los arcos que componen la arborescencia minimal junto con el coste de los mismos, además del coste total de dicha arborescencia. Como se ha indicado expresamente que muestre el grafo, la función nos devuelve también la figura que representa el problema junto con el resultado resaltado en rojo. Hay que tener en cuenta que en el caso de existir más de una arborescencia de mínimo coste, solamente se aportará como solución una de ellas.

```
# Arborescencia de mínimo coste
getMinimumArborescence(nodes, arcs, stages.data = TRUE,
show.data = TRUE, show.graph = TRUE)

##
##  Minimum cost spanning arborescence
##  Algorithm: Edmonds
##  Stages:  3 | Time:  0
##  ------------------------------
##       head     tail    weight
##          1        4         6
##          3        2         2
##          4        3         4
##  ------------------------------
##                    Total = 12
```



En lo concerniente a los juegos cooperativos asociados a los problemas de arborescencias, en el paquete cooptrees únicamente se contempla la obtención del juego pesimista, pudiendo obtenerse tanto para el problema original como para su forma irreducible. La función maCooperative nos dará esta información, además de otros resultados no pertinentes en este caso y que por lo tanto se obviarán.

```
# Juegos cooperativos
maCooperative(nodes,arcs)

##
##  Cooperative games:
##  --------------------------
##      1  2 3 1,2 1,3 2,3 1,2,3
## Pes. 5 20 6  11   8  10    12
## Irr. 4  8 6  10   8  10    12
##  --------------------------
```

Otro de los resultados en relación a nuestro trabajo y que es posible calcular mediante el paquete cooptrees es la forma irreducible de un problema de arborescencias. Del mismo modo que sucedía con getMinimumArborescence, con la función maIrreducible se obtendrá una única forma irreducible para el caso en que exista más de una. El resultado se muestra a través una tabla que contiene todos los arcos indicados mediante su nodo inicial y nodo final, junto con los nuevos pesos adjudicados.

```
# Forma irreducible
maIrreducible(nodes, arcs)

##       head tail weight
##  [1,]    1    2      4
##  [2,]    1    3      8
##  [3,]    1    4      6
##  [4,]    2    3      6
##  [5,]    2    4      4
##  [6,]    3    2      2
##  [7,]    3    4      2
##  [8,]    4    2      2
##  [9,]    4    3      4
```

Para finalizar, con el paquete cooptrees existe la opción de obtener dos reglas de reparto de los costes, la regla Bird y la ERO. Para el caso que nos ocupa nos limitaremos a computar la regla ERO, que como hemos visto anteriormente coincide con la regla Folk, haciendo uso de la función maERO. La salida de la función recoge en una tabla el coste asignado a cada uno de los agentes.

```
# Regla ERO
maERO(nodes, arcs)

##      agent     cost
## [1,]     2 2.666667
## [2,]     3 5.666667
## [3,]     4 3.666667
```

Resulta pertinente para el fin que nos ocupa, tener en cuenta que el cálculo de esta regla de reparto de costes está basada en el valor de Shapley. Este hecho hace que el tiempo del cómputo de la regla tenga un crecimiento exponencial y deje de resultar eficiente a medida que el tamaño del problema y su complejidad incrementen. Este es el motivo principal que nos lleva a programar dicha regla siguiendo las etapas del algoritmo de Edmonds, ya que parece razonable que la implementación de un algoritmo polinomial resulte más efectivo computacionalmente y optimice los tiempos para generar los resultados.

5.2. La regla Folk mediante el algoritmo de Edmonds.

La función creada para calcular la regla Folk a través de los costes obtenidos a los largo de las etapas del algoritmo de Edmonds lleva por nombre maFolk y devolverá como resultado el reparto de los costes para cada uno de los agentes en un problema de arborescencias de mínimo coste tanto para aquellos problemas sin grupos como para los que sí contemplan esta opción. Para ello partiremos de las etapas del algoritmo de Edmonds obtenidas mediante la función getMinimumArborescence del paquete optrees. Tan pronto se tienen todas las etapas seguidas para obtener la arborescencia minimal será necesario tener en cuenta los costes de los arcos seleccionados en cada una de esas etapas además de los agentes implicados en las mismas, para proceder el reparto de dichos costes etapa por etapa.

Concretamente, nos basaremos en tres resultados de la función getMinimumArborescence. En primer lugar, nos serviremos de los arcos de menor coste incidentes en cada uno de los nodos en cada etapa y de sus costes, también aprovecharemos la detección de supernodos y, finalmente, usaremos las equivalencias entre los nodos renombrados después de ser detectado un supernodo y los nombres de esos nodos en la etapa precedente. Habrá que comenzar seleccionando únicamente un arco de menor coste incidente en cada nodo en caso de existir empate en los pesos. Después, será necesario trabajar en ese renombramiento de nodos constante en cada etapa, con el fin de poder relacionar cada uno de los nodos renombrados de cada etapa con los nodo originales y no con los nodos de la etapa anterior. Una vez solventada esta problemática, se tendrán que repartir los costes seleccionados en cada una de las etapas entre los agentes originales implicados en la construcción de esos arcos teniendo en consideración la existencia o ausencia de grupos en el problema a resolver.

Comprobamos el correcto funcionamiento del código aplicándolo al problema base de arborescencias del trabajo y corroboramos que la nueva función maFolk devuelve los resultados esperados para la regla Folk, que han de coincidir necesariamente con los obtenidos para la regla ERO a través de maERO.

```
## Nodos y arcos
nodes<-1:4
arcs<-matrix(c(1,2,5, 1,3,20, 1,4,6, 2,3,6, 2,4,4, 3,2,2,
3,4,2, 4,2,2, 4,3,4), byrow=T, ncol = 3)

# Regla Folk
maFolk(nodes, arcs)

## Cost allocation rule: Folk
##      Agent Cost
## [1,]     2 2.67
## [2,]     3 5.67
## [3,]     4 3.67
```

Además, aplicaremos la función maFolk a un problema de mayor envergadura, para corroborar que sigue respondiendo y ver si se acepta nuestra hipótesis de que su tiempo de ejecución se reduce con respecto al de maERO. Para ello introduciremos un problema de arborescencias con diez nodos, nueve agentes además de la fuente, y un total de ochenta y un arcos con sus respectivos pesos.

```
nodes<-1:10
arcs<-matrix(c(1,2,4, 1,3,6, 1,4,5, 1,5,4, 1,6,8, 1,7,5, 1,8,7, 1,9,4, 1,10,9,
2,3,4, 2,4,7, 2,5,5, 2,6,9, 2,7,6, 2,8,5, 2,9,6, 2,10,6,
3,2,5, 3,4,8, 3,5,1, 3,6,7, 3,7,11, 3,8,1, 3,9,7, 3,10,6,
4,2,6, 4,3,4, 4,5,8, 4,6,5, 4,7,4, 4,8,9, 4,9,5, 4,10,4,
5,2,9, 5,3,2, 5,4,6, 5,6,6, 5,7,6, 5,8,2, 5,9,9, 5,10,7,
6,2,2, 6,3,5, 6,4,6, 6,5,4, 6,7,10, 6,8,6, 6,9,5, 6,10,7,
```

```
7,2,10, 7,3,9, 7,4,5, 7,5,9, 7,6,7, 7,8,5, 7,9,8, 7,10,7,
8,2,7, 8,3,1, 8,4,9, 8,5,2, 8,6,6, 8,7,6, 8,9,5, 8,10,6,
9,2,9, 9,3,4, 9,4,6, 9,5,6, 9,6,7, 9,7,6, 9,8,9, 9,10,4,
10,2,8, 10,3,9, 10,4,5, 10,5,4, 10,6,4, 10,7,11, 10,8,5, 10,9,5),
byrow=T, ncol = 3)

maFolk(nodes,arcs)

## Cost allocation rule: Folk
##       Agent Cost
##  [1,]     2 2.00
##  [2,]     3 2.17
##  [3,]     4 5.00
##  [4,]     5 1.67
##  [5,]     6 4.00
##  [6,]     7 4.00
##  [7,]     8 2.17
##  [8,]     9 4.00
##  [9,]    10 4.00
```

Se comprueba que la función obtiene el resultado para problemas de mayor complejidad sin ningún tipo de problema. Una vez cerciorada la efectividad de la función para lograr el resultado deseado, se procede a comprobar la utilidad de la misma a la hora de acortar los tiempos de ejecución. Consultamos los tiempos de ejecución de ambas funciones para el problema anterior, de tamaño considerable y cuya resolución manual resulta engorrosa y poco viable.

```
# Tiempo de ejecución
system.time(maFolk(nodes,arcs))

##    user  system elapsed
##    0.02    0.00    0.02

system.time(maERO(nodes,arcs))

##    user  system elapsed
##  109.82    0.12  110.77
```

Los resultados obtenidos parecen ratificar la idea de que el uso de un algoritmo polinomial frente al cálculo del valor de Shapley resulta en tiempos de ejecución menores. Mientras que maFolk ha obtenido el resultado en 0.02 segundos, la función maERO ha necesitado de alrededor de 110 segundos para llegar a la misma conclusión. Se trata de una diferencia notable, más aún si se tiene en cuenta que se trata de un problema que contempla un número manejable de nodos. Estos tiempos obtenidos hacen que nuestra propuesta de programar la regla Folk mediante las etapas del algoritmo de Edmonds resulte una mejora con respecto a la fórmula existente hasta el momento, traducida en procesos más ágiles y menor tiempo de ejecución. Esta mayor funcionalidad puede resultar muy ventajosa en problemas que involucren una cantidad mucho mayor de nodos y conexiones.

Por otra parte, si se quiere obtener el reparto de costes para un problema de arborescencias con grupos bastará con añadir la estructura coalicional de los agentes mediante una matriz de dos columnas, en la que la primer columna contenga a los agentes y la segunda el número del grupo al que pertenecen. Conviene recordar que el nodo fuente siempre estará asociado al nodo 1, y que por tanto, será necesario indicar que dicho nodo pertenece a la partición 1, y a partir de ahí continuar enumerando particiones. Después será suficiente con añadir a la función maFolk el argumento unions, que haga

referencia a dicha matriz. Para ver cómo habría que operar en estos casos, aplicaremos la función al problema de arborescencias resuelto en el capítulo anterior y contenido en el cuadro 4.3, con las tres posibles combinaciones de agentes.

Introducimos la matriz de las uniones para cada una de las particiones contempladas en el capítulo 4. Comenzaremos con la partición $P = \{\{2\}, \{3, 4\}\}$, indicando que el agente 2 pertenece a la partición 2 y los agentes 3 y 4 a la partición 3.

```
# Uniones
unions<-matrix(c(1,1, 2,2, 3,3, 4,3),byrow=T, ncol = 2)
maFolk(nodes,arcs,unions)

## Cost allocation rule: Folk
##      Agent Cost
## [1,]     2  3.0
## [2,]     3  5.5
## [3,]     4  3.5
```

Ahora se introducen los datos en la matriz para la partición $P = \{\{2, 3\}, \{4\}\}$.

```
# Uniones
unions<-matrix(c(1,1, 2,2, 3,2, 4,3),byrow=T, ncol = 2)
maFolk(nodes,arcs,unions)

## Cost allocation rule: Folk
##      Agent Cost
## [1,]     2  2.5
## [2,]     3  5.5
## [3,]     4  4.0
```

Por último se modifica la matriz para que los grupos se correspondan con la partición $P = \{\{2, 4\}, \{3\}\}$.

```
# Uniones
unions<-matrix(c(1,1, 2,2, 3,3, 4,2),byrow=T, ncol = 2)
maFolk(nodes,arcs,unions)

## Cost allocation rule: Folk
##      Agent Cost
## [1,]     2  2.5
## [2,]     3  6.0
## [3,]     4  3.5
```

Se comprueba que los resultados obtenidos a través de la función maFolk coinciden con los expuestos en el cuadro 4.3, lo cual indica que la función propuesta funciona también en el caso de los problemas de arborescencias de mínimo coste con grupos.

Con el fin de mostrar más aplicaciones de la función maFolk, acudiremos al ejemplo compuesto por diez nodos utilizado previamente y añadiremos una matriz que recoja una de las posibles agrupaciones entre agentes. La partición introducida es la que sigue: $P = \{\{2, 3, 9\}, \{4, 6, 7\}, \{5, 8\}, \{10\}\}$.

```
unions<-matrix(c(1,1, 2,2, 3,2, 4,3, 5,4, 6,3, 7,3, 8,4, 9,2, 10,5),
byrow=T, ncol = 2)

maFolk(nodes,arcs,unions)

## Cost allocation rule: Folk
##       Agent Cost
##  [1,]     2  2.0
##  [2,]     3  2.5
##  [3,]     4  5.0
##  [4,]     5  1.5
##  [5,]     6  4.0
##  [6,]     7  4.0
##  [7,]     8  2.0
##  [8,]     9  4.0
##  [9,]    10  4.0
```

En este problema específico y con esta partición en concreto, vemos que el reparto de los costes entre los agentes difiere del problema en el que no se contemplaban grupos. Sabemos que esto no siempre es así y que dependerá principalmente del coste de los arcos y de como se hayan formado los grupos. Como vimos en el ejemplo previo, hay casos en que se dan repartos diferenciados en función de los grupos que se formen dentro de un mismo problema.

Para terminar con esta parte dedicada a la herramienta de programación, se adjunta el tiempo de ejecución de este problema con diez nodos y cuatro grupos de agentes diferentes. Vemos que el tiempo sigue siendo muy bueno, proporcionando el resultado de manera inmediata.

```
# Tiempo de ejecución
system.time(maFolk(nodes,arcs,unions))

## Cost allocation rule: Folk
##    user  system elapsed
##    0.03    0.00    0.03
```

Con la herramienta maFolk se ha logrado mejorar los tiempos de obtención del reparto de costes para problemas de arborescencias sin grupos con respecto a la función maERO. A mayores, se aporta la opción de resolver problemas con grupos. Teniendo en cuenta que los problemas de árboles de mínimo coste son un tipo específico de problemas de arborescencias de mínimo coste, la aplicación de maFolk puede extenderse también a este tipo de problemas. Para poder trabajar con árboles de mínimo coste bastará con introducir los arcos y sus respectivos pesos dos veces, teniendo en cuenta los dos sentidos de los mismos.

Capítulo 6

Conclusiones y trabajo futuro

La intención a lo largo del presente trabajo ha sido la de realizar un estudio profundo, conciso y práctico acerca de los problemas de arborescencias de mínimo coste que, además, resulte útil y accesible a cualquier lector, con independencia de su formación y conocimientos previos del asunto. Siempre estuvo presente la idea de que finalmente pudiera servir tanto como introducción para aquel público ajeno a los mismos, como de guía rápida y esquemática a aquellas personas familiarizadas con dichos problemas. Con este fin se han expuesto brevemente los orígenes de este tipo de problemas y se ha contextualizado su uso y aplicación a ciertas situaciones y procesos reales que facilitan su compresión y finalidad. Se ha tratado, además, que la introducción de definiciones sea de palabra, de manera que resulte fácilmente comprensible, acompañadas siempre que sea posible de su expresión matemática para no perder rigurosidad.

Una vez llegados a la parte específica para cada uno de los tipos de problemas se ha procurado una exposición esquemática y coherente a lo largo de los diferentes apartados del trabajo. Cada aplicación práctica de un algoritmo o cálculo de un juego cooperativo o regla de reparto ha sido acompañada paso a paso por grafos de apoyo para facilitar la interpretación de los mismos. Al adoptar estas medidas creemos estar cumpliendo con el objetivo de realizar una revisión de los problemas de arborescencias de mínimo coste, así como de hacerlo de modo que los conceptos de accesibilidad y funcionalidad del trabajo se mantienen.

Ciertamente el grueso de este trabajo lo forman la revisión, exposición y aplicación de los problemas de arborescencias de mínimo coste, pero también hay que tener en cuenta la aportación para la obtención de la regla Folk mediante las etapas del algoritmo de Edmonds con grupos y la programación de la regla Folk en  para problemas con y sin grupos. Si bien es verdad que la programación de dicha regla para problemas sin grupos no resulta novedosa en cuanto a la obtención de un resultado, ya que la función maERO ya calculaba dicha regla de reparto, sí ha supuesto una notable mejora en cuanto a tiempos de ejecución computacional. Creemos que se trata de un resultado positivo y de una mejora a tener en cuenta a la hora de trabajar con problemas complejos y de gran tamaño. Por otro lado, la posibilidad de obtener el reparto de costes cuando existe una estructura grupal sí que supone una nueva aportación de gran utilidad para este tipo de problemas. Nótese, además, que por extensión esta función sirve también para resolver problemas de árboles de mínimo coste, con y sin grupos, con la fluidez de un algoritmo polinomial.

En cuanto a la posibilidad de continuar con esta propuesta de trabajo, varias son las opciones que parecen posibles. En lo relativo a la parte más teórica del trabajo, sería posible ampliarla de diversas maneras diferentes. Podría resultar de interés exponer otros algoritmos de búsqueda distintos o bien profundizar en los juegos cooperativos abordando sus propiedades y caracterización. Con relación a la parte de programación en cabría la posibilidad de implementar otras reglas de reparto que guarden

relación con este tipo de problemas.

Apéndice A

Función maFolk

```
library(optrees)

maFolk<-function(nodes,arcs,unions=NULL){

# Get Edmonds algorithm's steps data
mca <- getMinimumArborescence(nodes, arcs, stages.data = TRUE,
show.data = FALSE, show.graph = F )

# Number of stages of Edmonds algorithm
numStages <- mca$stages
i <- numStages; i

# Historical of matchesNodes
matchesNodesH <- matrix(, nrow = nodes, ncol = 0)

# Index of new matchesNodes
vectorIndexM <- vector()

# Result matrix where to sum the weights in each iteration
vectorzeros <- rep(0, length(nodes))
result <- matrix(c(nodes,vectorzeros),byrow = FALSE, ncol = 2)
colnames(result) <- c("Agent", "Cost")

superNode <- NULL

for (k in 1:i){
# Arcs with minimum weights for each iteration
prevArcs <- mca$stage[[k]]$mcArcs

# dupTails vector is used to check cost tie
dupTails <- vector()

for (t in 1:nrow(prevArcs)) {
# Check if the node tail is already in the dupTails vector
if (!(prevArcs[t,2] %in% dupTails)){ #Tail in prevArcs is not in dupTails
# If superNode is null we just sum the weights in the  last column of the
```

```
#result matrix (there is no division of the weights)
if((is.null(superNode))) {
result[prevArcs[t,2],2] <- result[prevArcs[t,2],2] + prevArcs[t,3]
# Add the new tail in the duptails vector
dupTails <- append(dupTails,prevArcs[t,2])
} else {
if (!is.null(unions)) {
# Get the original node number
vectorIndexM <- which(matchesNodesH[, length(matchesNodesH[1,])] == prevArcs[t,2])
vectorUnions <- vector()
vectorUnionsValues <- vector()
for (d in vectorIndexM) {
# Vector with group's number of the elements in vectorIndexM
vectorUnions <- append(vectorUnions, unions[d,2])
if (!unions[d,2] %in% vectorUnionsValues) {
# Vector with group's number of the elements in vectorIndexM without rep
vectorUnionsValues <- append(vectorUnionsValues, unions[d,2])
}
}
# Cost per group
weightPerGroup <- prevArcs[t,3]/length(vectorUnionsValues)
for (f in 1:length(vectorIndexM)) {
# Result whith groups
result[vectorIndexM[f],2] <- result[vectorIndexM[f],2] +
weightPerGroup/length(which(vectorUnions==vectorUnions[f]))
}
dupTails <- append(dupTails,prevArcs[t,2])

} else {
# Get the original node number
vectorIndexM <- which(matchesNodesH[, length(matchesNodesH[1,])] == prevArcs[t,2])
for (j in vectorIndexM) {
# Result without groups
result[j,2] <- result[j,2] + prevArcs[t,3]/length(vectorIndexM)
}
dupTails <- append(dupTails,prevArcs[t,2])
}
}
}
}

# Reset superNode to NULL
superNode <- NULL

# Gives the superNode in every stage
superNode <- mca$stage[[k]]$superNode
# Reciprocation between nodes
matchesNodes <- mca$stage[[k]]$matches

if(length(matchesNodesH[1,]) == 0) {
matchesNodesH <- matchesNodes
```

```
} else {
# Vector with all the matches for original nodes
vectorM <- vector()
for (m in 1:nrow(matchesNodesH)) {
vectorM <- append(vectorM,
matchesNodes[matchesNodesH[m,length(matchesNodesH[1,])],2])
}
if (!is.null(vectorM)) {
matchesNodesH <- cbind(matchesNodesH, vectorM)
}
vectorM <- NULL
}
}

# Matrix to store the corresponding costs for each agent
carFolk <- round((result[-1,]),2)
writeLines("Cost allocation rule: Folk")
return(carFolk)
}
```

Bibliografía

[1] Bahel E, Trudeau C (2017) Minimum incoming cost rules for arborescences. Social Choice and Welfare 49:287-314.

[2] Bergantiños G, Gómez-Rúa M (2010) Minimum cost spanning tree problems with groups. Econ Theory 43:227-262.

[3] Bergantiños G, Gómez-Rúa M (2015) An axiomatic approach in minimum cost spanning tree problems with groups. Annals of Operations Research 225:45-63.

[4] Bergantiños G, Gómez-Rúa M, Llorca N, Pulido M, Sánchez-Soriano J (2014) A new rule for source connection problems. European Journal of Operational Research 234:780-788.

[5] Bergantiños G, Lorenzo L (2004) A non-cooperative approach to the cost spanning tree problem. Mathematical methods of operations Research 59:393-403.

[6] Bergantiños G, Vidal-Puga J (2007a) The optimistic TU game in minimum cost spanning tree problems. International Journal of Game Theory 36:223-239.

[7] Bergantiños G, Vidal-Puga J (2007b) A fair rule in minimum cost spanning tree problems. Journal of Economic Theory 137:326-335.

[8] Bergantiños G, Vidal-Puga J (2011) The folk solution and Boruvkas algorithm in minimum cost spanning tree problems. Discrete Applied Mathematics 159:1279-1283.

[9] Bergantiños G, Vidal-Puga J (2021) A review of cooperative rules and their associated algorithms for minimum-cost spanning tree problems. SERIEs 12:73-100.

[10] Bird CG (1976) On cost allocation for a spanning tree: a game theoretic approach. Networks 6:335-350.

[11] Boruvka O (1926) About a certain minimal problem (in zech). Praca Moravske Prirodovedecke Spolecnosti 3:37-58.

[12] Chu YJ, Liu TH (1965) On the shortest arborescence of a directed graph. Science Sinica 14:1396-1400.

[13] Dutta B, Kar A (2004) Cost monotonicity, consistency and minimum cost spanning tree games. Games and Economic Behavior 48:222-248.

[14] Dutta B, Mishra D (2012) Minimum cost arborescences. Games and Economic Behavior 74:120-143.

[15] Edmonds J (1967) Optimum branchings. Journal of Research of the National Bureau of Standards 71B:233-240.

[16] Feltkamp V, Tijs S, Muto S (1994) On the irreducible core and the equal remaining obligation rule of minimum cost extension problems. Mimeo. Tilburg University.

[17] Fontenla, M (2014a) optrees: Optimal Trees in Weighted Graphs. R package version 1.0, https://CRAN.R-project.org/package=optrees.

[18] Fontenla, M (2014b) cooptrees: Cooperative aspects of Optimal Trees in Weighted Graphs. R package version 1.0, https://CRAN.R-project.org/package=cooptrees.

[19] Fontenla, M (2014c) Optimal Trees Programación en R de problemas de búsqueda de árboles óptimos. Trabajo fin de Master, Universidade da Coruña.

[20] Kruskal, J (1956) On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society 7:48-50.

[21] Prim, RC (1957) Shortest connection networks and some generalizations. Bell Systems Technology Journal 36:1389-1401.

[22] Owen, G (1977) Values of games with a priori unions. En: Henn RO, Moeschlin O (eds) Mathematical Economics and Game Theory. Lecture Notes in Economics and Mathematical Systems, vol 141. Springer, Berlin, pp 76-88.

[23] Shapley, LS (1953) A Value for n-person Games. En: Kuhn HW, Tucker AW (eds) Contributions to the Theory of Games II. Princeton University Press, Princeton, pp 307-317.

[24] von Neumann J, Morgenstern O (1944) Theory of Games and Economic Behaviour. Princeton University Press, Princeton, NJ.

Índice de figuras

Índice de cuadros