Trabajo Fin de Máster

Penalizaciones en proyectos

Alfredo Manuel Amado Castro

Máster en Técnicas Estadísticas

Curso 2018-2019

Propuesta de Trabajo Fin de Máster

Título en galego: Penalizacións en proxectos
Título en español: Penalizaciones en proyectos
English title: Penalties in projects
Modalidad: Modalidad A
Autor/a: Alfredo Manuel Amado Castro, Universidade de Vigo
Director/a: Leticia Lorenzo Picado, Universidade de Vigo
Breve resumen del trabajo: A la hora de realizar un proyecto es muy habitual que en el contrato, firmado con las distintas empresas involucradas en el mismo, figure una penalización para las empresas en el caso de que haya retrasos. Este tipo de penalizaciones ya aparecen recogidas en el BOE en lo que a proyectos con la administración pública se refiere. En el presente trabajo se pretende hacer un estudio del cálculo de la penalización óptima que se ha de establecer en el contrato.
Recomendaciones: Haber cursado la materia Redes y Planificación.

Doña Leticia Lorenzo Picado, Categoría 1 de la Universidade de Vigo, informa que el Trabajo Fin de Máster titulado

Penalizaciones en proyectos

fue realizado bajo su dirección por don Alfredo Manuel Amado Castro para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, da su conformidad para su presentación y defensa ante un tribunal.

En Santiago de Compostela, a 1 de julio de 2019.

La directora:	El autor:
Doña Leticia Lorenzo Picado	Don Alfredo Manuel Amado Castro

Agradecimientos

A mis padres. Por su apoyo y comprensión.

Índice general

Resumen

Resumen en español

Motivamos este trabajo a partir de varias noticias de prensa, acerca de diferentes retrasos de obras públicas en España y el extranjero. Introducimos las leyes que regulan dichos retrasos.

En un primer capítulo, presentamos el método PERT/CPM que permite programar un calendario para la realización de un proyecto y de las actividades que lo constituyen, estimar su duración y permitir conocer si las actividades tendrán la capacidad de retrasarse sin que esto afecte al conjunto del proyecto.

Posteriormente, planteamos la gestión de los retrasos que puedan producirse en un proyecto mediante teoría de juegos. Para ello realizamos una revisión bibliográfica. Utilizando juegos cooperativos, presentamos diversas reglas para el reparto de los costes asociados a los retrasos producidos. Por contra, mediante un planteamiento no cooperativo, estudiamos las estrategias óptimas que deberán seguir los diferentes jugadores involucrados (planificador del proyecto y empresas encargadas de la realización de las actividades) para maximizar su beneficio.

Haciendo uso de lo estudiado, construimos un código en lenguaje de programación R que permita obtener la planificación de un proyecto y establecer las estrategias óptimas de los jugadores, en un planteamiento no cooperativo del problema de retrasos.

Por último, proponemos líneas de estudio futuras para el planteamiento no cooperativo del problema de retrasos.

English abstract

We motivate this work from several press news, about different delays of public works in Spain and abroad. We introduce the laws that regulate these delays.

In a first chapter, we present the PERT/CPM method that allows to schedule a calendar for the realization of a project and the activities that involve it, estimate its duration and allow knowing if the activities will have the capacity to delay without affecting the whole project.

Subsequently, we propose the management of delays that may occur in a project through game theory. For that aim, we carried a literature review. Using cooperative games, we present several rules for the distribution of the costs associated with the delays using cooperative games. By contrast, through a non-cooperative approach, we study the optimal strategies that the different players involved (project planner and companies in charge of carrying out the activities) should follow to maximize their benefit.

Making use of what has been studied, we build a code in the programming language R that allows us to obtain the planning of a project and establish the optimal strategies of the players, in a non-cooperative approach to the problem of delays.

Finally, we propose future study lines for the non-cooperative approach to the problem of delays.

Prefacio

Todos los grandes proyectos que ha realizado la humanidad a lo largo de su historia parecen inconcebibles sin un plan previo que organizase los recursos y las actividades en las que consistían dichos proyectos. Así, la historia de la gestión de proyectos se puede remontar a la construcción de las pirámides, a las grandes obras romanas o a las grandes catedrales europeas.

A pesar de esta larga historia, durante la primera mitad del siglo XX, la única herramienta disponible para la planificación y programación de proyectos complejos era el diagrama de Gantt. Las primeras versiones de esta herramienta datan de finales del siglo XIX, de la mano de Karol Adamiecki; pero no fue hasta los años 1910-1915 que Henry Gantt desarrolló el gráfico que lleva su nombre.

El diagrama de Gantt es una herramienta gráfica que permite representar el tiempo a dedicar a cada actividad a lo largo de un período temporal. A pesar de su utilidad, el diagrama de Gantt no especifica las relaciones existentes entre las diferentes actividades que incluye un proyecto, por lo que para la planificación de proyectos complejos que constan de más de 25 actividades resulta necesario el uso de otras técnicas.

No es hasta la segunda mitad del siglo XX que comienzan a surgir otras técnicas para la gestión de proyectos, principalmente los métodos PERT y CPM. Los primeros desarrollos del método CPM (Critical Path Method, o método del camino crítico en español) fueron realizados por las empresas inglesas "Imperial Chemical Industries" y "Central Electricity Generating Board" entre los años 1955 y 1957. Con ello, consiguieron reducir en torno a un 40 % el tiempo previsto en diferentes proyectos que llevaron a cabo. Como consecuencia de que estas empresas no publicaron dichas innovaciones, los avances en materia de gestión de proyectos tuvieron lugar al otro lado del océano Atlántico, en Estados Unidos.

También en los años 50, la compañía química DuPont estaba estudiando qué uso darle a los primeros ordenadores comerciales. Concluyeron que el mejor uso podía ser el planificar, estimar y programar, por lo que se asignó a la tarea de crear una técnica para ello al matemático J. E. Kelly, que trabajaba en la empresa de maquinaria de oficina Remington Rand, y a M. R. Walker, de la propia DuPont. El equipo de investigadores dirigido por estos dos hombres consiguió poner a punto el método CPM. En el año 1957, DuPont planeó la ampliación de casi 300 fábricas, lo cual suponía una cantidad de actividades (cerca de 30000) inabarcable con los diagramas de Gantt, por lo que el uso del método CPM fue esencial para controlar y optimizar los costes del proyecto de ampliación empresarial.

Casi paralelamente al desarrollo del método CPM, y también en Estados Unidos, se crea el método PERT (Program Evaluation Research Task), a cargo de la oficina de proyectos especiales de la marina de Estados Unidos, en colaboración con la empresa fabricante de proyectiles balísticos Lockheed y la consultoría de ingenieros Booz, Alien and Hamilton. El objetivo consistía en conseguir un método de gestión de proyectos que permitiese planificar, programar y controlar el proyecto de construcción de submarinos atómicos armados con proyectiles "Polaris". El proyecto "Polaris" tenía una duración de cinco años, e involucraba a 250 empresas, 9000 subcontratistas y numerosas agencias gubernamentales. Tal fue el éxito del uso del método PERT que el proyecto se adelantó dos años sobre los cinco previstos

inicialmente. Debido al éxito de esta primera aplicación del método, en 1959, los miembros del equipo de investigación encargado del desarrollo del método PERT, D. G. Malcolm, J. H. Roseboom, C. E. Clark y W. Fazar, publican el primer artículo sobre el método PERT al que renombran como Program Evaluation and Review Technique (técnica de revisión y evaluación de programas).

Ambos métodos, CPM y PERT, en origen eran diferentes; mientras el método PERT considera que el tiempo de realización de las diferentes actividades depende de una variable aleatoria, el método CPM supone que dichos tiempos dependen exclusivamente de la cantidad de recursos utilizados, siendo conocidos de antemano de forma determinista. A pesar de esta diferencia, y otras lógicas debido a su diferente origen, como las de notación, ambas técnicas son muy similares, por lo que con el tiempo han dado lugar a un único método.

PERT/CPM se basan en diagramas de redes que identifican las diferentes relaciones entre las actividades que se pretenden programar. Además, gracias a estas redes podemos establecer el calendario óptimo para el proyecto y sus actividades, y establecer el camino crítico de este. Un camino crítico consiste en la secuencia de actividades que se desarrollan desde el inicio hasta el final del proyecto y marcan su duración, por lo que cualquier retraso en alguna de ellas producirá un retraso del proyecto.

Hoy en día, los modelos PERT/CPM son frecuentemente usados para la gestión de grandes proyectos que incluyen una gran cantidad de actividades. Algunos ejemplos son los proyectos de construcción, ingeniería, desarrollo de software o productos... Ámbitos en los que resulta imprescindible la realización de un plan previo a la puesta en marcha del proyecto.

A pesar de la revolución que supuso el método PERT/CPM en la planificación de proyectos, en la práctica es común que existan retrasos. Estos retrasos pueden ser causados por diversos factores, tanto externos como de las propias empresas, que pueden obtener un beneficio al no dedicar todos los recursos necesarios a la realización de las actividades por derivarlos a otros proyectos. Debido a esto, es cada vez más usual que los contratos incluyan cláusulas que marquen unas penalizaciones monetarias cuando los proyectos incurren en un retraso.

La inclusión de penalizaciones en los contratos deriva en nuevos problemas para las empresas responsables del proyecto. Principalmente el reparto de los costes de la penalización entre las empresas encargadas de las actividades que han ocasionado el retraso, y la elaboración de estrategias que definan para cada empresa cuantos recursos dedicar a la actividad que deben realizar. Para modelar estos problemas será necesario el uso de teoría de juegos, en particular de los juegos cooperativos para repartir los costes y de los juegos no cooperativos para definir las estrategias de cada empresa.

John von Neumann y Oskar Morgenstern sentaron las bases de la teoría de juegos con su libro Theory of Games and Economic Behavior (1944), que a lo largo del siglo XX ha sido desarrollada y ampliada por otros autores como Nash, Shapley, Selten, Harsanyi... y ha tomado una gran importancia en diversos campos como la biología, la sociología o las ciencias políticas, y principalmente en la economía. Prueba de esto son los premios Nobel de esta última disciplina que han sido entregados por los avances en este campo.

La teoría de juegos es una rama de las matemáticas que permite la modelización de problemas donde interactúan varios jugadores que deben tomar decisiones, ya sea de forma cooperativa, o no cooperativa. En un modelo cooperativo, los jugadores podrán formar coaliciones y tomar acuerdos vinculantes previos al juego con el objetivo de maximizar sus beneficios. El problema se centrará en determinar el peso que cada jugador tiene para realizar un reparto de los beneficios más justo y que beneficie a todos. En los modelos no cooperativos o competitivos, en cambio, se prohíbe expresamente cualquier acuerdo previo. El estudio se centrará en las diferentes estrategias de cada jugador que intentará maximizar la función de pagos que dependerá de las estrategias del conjunto de los jugadores.

Capítulo 1

Retrasos en proyectos

En este capítulo presentaremos el método PERT/CPM para la planificación y programación de un proyecto, que será ilustrado mediante un ejemplo. Para ello será necesaria la introducción de conceptos como los caminos, que marcarán el tiempo necesario para realizar el proyecto, o las holguras, que nos permitirán una relativa flexibilidad a la hora de planificar determinadas actividades. Finalmente llegaremos a un calendario para el proyecto que representaremos mediante un diagrama de Gantt.

En la parte final del capítulo motivaremos el estudio de los retrasos en proyectos, mediante ejemplos reales, y comentaremos cómo se tratan estos retrasos desde las administraciones públicas, concretamente desde el Gobierno de España.

1.1. Calendario inicial de un proyecto

Un modelo PERT/CPM utiliza una red para representar las diferentes relaciones entre las actividades que constituyen un proyecto y así poder diseñar la planificación y programación del mismo. Estas relaciones pueden ser de precedencia, cuando una actividad debe ser realizada antes que otra, o paralela, si ambas actividades pueden realizarse simultáneamente y no dependen una de la otra. Es fácil pensar en ejemplos de actividades que precisan ser realizadas antes que otras y de actividades que pueden realizarse simultáneamente, por ejemplo no podemos construir el tejado de una casa si previamente no está construida la estructura, pero sí que podemos instalar el sistema de fontanería al mismo tiempo que el eléctrico.

La red utilizada por el modelo está constituida por un conjunto de arcos, que representarán las diferentes actividades, que denotaremos por $N = \{1, 2, .., n\}$, y un conjunto de nodos que representarán el inicio y final de las actividades. Para cada arco $i \in N$, b_i y e_i denotan los nodos inicial y final de la actividad i respectivamente. Estos nodos, en general, no serán únicos y cada uno puede ser el final o el inicio de varios arcos. De esta forma una actividad que se inicia en un determinado nodo no puede comenzar hasta que todas las actividades que finalizan en dicho nodo hayan sido completadas. Debido a esto, no habrá ciclos en la red.

Consideraremos dos nodos especiales, el nodo inicio u origen, b, y el nodo final, e, que marcan el respectivo inicio y final del proyecto. Por lo que no habrá ninguna actividad que finalice en el nodo inicio ni ninguna que comience en el nodo final.

Debido a la construcción de la red, en algunas situaciones tendremos que usar actividades auxiliares, que llamaremos ficticias, para mostrar que una actividad debe realizarse antes que otra. Por

definición estas actividades tendrán un tiempo de realización cero, y su uso es exclusivo para reflejar estas relaciones entre actividades.

Denotaremos por t_i la duración de la actividad i. Esta duración puede ser aleatoria o determinista, según estemos tratando con modelo PERT o CPM. Aunque la duración sea aleatoria trabajaremos igualmente con un número fijo, que calcularemos de la siguiente forma:

$$t_i = \frac{(O + 4M + P)}{6}$$

Donde O es el tiempo mas optimista, M el mas probable suponiendo que todo sea normal y P es el tiempo mas pesimista (excluyendo grandes catástrofes).

Una de las críticas que recibe el método PERT proviene del cálculo de estos tiempos, pues en la práctica suele resultar más optimista de lo que finalmente es.

Ejemplo 1.1 *Cuando modelamos un proyecto, inicialmente la información necesaria puede venir dada por un cuadro, como el Cuadro 1.1, que ofrezca, para cada actividad, las actividades predecesoras (que debemos finalizar previamente) y su duración.*

Actividad	Predecesoras	Duración
A	–	2
B	A	4
C	B	3
D	B	2
E	C,D	10
F	B	4
G	F	2
H	E,G	1

Cuadro 1.1: Actividades predecesoras y duraciones

En el cuadro tenemos todos los datos necesarios para la construcción del modelo y la posterior planificación de las actividades. Construimos la red atendiendo a las actividades y sus predecesoras.

Como vemos en la Figura 1.1, ha sido necesario la introducción de una actividad ficticia, a la que se le ha dado una duración de 0, y que no influirá en la planificación final del proyecto.

Definimos un camino π del nodo a al nodo c como la sucesión de actividades (arcos) consecutivas de a a c. Formalmente, dada una red y dos nodos a y c, un camino entre ambos nodos es una colección de arcos, $\{i_m\}_{m=1}^{s}$, que satisface las siguientes condiciones:

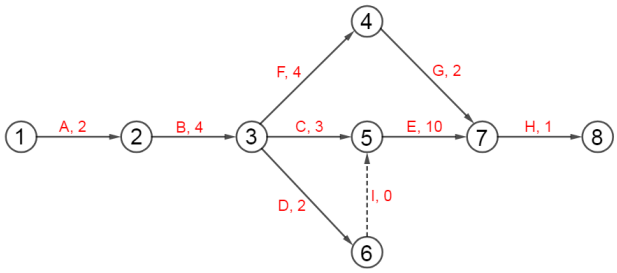

Figura 1.1: Grafo del problema

- El nodo inicio del arco i_1 es $b_{i_1} = a$.
- Para cada $m = 1, .., s-1$, el nodo final del arco i_m coincide con el nodo inicio del arco i_{m+1}, $e_{i_m} = b_{i_{m+1}}$.
- El nodo final del arco i_s es $e_{i_s} = c$.

Denotamos por Π al conjunto de todos los caminos existentes en la red. La duración del camino π, t_π, vendrá dada por la duración de las actividades que lo componen:

$$t_\pi = \sum_{i \in \pi} t_i$$

Una clase de caminos importante será los caminos críticos. Diremos que un camino es crítico cuando no exista otro camino con una duración mayor. Este camino no tiene por qué ser único. Además, es trivial observar que un camino crítico será necesariamente un camino entre los nodos b y e. De aquí en adelante, al hablar de caminos nos restringiremos únicamente a los caminos entre los nodos b y e.

El tiempo mínimo de realización del proyecto, T, se corresponde con la duración del camino, o caminos, críticos:

$$T = \max_{\pi \in \Pi} t_\pi$$

Ejemplo 1.2 *Siguiendo con el proyecto dado en el Ejemplo 1.1, calculamos sus caminos críticos y la duración total del proyecto.*

Sabemos que un camino crítico, necesariamente, es un camino entre los nodos origen y final de la red. En este caso, solo hay tres posibles caminos entre los nodos b y e. Sus duraciones son de 13, 20 y 19, por lo que el camino crítico será el de duración $t_\pi = 20$.

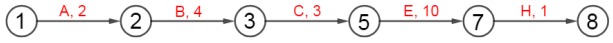

Figura 1.2: Camino crítico.

Ya tenemos calculado el camino crítico, que en este caso es único, cuya duración determina la del proyecto. Por lo tanto $T = 20$.

En consecuencia, todo retraso en alguna actividad de uno de estos caminos resultará en un retraso del propio proyecto. Así pues, las actividades que conforman estos caminos críticos serán llamadas a su vez actividades críticas, pues todo retraso de estas actividades causará un retraso del proyecto.

Las actividades críticas serán las que supongan una mayor atención por parte del planificador, mientras que las actividades no críticas (aquellas que no pertenecen a ningún camino crítico) tendrán un margen para finalizar con retraso sin causar un perjuicio a la duración general del proyecto. Para estudiar este margen, que llamaremos holgura, debemos definir previamente algunos conceptos.

Definición 1.1 *Para cada nodo a definimos el tiempo más temprano de comienzo de una actividad, o simplemente tiempo más temprano, que denotaremos* $t^e(a)$*, como el menor tiempo que tardan en finalizar todas las actividades que terminan en ese nodo. Dicho de otro modo, es el tiempo del camino más largo hasta el nodo.*

Definición 1.2 *Para cada nodo a definimos el tiempo más tardío de comienzo de una actividad, o simplemente tiempo más tardío, que denotaremos* $t^l(a)$*, como el último instante de tiempo en el que podemos iniciar las actividades que comienzan en este nodo sin causar un retraso en el proyecto. Dicho de otro modo, es la duración total del proyecto menos el tiempo del camino mas largo desde el nodo a hasta el nodo final de la red.*

El tiempo más temprano de un nodo será siempre igual o menor que el tiempo más tardío del mismo nodo. En realidad, ambos tiempos únicamente coincidirán cuando el nodo sea un nodo crítico (pertenezca a un camino crítico).

Ejemplo 1.3 *Continuando con el proyecto presentado en el Ejemplo 1.1, obtenemos, en el Cuadro 1.2, los tiempos más temprano y más tardío de inicio para cada nodo.*

Vemos que, para cada nodo, el tiempo más tardío siempre es mayor o igual que el tiempo más temprano.

Definimos la holgura de la actividad i, as_i, como la máxima cantidad de tiempo que la actividad i puede retrasarse sin causar un retraso en el proyecto, asumiendo que el resto de actividades se completan en el tiempo estimado. De igual forma, definimos la holgura de un camino π, ps_π, como la máxima cantidad de tiempo que las actividades que lo conforman pueden retrasarse sin causar un retraso en el proyecto. Formalmente, definimos la holgura de un camino π y de una actividad i como:

$$ps_\pi = T - t_\pi$$

$$as_i = \min_{\pi \in \Pi : i \in \pi} ps_\pi$$

También podemos calcular la holgura de una actividad i a partir de los tiempos más temprano y más tardío de inicio de sus nodos origen y final:

$$as_i = t^l(e_i) - t^e(b_i) - t_i$$

Podemos ver fácilmente que una actividad será crítica si y solo si su holgura es cero y que lo mismo sucederá con los caminos críticos.

Nodo	$t_e(\cdot)$	$t_l(\cdot)$
1	0	20-20=0
2	2	20-18=2
3	6	20-14=6
4	10	20-3=17
5	9	20-11=9
6	8	20-11=9
7	19	20-1=19
8	20	20-0=20

Cuadro 1.2: Tiempos más temprano y más tardío de inicio para cada nodo.

Ejemplo 1.4 *Calculamos, en el Cuadro 1.3, la holgura de cada actividad del ejemplo que ya hemos introducido. Lo hacemos utilizando la fórmula* $as_i = t^l(e_i) - t^e(b_i) - t_i$ *y los resultados obtenidos en el Cuadro 1.2, donde obtenemos los tiempos más temprano y más tardío para cada nodo.*

Las actividades críticas, como era de esperar, tienen una holgura igual a cero.

Dadas dos actividades $i, j \in N$, decimos que i precede a j, y denotamos $i \prec j$, si la actividad i precisa ser completada antes del inicio de la actividad j, y consecuentemente diremos que j sucede a i. Definimos las actividades que preceden a la actividad i en el camino π como las actividades pertenecientes al camino que deben ser completadas para que la actividad i pueda comenzar y las denotaremos por $Pre(i, \pi)$. De la misma forma, definimos las actividades que suceden a la actividad i en el camino π como aquellas pertenecientes al camino π que precisan de su finalización para comenzar y las denotaremos por $Suc(i, \pi)$. Formalmente:

$$Pre(i, \pi) = \{j \in \pi : j \prec i\},$$

$$Suc(i, \pi) = \{j \in \pi : i \prec j\}.$$

Al conjunto de todas las actividades que preceden a i lo denotaremos por $Pre(i)$, y al conjunto de todas las actividades que suceden a i lo denotaremos $Suc(i)$. Formalmente:

$$Pre(i) = \bigcup_{\pi \in \Pi : i \in \pi} Pre(i, \pi),$$

$$Suc(i) = \bigcup_{\pi \in \Pi : i \in \pi} Suc(i, \pi).$$

En general, para programar las actividades que componen un proyecto es necesario tener en cuenta varios principios:

- La actividad i no puede iniciarse antes del tiempo más temprano de su nodo inicial, $t^e(b_i)$. De otra manera, las actividades que preceden a i podrían no haber finalizado.

Actividad	as_i
A	2-0-2=0
B	6-2-4=0
C	9-6-3=0
D	9-6-2=1
E	19-9-10=0
F	17-6-4=7
G	19-10-2=7
H	20-19-1=0
I	9-8-0=1

Cuadro 1.3: Holguras de las actividades.

- La actividad i no puede iniciarse después del tiempo más tardío de su nodo final menos el tiempo de la actividad, $t^l(e_i) - t_i$. De otra manera, el proyecto podría retrasarse puesto que i finalizaría después del tiempo más tardío de e_i.
- La actividad i no puede terminar antes del tiempo más temprano de su nodo inicial más el tiempo de la actividad, $t^e(b_i) + t_i$. De otra manera, las actividades que preceden a i podrían no haber finalizado.
- La actividad i no puede terminar después del tiempo más tardío de su nodo final, $t^l(e_i)$. De otra manera, el proyecto podría retrasarse puesto que i finalizaría después del tiempo más tardío de e_i.

Teniendo en cuenta estos cuatro puntos, programamos las actividades diferenciando entre las que son críticas y las que no. Si la actividad i es crítica, entonces debe iniciarse en el tiempo más temprano de su nodo inicial y finalizar en el tiempo más temprano de su nodo final. En cambio, si la actividad i es no crítica, debe iniciarse entre $t^e(b_i)$ y $t^e(b_i)+as_i$, y finalizar entre $t^l(e_i)-as_i$ y $t^l(e_i)$. Además, la diferencia entre el inicio y el final de una actividad no puede ser menor que el tiempo de la propia actividad.

Ejemplo 1.5 *Calculamos ahora los tiempos de inicio y final posibles para cada actividad del proyecto enunciado en el Ejemplo 1.1.*

Para cada actividad i no crítica, su inicio pertenecerá al intervalo $[t^e(b_i), t^e(b_i) + as_i]$ *y su final al intervalo* $\left[t^l(e_i) - as_i, t^l(e_i)\right]$.

Las actividades críticas tienen establecido su tiempo inicial y final sin posibilidad de ningún margen de retraso, como vemos en el Cuadro 1.4. Sin embargo las actividades no críticas poseen un pequeño margen para comenzar o finalizar antes o después, lo que ayudará a minimizar los posibles retrasos.

Actividad	Inicio	Final
A	0	2
B	2	6
C	6	9
D	[6, 7]	[8, 9]
E	9	19
F	[6, 13]	[10, 17]
G	[10, 17]	[12, 19]
H	19	20
I	[8, 9]	[8, 9]

Cuadro 1.4: Tiempos de inicio y final para las actividades.

Con los criterios anteriores, realizamos la programación de las actividades, asignando a cada una un intervalo de tiempo para su realización. Esta asignación se hará minimizando la probabilidad de retraso del proyecto y asumiendo que habrá un coste si se incurre en un retraso. Para ello, al programar las actividades no críticas (los tiempos de las actividades críticas ya están definidos de forma determinada), se tratará de dividir el tiempo extra de una manera justa.

Ejemplo 1.6 *Finalmente, y continuando con el proyecto introducido en el Ejemplo 1.1, programamos las actividades procurando distribuir el tiempo extra de las actividades no críticas de manera que se minimice la probabilidad de retrasar el proyecto y teniendo en cuenta que el tiempo entre el inicio y el final de cada actividad debe ser igual o mayor a la duración de la misma. Ilustramos los repartos finales de tiempos para cada actividad con el diagrama de Gantt representado en la Figura 1.3.*

1.2. Retrasos en España

En los modelos PERT/CPM debemos tener en cuenta la posibilidad de que las empresas contratistas incurran en retrasos, ya sea por causas que no pueden controlar como errores en la ejecución, malas estimaciones, factores externos, etc. o por no adjudicar todos los recursos necesarios a la ejecución de las actividades.

Los modelos PERT/CPM procuran minimizar estos retrasos pero, además, se suelen intentar evitar a partir de cláusulas contractuales entre el planificador y las empresas que impongan penalizaciones cuando se incurra en retrasos. Estas penalizaciones son comunes tanto en el ámbito privado como en el público, donde suelen estar establecidas por ley.

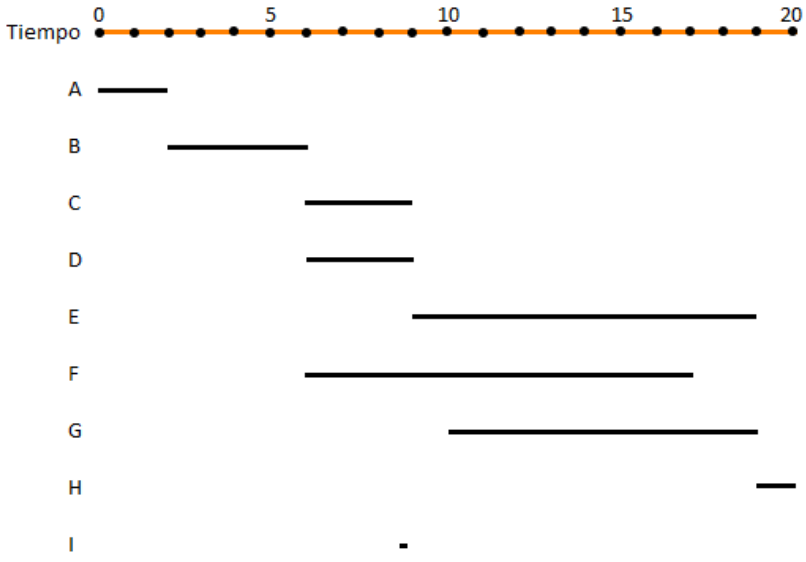

Figura 1.3: Diagrama de Gantt del proyecto.

Los retrasos en proyectos de la administración pública son comunes en prácticamente todos los países. Por ejemplo, un informe del Ministerio de Medio Ambiente y Construcción alemán (Palop, J., 2014) sobre los grandes proyectos que estaban siendo promovidos por el Gobierno federal, a fecha de 2014, cifraba en 14, sobre un total de 40, los proyectos que no tenían un retraso en ese momento.

En el caso particular de España podemos encontrar multitud de ejemplos de proyectos retrasados por diversas razones y que han causado incuantificables costes tanto a la propia administración como a la sociedad y a las empresas derivados de la imposibilidad de uso del proyecto finalizado. Ejemplos claros de esto pueden ser los retrasos en las construcciones de vías de comunicación, que causan un claro perjuicio a todos sus potenciales usuarios. Los costes del retraso en la construcción del corredor mediterráneo suponen, según algunos cálculos, 200 millones de euros de sobrecoste a la exportación de cítricos de la Comunidad Valenciana (Félix, F., 2018).

Otros casos de retrasos en proyectos que han supuesto un coste a la administración de la que dependía su realización son la construcción del submarino S-80 (González, M., 2018) y el almacén temporal centralizado de residuos nucleares de Villar de Cañas (Vélez, A. M., 2017), ambos con un coste claro y perfectamente cuantificable para las arcas públicas. En lo correspondiente al proyecto de los submarinos S-80, cuyo retraso fue debido a errores en su fabricación, el coste por el retraso del proyecto asciende a 130 millones de euros derivados de la obligada revisión y puesta a punto de los submarinos que se encuentran todavía en servicio y que iban a ser sustituidos por los nuevos S-80. Por otro lado, el retraso en la construcción del almacén de residuos nucleares supone el pago directo a Francia de 67.000 euros al día por el almacenaje de los residuos.

A la vista de estos ejemplos de retrasos en proyectos y de los costes que generan tanto al estado como a la sociedad, resulta evidente la necesidad de que se penalice por ley a las empresas contratistas de los proyectos que no son finalizados en el tiempo estipulado. Según la Ley 3/2011, de 14 de noviembre, en España, se establece que “cuando el contratista, por causas imputables al mismo, hubiere incurrido en demora respecto al cumplimiento del plazo total, la Administración podrá optar indistintamente por la resolución del contrato o por la imposición de las penalidades diarias en la proporción de 0,20

euros por cada 1.000 euros del precio del contrato". Además, se estipula que cuando estas penalizaciones alcancen un múltiplo del 5 por 100 del precio del contrato, la administración estará facultada para proceder a su resolución o acordar su continuidad bajo nuevas penalizaciones. A pesar de estos criterios, la ley permite la inclusión de cláusulas con penalizaciones distintas cuando se considere necesario para la correcta ejecución del proyecto. Permite también, la inclusión de cláusulas para imponer penalizaciones, o la rescisión del contrato, cuando se produzca el incumplimiento de plazos parciales, en la misma medida que se estipula para el total del contrato.

Vemos, entonces, que la ley española establece un sistema de penalizaciones lineal sobre los retrasos en los proyectos, dando opción a ampliarlo a plazos intermedios del mismo, siempre que dichos retrasos sean causados por el adjudicatario del proyecto. De no ser así, la ley establece que se concederá una ampliación del plazo de finalización, a lo sumo, igual al retraso causado por razones externas.

Capítulo 2

Gestión de los retrasos usando teoría de juegos

Una vez realizada la planificación del proyecto, es muy común en la práctica que el calendario de actividades no sea cumplido, ya sea por causas externas impredecibles, por errores de planificación o de ejecución o porque alguna empresa obtiene un beneficio al no dedicar todos los recursos necesarios a la actividad. Como hemos visto anteriormente, estos retrasos son habituales y hasta la propia administración pública marca, por ley, una serie de penalizaciones cuando estos ocurren. Pero no solo los contratos públicos están sujetos a penalizaciones en caso de retrasos, los contratos privados, en general, también suelen incluir cláusulas para penalizar a las empresas contratadas que no cumplan con los plazos señalados en el proyecto. En este trabajo nos centraremos en los contratos públicos, por tener una regulación más uniforme, mientras que los privados están sujetos a los diferentes intereses de los contratantes.

En algunas ocasiones será beneficioso para el planificador incentivar que el proyecto finalice antes del tiempo planeado, para lo que podrá estipular una prima cuando así suceda. Estas bonificaciones por finalizar antes del tiempo previsto no están tan generalizadas como las penalizaciones, sobre todo en el ámbito público, por lo que también están menos estudiadas desde un punto de vista teórico.

Para el estudio del reparto de las penalizaciones, y de las mejores estrategias que deben seguir las empresas a la hora de adjudicar recursos a las actividades, resulta muy útil la teoría de juegos. La propia teoría de juegos se divide en dos grandes clases atendiendo a la posibilidad de cooperación, o no, de los jugadores involucrados, lo que nos permitirá estudiar el problema de reparto de costes en proyectos desde dos aproximaciones diferentes. Haciendo uso de los juegos cooperativos podemos estudiar los repartos de penalizaciones, o de recompensas, que más beneficien al conjunto de empresas una vez finalizado el proyecto, estableciendo así repartos que resulten beneficiosos para todas las empresas involucradas.

Por otra parte, gracias a los juegos no cooperativos, será posible modelar situaciones en las que las empresas, sin colaborar entre ellas, deban decidir la cantidad de recursos que dedican a su actividad y, en consecuencia, el retraso que generarán. Para elaborar estas estrategias que optimicen sus ganancias, cada empresa no solo tendrá que atender al beneficio que obtendrá si destina recursos a actividades en otros proyectos y al perjuicio que generan las penalizaciones, sino que además tendrá que actuar teniendo presentes las estrategias que utilizarán, a priori, o que ya han utilizado el resto de empresas y de si estas estrategias generarán un retraso en el proyecto.

El reparto de costes en proyectos mediante juegos cooperativos está ampliamente estudiado desde

diferentes perspectivas, veremos una aproximación a los estudios de Branzei, Ferrari, Fragneli y Tijs (2002), Bergantiños y Sánchez (2002) y Estévez-Fernández, Borm y Hamers (2007), que extiende estos problemas a los casos en los que se fija una compensación cuando el proyecto finaliza antes del tiempo estipulado.

Posteriormente, en la segunda parte de este capítulo, nos centraremos en los juegos no cooperativos asociados a los métodos PERT/CPM. En el trabajo de Bergantiños y Lorenzo (2017) se definen juegos no cooperativos en varias etapas, dependiendo de si la penalización sobre el retraso se aplica solo en caso de que el proyecto entero se retrase o de si existe una penalización por el retraso de cualquier actividad.

2.1. Planteamiento cooperativo

2.1.1. Introducción a los juegos cooperativos

Es común que cuando un conjunto de jugadores (individuos, empresas...) precisan repartir un beneficio (o un coste) entre ellos, deban llegar a acuerdos que sean beneficiosos para todos los implicados. En un primer momento, podemos pensar en diversos métodos de reparto, tales como dividir igualitariamente, de forma proporcional a las demandas de cada jugador... Pero estos repartos, en los que no tendríamos en cuenta las posibles alianzas, no siempre son los más justos, ni los que más ganancias generen, para los diferentes jugadores. Es por eso que utilizamos la teoría de juegos cooperativos que nos proporciona herramientas para optimizar los beneficiosos para el conjunto de jugadores.

Como ya hemos comentado, en un juego cooperativo los jugadores poseen mecanismos para establecer acuerdos vinculantes, por lo que pueden colaborar con el objetivo de obtener mejores acuerdos según sus propios intereses. Cuando un grupo de jugadores colabora diremos que se forma una coalición entre estos jugadores, que actuarán con el objetivo de maximizar el beneficio de la propia coalición. Una coalición podrá estar formada por cualquier grupo de jugadores, desde uno solo hasta el conjunto completo.

Cuando cualquier reparto entre los jugadores sea posible, o existan medios con los cuáles se puedan compensar mutuamente mediante la transferencia de utilidad, diremos que es un juego cooperativo con utilidad transferible, o TU. En los juegos asociados a problemas de retrasos en proyectos trataremos con repartos de tiempo, o dinero, que supondremos perfectamente divisible entre los jugadores, por lo que emplearemos juegos cooperativos con utilidad transferible.

Definiremos el conjunto de jugadores por $N = \{1, 2, ..., n\}$, que en nuestro caso particular será el conjunto de actividades, o de empresas que realizan dichas actividades. Al conjunto de todos los subconjuntos de N lo denotamos por 2^N, y a cada elemento de este conjunto lo llamaremos coalición.

Definición 2.1 *Un juego cooperativo con utilidad transferible, o juego TU, es un par* (N, v)*, siendo* N *un conjunto finito y* v *una función*

$$v : 2^N \longrightarrow \mathbb{R}$$

tal que $v(\emptyset) = 0$.

Llamaremos función característica del juego (N, v) a la función v, que asocia a cada coalición S el beneficio que puede garantizarse por sí misma, independientemente de cómo actúe el resto de jugadores. A su vez, denotaremos por G^N a la clase de todos los juegos con conjunto de jugadores N.

De forma natural, a partir de la definición del juego coalicional, surge la definición de subjuego.

Definición 2.2 *Sea $(N, v) \in G^N$. Un subjuego de (N, v) es un par (R, v_R) tal que $R \subset N$ y $v_R(R) = v(R)$ para cada $R \subset 2^R$.*

A pesar de que su definición se presenta normalmente de esta forma, los juegos cooperativos no modelizan, en general, situaciones en las que se reparte un beneficio. También pueden modelizar problemas en los que se trata el reparto de costes. Cuando estamos en esta situación podemos obtener, asociado al juego de costes (N, c), el juego (N, v_c) que indicará el ahorro de cada coalición en la que sus componentes han decidido cooperar. Matemáticamente obtenemos este juego de la siguiente forma:

$$v_c(S) = \sum_{i \in S} c(i) - c(S), \text{ para todo } S \in 2^N.$$

Dado que podemos identificar fácilmente un juego de costes con uno de beneficios, podemos presentar las definiciones y resultados correspondientes centrándonos únicamente en los juegos cooperativos en los que se reparte un beneficio.

Definiciones básicas

Presentamos a continuación unas definiciones y resultados básicos de los juegos cooperativos, que nos serán útiles para caracterizar las soluciones que presentaremos a continuación.

Definición 2.3 *Sea $(N, v) \in G^N$. Diremos que (N, v) es*

- *Superaditivo si $v(S \cup T) \geq v(S) + v(T)$ para todo $S, T \subset N, S \cap T = \emptyset$.*
- *Subaditivo si $v(S \cup T) \leq v(S) + v(T)$ para todo $S, T \subset N, S \cap T = \emptyset$.*
- *Aditivo si $v(S \cup T) = v(S) + v(T)$ para todo $S, T \subset N, S \cap T = \emptyset$.*

Es sencillo observar que en un juego superaditivo los jugadores tendrán incentivos para formar coaliciones y, en última instancia, la coalición total. Es por ello que la subclase de los juegos superaditivos ha sido particularmente estudiada.

Definición 2.4 *Sea $(N, v) \in G^N$. Diremos que (N, v) es*

- *Convexo si $v(S) + v(T) \leq v(S \cup T) + v(S \cap T)$ para todo $S, T \subset N$.*
- *Estrictamente convexo si $v(S) + v(T) < v(S \cup T) + v(S \cap T)$ para todo $S, T \subset N$.*
- *Cóncavo si $(N, -v)$ es convexo.*
- *Estrictamente cóncavo si $(N, -v)$ es estrictamente convexo.*

Proposición 2.1 *Todo juego superaditivo es convexo.*

Proposición 2.2 *$(N, v) \in G^N$ es convexo si y solo si $v(S \cup i) - v(S) \geq v(T \cup i) - v(T)$ para todo $S, T \in 2^N$ tal que $T \subset S \subset N \setminus i$.*

Soluciones para un juego cooperativo

Como explicamos en la anterior sección, el objetivo implícito de los juegos superaditivos es que se forme la gran coalición N y que los jugadores se repartan entre ellos el beneficio de la cooperación.

Definición 2.5 *Un reparto es un vector $x = (x_1, ..., x_n) \in \mathbb{R}^N$, donde cada elemento x_i, con $i \in N$, representa la asignación al jugador i. Dado un reparto $x = (x_1, ..., x_n) \in \mathbb{R}^N$, la suma de las cantidades asignadas a los miembros de una coalición $S \subset N$ se denota por $x(S) = \sum_{i \in S} x_i$.*

Un reparto, podría ser, por ejemplo, la asignación de todo el beneficio a un único jugador, o la asignación de la mitad del beneficio a partes iguales entre los jugadores. A priori, estos reparto no resultan los más convenientes, por lo que parece necesaria la imposición de requerimientos para establecer reglas de reparto. Los más comunes en la teoría de juegos son las propiedades de racionalidad individual y eficiencia. La primera de ellas propone que cada jugador debe obtener al menos lo que puede garantizarse por sí solo. Mientras que un reparto será eficiente si distribuye el valor de la coalición total, $v(N)$.

Un reparto eficiente que sea individualmente racional se denomina imputación. Formalmente:

Definición 2.6 *Sea $(N, v) \in G^N$. Una imputación del juego (N, v) es un reparto que satisface las dos siguientes condiciones:*

1. *$x_i \geq v(i)$, para todo $i \in N$.*
2. *$\sum_{i \in N} x_i = v(N)$.*

Denotaremos por $I(N, v)$ al conjunto de imputaciones del juego (N, v).

Definición 2.7 *Una solución definida en algún dominio $\Omega \subset G^N$ es una correspondencia $\phi :\longrightarrow \mathbb{R}^N$ que asocia a cada juego (N, v) en Ω un subconjunto $\phi(N, v) \subset \mathbb{R}^N$.*

El núcleo y conceptos relacionados

A continuación veremos los ejemplos de solución tipo conjunto y puntual más utilizadas: el núcleo de un juego (que será un conjunto) y el valor de Shapley (que será un reparto determinado).

Definición 2.8 *El núcleo de un juego (N, v) es el conjunto:*

$$C(N, v) = \{x \in I(N, v) : \sum_{i \in S} x_i = v(S) \ \forall S \in 2^N\}.$$

Los repartos del núcleo son eficientes, pues reparten el beneficio global de la coalición N. Además, garantizan que ninguna coalición S tendrá incentivos para desviarse y obtener un reparto mejor, ya que toda coalición obtiene como mínimo lo que puede garantizarse por sí misma, por lo que diremos que estas asignaciones son estables.

En el caso de un juego de costes (N, c), el núcleo del juego se define como el conjunto:

$$C(N, c) = \{x \in I(N, v) : \sum_{i \in S} x_i \leq c(S) \ \forall S \in 2^N\}.$$

El núcleo de un juego puede ser vacío. De no serlo, será un politopo convexo y compacto de $\mathbb{R}^N$. Dado que no sabemos si el núcleo del juego es vacío, gran parte del estudio teórico sobre esta solución se centra en determinar si es no vacía. Para ello, presentaremos a continuación una serie de definiciones y resultados que nos ayudaran a determinar esta cuestión.

Proposición 2.3 *Si el núcleo de un juego (N, v) es no vacío, $C(N, v) \neq \emptyset$, entonces $\sum_{i=1}^{k} v(P_i) \leq v(N)$, para toda partición $\mathcal{P} = \{P_1, ..., P_k\}$ de N.*

Definición 2.9 *Una familia $\mathcal{F}$ de coaliciones no vacías de N se dice equilibrada si existen unos pesos positivos $\{\delta_S\}_{S\in\mathcal{F}}$, $\delta_S > 0$ para todo $S \in \mathcal{F}$, tales que $\sum_{S\in\mathcal{F}} \delta_S e^S = e^N$, siendo $e^S \in \mathbb{R}^N$ el vector característico de S, es decir, $e_i^S = 1$ si $i \in S$ y $e_i^S = 0$ si $i \notin S$.*

Definición 2.10 *Un juego (N, v) se dice equilibrado si para toda familia $\mathcal{F}$ equilibrada, con pesos asociados $\{\delta_S\}_{S\in\mathcal{F}}$, se verifica que*

$$\sum_{S\in\mathcal{F}} \delta_S v(S) \leq v(N)$$

Definición 2.11 *Un juego (N, v) se dice totalmente equilibrado si todo subjuego (R, v_R), con $R \subset N$, $R \neq \emptyset$, es equilibrado.*

Proposición 2.4 *Todo juego totalmente equilibrado es superaditivo.*

Por último enunciamos el Teorema de Bondareva-Shapley, que nos ofrece una condición necesaria y suficiente para garantizar que el núcleo del juego es no vacío.

Teorema 2.1 ***(de Bondareva-Shapley)*** *El núcleo de un juego (N, v) es no vacío si y solo si el juego es equilibrado.*

Deducimos, por la Proposición 2.4 y este último teorema, que la propiedad de superaditividad es una condición necesaria para que el núcleo de un juego sea no vacío. Por lo que si un juego es no superaditivo su núcleo será vacío y no tendrá sentido el concepto de núcleo.

El valor de Shapley

El concepto de solución puntual más empleado en teoría de juegos cooperativos es el valor de Shapley, que definimos a continuación.

Definición 2.12 *El valor de Shapley de un juego cooperativo (N, v) es el reparto*

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

para todo $i \in N$, donde s y n denotan los cardinales de los conjuntos S y N, respectivamente.

Definimos a continuación una serie clases de jugadores que nos permitirán ofrecer una definición alternativa para el valor de Shapley basada en sus cuatro propiedades características.

Definición 2.13 *Un jugador $i \in N$ en un juego $(N, v) \in G^N$ se dice títere si no aporta ningún beneficio extra al formar coaliciones con el resto de jugadores, es decir, si $v(S \cup i) = v(S) + v(i)$ para todo $S \subset N \setminus i$.*

Si además verifica que $v(i) = 0$, se dice que es un jugador nulo.

Definición 2.14 *Sean dos jugadores $i, j \in N$ en un juego $(N, v) \in G^N$ se dice que ambos son simétricos si $v(S \cup i) = v(S \cup j)$ para todo $S \subset N \setminus \{i, j\}$.*

Teorema 2.2 *Sea ϕ una regla de reparto que verifica las siguientes propiedades:*

- *Eficiencia: Para todo $(N, v) \in G^N$, $\sum_{i \in N} \phi_i(N, v) = v(N)$. La propiedad de eficiencia especifica que en la regla de reparto la suma de los pagos de cada jugador debe coincidir con la del conjunto total N.*
- *Jugador nulo: Para todo $(N, v) \in G^N$ y para todo $i \in N$ jugador nulo, $\phi_i(N, v) = 0$. Esta propiedad verifica que si un jugador $i \in N$ no realiza ninguna aportación a las coaliciones, recibirá un pago igual a cero.*
- *Simetría: Para todo $(N, v) \in G^N$ y para todo par de jugadores $i, j \in N$ simétricos, $\phi_i(N, v) = \phi_j(N, v)$. La propiedad de asimetría nos dice que si un par de jugadores realizan las mismas contribuciones a las coaliciones recibirán el mismo pago.*
- *Aditividad: Para todos (N, v), $(N, w) \in G^N$, $\phi(N, v + w) = \phi(N, v) + \phi(N, w)$. Esta propiedad establece que si un juego se descompone en suma de dos, el pago que recibirá cada jugador será la suma de los pagos de cada uno de los juegos.*

Entonces ϕ es el valor de Shapley.

Teorema 2.3 *Sea $(N, v) \in G^N$ convexo, entonces $Sh(N, v) \in C(N, v)$.*

Deducimos con este último resultado que para todo juego superaditivo el valor de Shapley estará contenido en el núcleo del juego.

Se pueden definir muchas más soluciones para los juegos cooperativos, como el τ-valor, el nucleolo, etc. Y otras soluciones tipo conjunto, como el conjunto estable o el conjunto de Weber.

2.1.2. Juegos cooperativos asociados a problemas PERT/CPM

Cuando finaliza un proyecto es normal encontrarnos con que hay retrasos sobre la fecha de finalización. Como hemos visto, en estos casos suelen establecerse penalizaciones a las empresas implicadas en la realización del proyecto. En el caso de la Administración Pública en España, estas penalizaciones están establecidas por ley. Llegados a esta situación, podemos plantear juegos cooperativos para evaluar la responsabilidad de cada empresa en el retraso del proyecto y resolver el problema de repartir el coste producido por dicho retraso entre las empresas involucradas.

Partimos de las suposiciones de que, por contrato, ninguna empresa que no tenga retraso en la realización de su actividad podrá ser penalizada y, además, ninguna actividad podrá iniciarse antes de que finalicen todas las actividades precedentes.

Dado que vamos a tratar con retrasos, primero debemos introducir la notación que emplearemos en estos problemas. Denotaremos por d_i el retraso sufrido en la actividad i (o producido por la empresa i) y $T(d)$ el tiempo real de realización del proyecto, es decir, el tiempo estimado previamente para el proyecto más el retraso producido en este por las actividades retrasadas.

A continuación haremos una breve introducción bibliográfica presentando cómo se ha tratado este tema dentro de la literatura de juegos cooperativos, centrándonos principalmente en tres artículos.

Branzei et al. (2002)

Branzei et al. (2002) toma como referencia, en primer lugar, los problemas de bancarrota, un caso particular de juegos cooperativos que plantean un escenario donde un conjunto de acreedores deben repartirse un bien que por lo general no es suficiente para cubrir las deudas reclamadas.

Estos autores usan como información para efectuar el reparto del coste del retraso los retrasos individuales de cada actividad $(d_i)_{i\in N}$ y el retraso global del proyecto, $T(d) - T$. Dado que la suma de los retrasos individuales siempre será mayor o igual que el retraso del proyecto, esto recuerda mucho los modelos de bancarrota, donde las peticiones de los acreedores siempre son mayores o iguales al bien a repartir. Los autores identifican entonces las demandas de los acreedores con los retrasos individuales y el bien que se ha de repartir con el retraso global del proyecto. Una vez definido este problema de bancarrota aplican las reglas más conocidas como son: la proporcional, la regla de igual ganancia (CEA) y la de igual pérdida (CEL) entre otras.

En la regla proporcional dividiríamos el retraso global del proyecto (y, en consecuencia, las penalizaciones correspondientes) de forma proporcional al retraso individual de cada actividad. En la regla de igual ganancia, o CEA, repartimos $T(d) - T$ igualando el tiempo atribuido a cada actividad con la única restricción de que a ninguna actividad i le corresponderá un retraso mayor que d_i. Por el contrario, en la regla de igual pérdida, dividimos el retraso del proyecto entre las actividades adjudicando a todas un déficit idéntico sobre d_i, con la restricción de que el retraso no puede ser menor que cero.

Por otro lado, en Branzei et al. (2002) se define el siguiente juego de costes con función característica

$$c(S) = \text{mín}\left\{\sum_{i\in S} d_i, T(d) - T\right\},$$

para cada $S \subset N$. Donde N es el conjunto de actividades que han sufrido retraso, d_i es el retraso de la actividad i y $T(d) - T$ es el tiempo real del proyecto menos el tiempo estimado, es decir, el retraso del proyecto.

Prueban que las reglas derivadas de los problemas de bancarrota se encuentran en el núcleo del juego, por lo que son estables. También plantean el uso del juego cooperativo para efectuar repartos como el valor de Shapley o el τ-valor.

Por último, se plantea otra solución en dos etapas. En la primera se divide el coste del retraso entre los caminos cuyo retraso es mayor que 0 proporcionalmente a dicho retraso. En la segunda etapa se divide, a su vez, el coste calculado previamente para cada camino entre las actividades involucradas en el mismo que se han retrasado, también de forma proporcional a sus retrasos. Esta solución también se puede ver como un juego cooperativo si tomamos los caminos con retraso como jugadores.

Notemos que en esta última solución se tiene en cuenta que algunas actividades pueden tener una influencia mayor en el retraso global del proyecto, al provocar el retraso de más de un camino. Por ejemplo, supongamos un proyecto en el que solo dos actividades han sufrido retraso y que, además, este retraso es el mismo. Si una de las actividades pertenece a un solo camino y la otra pertenece a dos diferentes, el coste de la actividad que provoca un retraso en dos caminos será el doble que el de la que solo provoca que se retrase un camino.

A pesar de esta última aproximación, las dos primeras aproximaciones no distinguen las diferentes contribuciones que puede tener cada actividad en el retraso final, y mucho menos distinguen entre actividades críticas y no críticas. Como hemos visto al plantear el modelo PERT/CPM, no todos los retrasos tienen el mismo impacto en el retraso final del proyecto, ya que una actividad no crítica tendrá cierta holgura que le permitirá un retraso que no afecte a la duración total del proyecto, mientras que una actividad crítica que se retrase provocará inexorablemente un retraso. Por ello, es necesario establecer con más claridad la responsabilidad de cada actividad sobre el retraso global del proyecto.

Bergantiños y Sánchez (2002)

Bergantiños y Sánchez (2002) definen un problema PERT con retrasos como una 4-tupla (G, d, e, c) donde G es el grafo del proyecto, $d = (d_i)_{i \in N}$ es el vector de duraciones de las actividades, $e = (e_i)_{i \in N}$ es el vector de tiempos de finalización de cada actividad y c es la función que asocia un coste al retraso del proyecto. Inicialmente clasifican los retrasos en diferentes clases, para ello se fijan en las actividades finales del proyecto, aquellas cuyo nodo final coincide con el nodo de fin del proyecto y definen para cada una de ellas una clase como el conjunto de las actividades que la preceden en el grafo.

A la hora de proceder con el reparto del coste del retraso, Bergantiños y Sánchez proponen utilizar la regla de reparto de costes en serie introducida por Sprumont (1998). En esta regla el reparto se va realizando de manera secuencial considerando las clases ordenadas en función de los retrasos asociados a cada una de ellas. Primero se reparte el coste asociado al retraso más pequeño entre todas las clases a partes iguales, ya que todas han incurrido al menos en dicho retraso. Después se considera coste asociado al siguiente retraso menos el coste ya asignado en la fase anterior y se reparte a partes iguales entre las clases que han generado un retraso mayor o igual a éste, y así sucesivamente. Una vez se ha repartido el coste entre las clases se procederá a repartirlo dentro de cada una usando exactamente el mismo procedimiento. La asignación final para cada una de las actividades se corresponde con la suma de los costes asignados en cada una de las clases a las que pertenece.

Estos autores también proponen otro reparto basado en la teoría de juegos cooperativos. Para ello proponen de nuevo un reparto en dos etapas. En la primera de ellas definen un juego entre las actividades finales del proyecto donde el coste asociado a cada coalición se corresponde con el máximo retraso que producen, y se realiza un reparto aplicando el valor de Shapley. Posteriormente se reparte dentro de cada clase, formada por las predecesoras de cada una de las actividades finales, aplicando de nuevo el valor de Shapley a un nuevo juego de costes.

Notemos que en estos dos métodos algunas actividades pueden ser responsables del retraso de varias actividades finales, por lo que su responsabilidad en el retraso final será mayor, y en consecuencia el coste que deberán asumir. Esto supone una mejora con las primeras soluciones basadas en problemas de bancarrota (Branzei et al., 2002), al entender que no todas las actividades son igualmente responsables del retraso final.

Estévez-Fernández et al. (2002)

Estévez-Fernández et al. (2007) definen un problema PERT con retrasos como una 4-tupla (G, t, d, c), donde G es el grafo del proyecto, $t = (t_i)_{i \in N}$ son los tiempos estimados para las actividades, $d = (d_i)_{i \in N}$ los retrasos de cada actividad y c la función de coste. Plantean un juego cooperativo asociado al problema de retrasos en proyectos más complejo, donde se tienen en cuenta los retrasos de cada actividad de la coalición y los caminos del inicio al final del proyecto a los que pertenecen.

El conjunto de jugadores queda definido como el conjunto de actividades y el coste de la coalición se calcula como la máxima contribución de la coalición al retraso del proyecto causado por los caminos donde los miembros de la coalición están involucrados. Formalmente, sea (N, c) el juego cooperativo,

$$c(S) = \max_{\pi \in \Pi(S)} \left\{ \min \left\{ \sum_{i \in \pi \cap S} d_i, \left(\sum_{i \in \pi} d_i - ps_\pi \right)^+ \right\} \right\}$$

para cada $S \subset N$, donde $\Pi(S)$ representa el conjunto de caminos en los que hay al menos una actividad perteneciente al conjunto S y ps_π es, como vimos en el capítulo anterior, la holgura del camino π.

Además, en dicho trabajo, queda demostrado que el núcleo del juego es no vacío. De esta forma podemos garantizar siempre la existencia de al menos un reparto estable, es decir, un reparto que garantice que ninguna coalición S tenga incentivos para desviarse.

Estévez-Fernández et al. (2007) también estudian los problemas en los que existe un adelanto sobre la fecha de finalización del proyecto. A veces, en estos casos, se pactan cláusulas en el contrato para una compensación económica si el proyecto finaliza antes de lo establecido. De forma similar a los problemas con retraso, podemos asociar un juego cooperativo, que omitiremos aquí, al problema con el objetivo de obtener un reparto de los beneficios.

El planteamiento de este juego resulta interesante, en relación a los repartos vistos anteriormente, pues tiene en cuenta las holguras de los caminos retrasados, lo que lleva a ponderar mejor el retraso de cada actividad.

Ejemplo 2.1 *Sea el problema ilustrado por la red dada en la Figura 2.1.*

Tenemos un grafo con 6 actividades, dos de ellas ficticias, que no serán relevantes en el problema.

Los diferentes caminos del grafo son $\pi_1 = A - E - B$, $\pi_2 = A - C$ *y* $\pi_3 = F - D$. *El tiempo del proyecto se corresponde con el camino más largo, que en este caso es el camino* π_3, *con un tiempo de 10. Por lo que tenemos un tiempo estimado del proyecto* $T = 10$. *Las holguras del resto de los caminos son 1 para el camino* π_1 *y 2 para el camino* π_2.

El vector de retrasos de las actividades no ficticias es (1,3,2,0). Por lo que la duración real del proyecto, $T(d)$, *es 13, que viene dada por el tiempo real del camino* π_1. *Con estos datos calculamos el retraso del proyecto,* $T(d) - T = 3$. *Tendremos, que para todos los juegos presentados, el conjunto de jugadores será* $N = \{A, B, C\}$.

A continuación obtenemos los diferentes repartos, presentados en esta sección, del retraso sufrido por el proyecto entre las actividades que lo han generado.

Repartos presentados por Branzei et al. (2002)

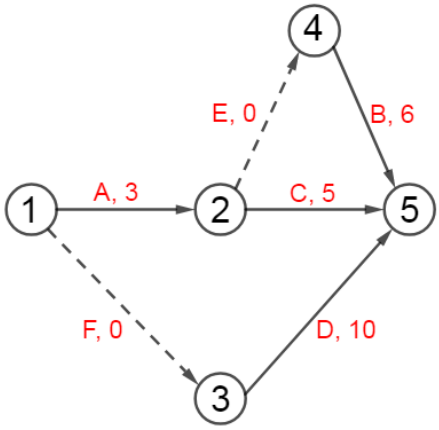

Figura 2.1: Red del problema.

En el Cuadro 2.1 vemos los diferentes repartos que derivan de los problemas de bancarrota.

Regla de reparto	A	B	C
Proporcional	0.5	1.5	1
CEA	1	1	1
CEL	0	2	1

Cuadro 2.1: Reglas de reparto basadas en problemas de bancarrota.

Estos repartos no diferencian entre actividades y adjudican indistintamente un coste sin tener en cuenta la responsabilidad en el retraso final. Por ejemplo, en la regla CEL, no resulta razonable que la empresa A no sea responsable del coste del retraso, a pesar de formar parte del camino con un mayor tiempo real.

El juego propuesto por Branzei et al. (2002) tiene como función característica

$$c_B = [1, 3, 2, 3, 3, 3, 3].$$

El valor de Shapley del juego (N, c_B) es (0.5, 1.5, 1). En este caso, el valor de Shapley del juego coincide con la regla proporcional. Este reparto adjudica un mayor coste a la actividad C que a la A, cuando el retraso del proyecto parece estar causado en mayor medida por el retraso de la actividad A, pues pertenece a los dos caminos que causan el retraso del proyecto y la actividad C solo pertenece al camino π_2, que es causante de un menor retraso que el camino π_1.

El núcleo del juego (N, c_B) es

$$C(N, c_B) = \{(x, y, z) \in \mathbb{R}^3 : x + y + z = 3, 0 \leq x \leq 1, 0 \leq y \leq 3, 0 \leq z \leq 2\}.$$

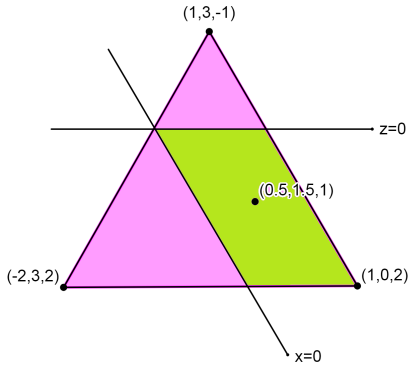

Figura 2.2: Representación de la imputaciones del juego.

El valor de Shapley del juego pertenece al núcleo, como vemos representado en la Figura 2.2, donde podemos ver el conjunto de las imputaciones conteniendo al núcleo del juego y al valor de Shapley. A pesar de ello, algunos repartos aunque pertenecientes al núcleo, no parecen una buena elección al dividir el retraso. Por ejemplo, el reparto (1,0,2) pertenece al núcleo del juego, pero no adjudica ningún coste a la actividad B, que es la actividad con un mayor retraso del proyecto.

También podemos observar que, como se demuestra en Branzei et al. (2002), las reglas de reparto proporcional, de igual ganancia y de igual pérdida pertenecen al núcleo del juego. Además, el reparto proporcional coincide con el valor de Shapley.

En la tercera y última propuesta de reparto presentada en Branzei et al.(2002), consistente en dos etapas, se divide proporcionalmente el retraso, primero entre los caminos que no terminan en el tiempo establecido y, posteriormente, entre las actividades que generan, dentro de esos caminos, el retraso. Así, en la primera etapa asignamos un retraso al camino π_1 de $\frac{12}{7}$ y al camino π_2 de $\frac{9}{7}$. Por último, repartimos el retraso otorgándole a la actividad A un cuarto de lo correspondiente al camino π_1 (el resto a la actividad B) y un tercio de lo correspondiente al camino π_2 (el resto a la actividad C). Obtenemos, entonces, un vector de retrasos $(\frac{6}{7}, \frac{9}{7}, \frac{6}{7}, 0)$.

Como podemos ver, este reparto resulta una mejora con respecto, por ejemplo, al valor de Shapley del juego presentado anteriormente, ya que pondera el hecho de que la actividad A contribuya al retraso de dos caminos con retraso frente a la actividad C que solo retrasa uno y les adjudica el mismo coste del retraso a pesar de que la actividad C tenga un retraso dos veces mayor. Aún así, no se tiene en cuenta que las actividades A y B retrasan un camino crítico.

Repartos presentados por Bergantiños y Sánchez (2002)

Bergantiños y Sánchez (2002) plantean dos soluciones, ambas con dos etapas similares. En la primera dividimos el retraso entre las actividades finales. Repartimos, siguiendo el método que corresponda, el retraso entre las actividades finales B, C y D, que finalizan con un retraso acumulado de 3, 1 y 0 respectivamente. En la segunda etapa, repartimos el coste adjudicado a la actividad B entre las actividades predecesoras que tienen retraso, en este caso las actividades B y A, el coste adjudicado a C entre la propia C y A y el coste adjudicado a D a ella misma. Podemos ver, antes de realizar ningún cálculo, que la actividad D no tendrá adjudicado ningún coste, por lo que el problema se puede restringir a las actividades retrasadas.

Método	Primera etapa	Segunda etapa
Regla en serie de dos etapas	$Pre(B)$: 2.5 $Pre(C)$: 0.5	Actividad A: 0.25 Actividad B: 2.25
		Actividad A: 0 Actividad C: 0.5
Solución basada en el valor de Shapley en dos etapas	$Pre(B)$: 2.5 $Pre(C)$: 0.5	Actividad A: 0.5 Actividad B: 2
		Actividad A: 0.25 Actividad C: 0.25

Cuadro 2.2: Reparto para los métodos de Bergantiños y Sánchez (2002).

A partir del Cuadro 2.2, deducimos que los vectores de repartos, para las 4 actividades no ficticias, son (0.25, 2.25, 0.5, 0), para la regla en serie en dos etapas, y (0.75, 2, 0.25, 0) para la solución basada en el valor de Shapley en dos etapas.

En ambos métodos presentados por Bergantiños y Sánchez (2002) se valora de forma diferente la contribución al retraso final de las diferentes actividades aunque el retraso concreto de estas sea el mismo. Así, las actividades que contribuyan a un retraso de varias actividades finales, como la actividad A en nuestro ejemplo, verán aumentado su coste en mayor medida que el resto. Esto se refleja, por ejemplo, en el reparto basado en el valor de Shapley en dos etapas, en el que el coste de la actividad A es mayor que el de la C a pesar de que su retraso es menor.

Repartos presentados por Estévez-Fernández et al. (2007)

La función característica del juego obtenido siguiendo el planteamiento de Estévez-Fernández et al. (2007) es

$$c_{EF} = [1, 3, 1, 3, 1, 3, 3].$$

El valor de Shapley del juego (N, c_{EF}) es (0.33, 2.33, 0.33). El valor de Shapley adjudica un coste igual a la empresa A que a la C, a pesar de tener menos retraso. Esto se debe a que el juego trata por igual a ambas empresas, al estar la actividad C en un camino que a lo sumo retrasaría el proyecto 1. La empresa con mayor coste es la B, pues es la mayor causante del retraso, al ser la responsable de la mayor parte del retraso del camino π_1.

Ya sabemos, por los resultados ofrecidos en Estévez-Fernández et al. (2007), que el núcleo del juego (N, c_{EF}) es no vacío. Concretamente

$$C(N, c_E F) = \{(x, y, z) \in \mathbb{R}^3 : x + y + z = 3, 0 \leq x \leq 1, 2 \leq y \leq 3, 0 \leq z \leq 1\}.$$

Como vemos en la Figura 2.3, el valor de Shapley pertenece al núcleo del juego, por lo que ninguna coalición de empresas tendrá incentivos para desviarse y obtener un reparto mejor.

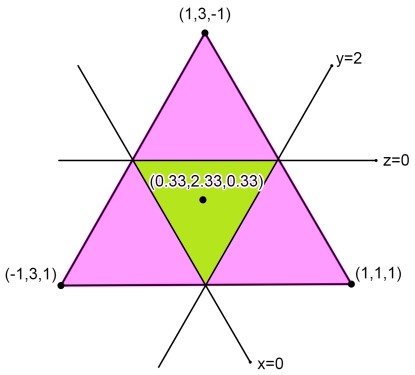

Figura 2.3: Núcleo y valor de Shapley del juego (N, c_{EF}).

El planteamiento de este juego resulta interesante, en relación a los repartos vistos anteriormente, pues tiene en cuenta las holguras de los caminos retrasados, lo que lleva a ponderar mejor el retraso de cada actividad. Así, la actividad C tiene un pago igual al de la actividad A a pesar de tener un retraso de 1 unidad de tiempo más, pues pertenece a un solo camino que no es crítico y tiene una holgura de 1. Además, la actividad B es la actividad a la que le corresponde una mayor parte del retraso, pues pertenece al camino crítico del proyecto y su retraso es el mayor de las actividades.

2.2. Planteamiento no cooperativo

2.2.1. Introducción a los juegos no cooperativos

Un juego no cooperativo se definirá, en oposición a un juego cooperativo, como aquel en el que los agentes no disponen de mecanismos que les permitan establecer acuerdos vinculantes. Dicho de otro modo, los jugadores no podrán colaborar y formar coaliciones para obtener mayor beneficio. En estos juegos, el objetivo de cada jugador será escoger una estrategia de actuación que maximice su pago teniendo presente las elecciones que harán el resto de jugadores, que podrán perjudicarle o beneficiarle. Podemos definir dos tipos de juegos no cooperativos:

- Los juegos en forma estratégica o normal, en los que los jugadores eligen simultáneamente su estrategia.
- Los juegos en forma extensiva, en los que los jugadores eligen su estrategia secuencialmente.

En los problemas de retraso en proyectos que estudiaremos en este capítulo tendremos situaciones en las que las empresas tomarán las decisiones sobre si retrasar o no sus correspondientes actividades al comenzar su realización, por lo que podrán tomar sus decisiones teniendo en cuenta la información sobre las actividades ya finalizadas. Debido a esto, para modelar los problemas de retraso en proyectos utilizaremos juegos en forma extensiva dada la estructura del problema y las relaciones de precedencia que hay entre las diferentes actividades, lo que supone una situación de decisión secuencial y multi-etápica.

Previamente a la presentación de los juegos en forma extensiva, vamos a introducir los juegos en forma estratégica, pues serán necesarios para definir el concepto de solución más importante en juegos

no cooperativos, el equilibrio de Nash, o NE.

Juegos en forma estratégica

Definición 2.15 *Un juego en forma estratégica G es una terna $(N, \{X_i\}_{i\in N}, \{U_i\}_{i\in N})$ donde:*

- $N = \{1, 2, ..., n\}$ *es el conjunto de jugadores.*
- $\{X_i\}_{i\in N}$ *son los conjuntos de estrategias de los jugadores.*
- $U_i : X = \prod_{i=1}^{n} \longrightarrow \mathbb{R}$ *son las funciones de pago de cada jugador $i \in N$.*

Denominaremos a cada $x \in X$ un perfil de estrategias, por lo que cada función de pago asignará el pago correspondiente a dicho perfil (que definen todos los jugadores) al jugador i.

El objetivo de cada jugador $i \in N$ será maximizar su función de pago U_i teniendo en cuenta las elecciones de sus rivales. Surge así, el concepto de solución de un juego en forma estratégica. En concreto, vamos a presentar la más importante de estas soluciones, el NE.

El NE de un juego Γ es un perfil de estrategias tal que ningún jugador obtendrá una mayor utilidad al desviarse de su estrategia, asumiendo que el resto de jugadores mantienen sus estrategias.

Definición 2.16 *Sea G un juego en forma estratégica, un equilibrio de Nash del juego es un perfil de estrategias $x \in X$ que verifica que*

$$U_i(x) \geq U_i(x_{-i}, x_i'),$$

para todo $x_i' \in X_i$ y todo $i \in N$. Donde

$$(x_{-i}, x_i') = (x_1, ..., x_{i-1}, x_i', x_{i+1}, ..., x_n).$$

Juegos en forma extensiva

Como hemos explicado anteriormente, nos centraremos en los juegos no cooperativos en forma extensiva, que nos permitirán modelar situaciones no estáticas. Nos centraremos únicamente en los juegos en forma extensiva finitos, ya que no será necesario el estudio de juegos infinitos para modelar problemas de retraso en proyectos. Formalmente, definimos un juego en forma extensiva como vemos a continuación.

Definición 2.17 *Un juego en forma extensiva Γ una 6-tupla $(N, (A, M), P, Q, K, u)$ donde:*

- $N = \{1, 2, ..., n\}$ *es el conjunto de jugadores.*
- (A, M) *es un árbol de juego, que consideraremos finito, donde A es un conjunto finito de nodos y M es un conjunto finito de arcos orientados entre los nodos de A que definirán el orden en que se juega. La estructura del árbol no podrá tener bucles.*
  Cada punto donde empieza un arco es un nodo de decisión, que podrá ser inicial, seguidor o terminal. Denotaremos por Z al conjunto de nodos terminales (que no son nodos de decisión), que son aquellos de los que no salen arcos. Tendremos, además, un único nodo inicial $a' \in A$ al que no llegará ningún arco.
- $P = \{P_1, ..., P_n\}$ *es una partición de $A \backslash Z$ que proporciona una indicación de los nodos en los que toma una decisión cada jugador. P_i será el conjunto de nodos de decisión de i, con $i \in N$.*

- $Q = \{Q_1, ..., Q_n\}$ *es una subpartición de* P*. Para cada* $i \in N$*,* Q_i *es una partición de* P_i *que describe la información de la que dispone el jugador* i *cuando tiene que decidir. Denominaremos a estos conjuntos como conjuntos de información.*
- $K = \{K_1, ..., K_n\}$ *es el conjunto de todas las posibles elecciones de los jugadores en sus conjuntos de información.* K_i*, para cada* $i \in N$*, es el conjunto de todas las acciones disponibles para el jugador* i*, i.e.,* K_i *es un conjunto formado por los arcos que salen de los nodos de decisión de* i*.*
- $u = (u_1, ..., u_n)$*, donde* $u_i : Z \longrightarrow \mathbb{R}$*, para cada* $i \in N$*, denotará la función de pago del jugador* i*.*

En este tipo de juegos, en los que las decisiones de los jugadores se van tomando a medida que avanza el juego y no de forma simultánea, resultará muy importante la información que maneje cada jugador en el momento que toma cada decisión (el conjunto de información). Definiremos los juegos de información perfecta como todo juego Γ en el que cada jugador i, en cada uno de sus nodos de decisión, está perfectamente informado de todo lo que ha pasado con anterioridad en el juego. En oposición, diremos que un juego Γ es de información imperfecta si no es de información perfecta.

Basándonos únicamente en la información que cada jugador maneja sobre sus propias decisiones, podemos definir un juego en forma extensiva Γ como un juego de memoria perfecta si cada jugador i recuerda cuáles han sido todas sus decisiones pasadas y toda la información que tenía en el momento de tomarlas. Como anteriormente, a un juego que no sea de memoria perfecta lo denominaremos de memoria imperfecta. De estas definiciones, resulta inmediato deducir que todo juego de información perfecta es también un juego de memoria perfecta.

Definición 2.18 *Sea* Γ *un juego en forma extensiva y sea* $i \in N$*. Denominaremos estrategia pura del jugador* i *a una aplicación que asigna, a cada* $q \in Q_i$*, una elección* $x_i(q) \in S_i$*. Un perfil de estrategias puras del juego* Γ *es un vector* $x = (x_1, ..., x_n)$ *tal que* x_i *es una estrategia pura del jugador* i *para todo* $i \in N$*. Denotaremos por* X_i *el conjunto de estrategias puras del jugador* i *en* Γ *y por* X *el conjunto de perfiles de estrategias puras (*$X = \prod_{i \in N} X_i$*).*

Definición 2.19 *Sea* Γ *un juego en forma extensiva y sea* $i \in N$*, diremos que una estrategia de comportamiento del jugador* i*,* b_i*, es una aplicación que atribuye a cada estrategia pura* $x_i \in X_i$ *una probabilidad. Denotaremos por* B_i *al conjunto de todas las estrategias de comportamiento del jugador* i*.*

Análogamente a las estrategias puras, definimos un perfil de estrategias de comportamiento del juego Γ como un vector $b = (b_1, ..., b_n) \in B = \prod_{i \in N} B_i$ tal que b_i es una estrategia de comportamiento del jugador i, para todo $i \in N$.

Notemos que toda estrategia pura de cada jugador se puede escribir como una estrategia de comportamiento si asignamos toda la probabilidad a dicha estrategia pura. Deducimos, entonces, que $X_i \subset B_i$ para todo $i \in N$ y, en consecuencia, $X \subset B$.

Tomando un juego en forma extensiva Γ vemos que para cada estrategia de comportamiento $b \in B$ y cada nodo del árbol $a \in A$ podemos calcular la probabilidad de que el nodo a sea alcanzado si los jugadores juegan b. Denotaremos dicha probabilidad por $\rho(a, b)$. Si nos restringimos únicamente a los nodos terminales del árbol obtenemos, para cada $b \in B$, una distribución de probabilidad sobre Z. En consecuencia, tendremos una función de pago para cada jugador $i \in N$ y cada $b \in B$ $U_i : B \longrightarrow \mathbb{R}$ definida por

$$U_i(b) = \sum_{z \in Z} \rho(z, b) u_i(z).$$

Concluimos que para todo juego en forma extensiva Γ podemos asociar un juego en forma estratégica G cuya función de pago será U_i para todo $i \in N$.

El equilibrio de Nash

El concepto de solución más importante para juegos en forma extensiva, al igual que para los juegos en forma estratégica, es el NE.

Definición 2.20 *Sea Γ un juego no cooperativo en forma extensiva, un NE del juego es un perfil de estrategias de comportamiento $b \in B$ tal que*

$$U_i(b) \leq U_i(b_{-i}, b_i').$$

Notemos que no todo juego tiene un NE y, de tenerlo, no necesariamente será único. Por ello, resultará importante ofrecer resultados que nos garanticen la existencia de, al menos, un NE.

Proposición 2.5 *Sea Γ un juego en forma extensiva y sea $x \in X$ un perfil de estrategias puras de Γ. Entonces, x es un NE de Γ si y solo si*

$$U_i(s) \leq U_i(x_{-i}, x_i'),$$

para todo $x_i' \in X_i$ y todo $i \in N$.

Corolario 2.1 *Sea Γ un juego en forma extensiva y sea $x \in X$ un perfil de estrategias puras de Γ, entonces x es un NE de Γ si y solo si x es un NE de $G(\Gamma)$, siendo $G(\Gamma)$ el juego en forma estratégica asociado al juego Γ.*

Teorema 2.4 *Sea Γ un juego en forma extensiva de memoria perfecta, entonces tiene al menos un NE.*

El equilibrio perfecto en subjuegos

A pesar de que el NE resulta una buena solución para encontrar perfiles de estrategias que resulten una buena elección para los diferentes jugadores, en algunas situaciones puede ser necesario refinar este concepto. El más importante de estos refinamientos, que introduciremos a continuación, es el equilibrio perfecto en subjuegos.

Definición 2.21 *Sea Γ un juego en forma extensiva, diremos que Γ puede ser descompuesto en el nodo $a \in A \setminus Z$ si no hay conjuntos de información conteniendo simultáneamente nodos sucesores de a y del resto de nodos.*

Definición 2.22 *Sea Γ un juego en forma extensiva y $a \in A \setminus Z$, tal que Γ puede ser descompuesto en a. Diremos que el juego Γ_a, inducido por Γ, cuyo árbol tiene como nodo inicial a, es un subjuego de Γ. Dado un perfil de estrategias de comportamiento b, la restricción de este al subjuego Γ_a se denotará por b_a.*

Definición 2.23 *Diremos que un perfil de estrategias $b \in B$ es un equilibrio perfecto en subjuegos de Γ si, para todo $a \in A \setminus Z$, b_a induce un NE para el subjuego Γ_a de Γ.*

Resulta inmediato observar que todo equilibrio perfecto en subjuegos de un juego en forma extensiva Γ también será un NE para dicho juego. Pues si tomamos el nodo a como el nodo inicial del juego Γ, el subjuego generado será $\Gamma_a = \Gamma$ y la restricción del perfil de estrategias de comportamiento $b \in B$ para Γ_a será $b_a = b$, por lo que de ser b un equilibrio perfecto en subjuegos será también un NE.

Presentamos a continuación un resultado que nos permitirá tener un procedimiento para identificar los equilibrios perfectos en subjuegos.

Proposición 2.6 *Sea Γ un juego en forma extensiva. Supongamos que el juego se puede descomponer en los nodos $a_1, ..., a_m$ tales que a_j no va después de a_k para todo $j, k \in \{1, ..., m\}$. Consideremos un perfil de estrategias de comportamiento $b \in B$ y denotemos por $\Gamma^{b_{a_1,...,a_m}}_{-(a_1,...,a_m)}$ el juego en forma extensiva resultante de Γ después de borrar los subjuegos $\Gamma_{a_1}, ..., \Gamma_{a_m}$ y de definir, para cada $i \in N$ y cada $j \in \{1, ..., m\}$,*

$$u_i^{\Gamma^{b_{a_1,...,a_m}}_{-(a_1,...,a_m)}}(a_j) = U_i^{\Gamma_{a_j}}(b_{a_j}).$$

Si b_{a_j} es un NE de Γ_{a_j} para todo $j \in \{1, ..., m\}$ y $b_{-(a_1,...,a_m)}$ es un NE de $\Gamma^{b_{a_1,...,a_m}}_{-(a_1,...,a_m)}$, entonces b es un NE de Γ.

Podemos definir, entonces, el procedimiento para obtener los equilibrios perfectos en subjuegos para un juego en forma extensiva Γ, denominado inducción hacia atrás. En primer lugar, descomponemos el juego en otros menos complejos y, empezando por los más simples (aquellos que se encuentran al final de Γ), obtenemos recursivamente los NE de los subjuegos sustituyendo los subjuegos de los que ya hayamos calculado su NE por los vectores de pago de dicho NE. A continuación ilustramos este procedimiento mediante un sencillo ejemplo.

Ejemplo 2.2 *Tenemos un pareja de jugadores, que llamaremos el jugador 1 y el jugador 2, que quieren organizar un plan para el viernes a la noche. El jugador 1 prefiere ir al cine (c) mientras que al jugador 2 le gustaría más ir a bailar (b). A pesar de esta diferencia, ambos prefieren pasar la noche juntos. La estructura del juego, como se ilustra en la Figura 2.4, consiste en que el jugador 1 decide primero el plan y luego elige el jugador 2 teniendo conocimiento de la elección previa del jugador 1.*

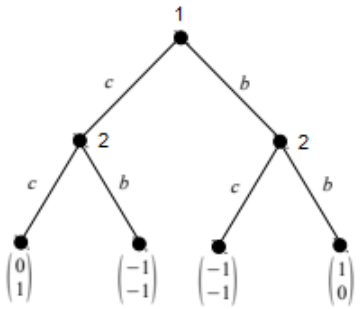

Figura 2.4: Representación del árbol del juego.

Tomamos los dos subjuegos, representados en la Figura 2.5, que tienen como nodo inicio el nodo en el que el jugador 2 debe tomar la decisión dependiendo de la decisión previamente tomada por el jugador 1. Es fácil ver que si nos encontramos en el caso de que el jugador 1 eligió cine, el jugador 2 elegirá también cine, pues era su elección predilecta ($1 > -1$), por lo que el NE del subjuego será c. Pero si nos encontramos en el subjuego generado cuando el jugador 1 elige baile entonces el jugador 2 escogerá baile ($0 > -1$), por lo que el NE del subjuego será b.

Sustituyendo en el juego original los subjuegos por las decisiones del jugador 2 obtenemos el juego representado en la Figura 2.6. Es fácil ver, entonces, que el jugador 1 elegirá baile, por lo que el NE de este juego sería b.

Tenemos entonces que (b,b) es un NE del juego y, por construcción, es también un equilibrio perfecto en subjuegos.

Figura 2.5: Subjuegos generados tras la elección del jugador 1 de cine (izquierda) o baile (derecha).

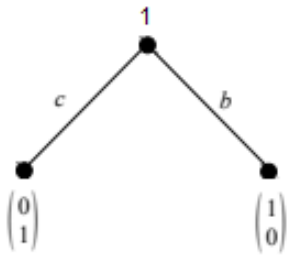

Figura 2.6: Juego inducido por elecciones del jugador 2.

Presentamos a continuación dos teoremas que ofrecen condiciones suficientes para la existencia de equilibrios perfectos en subjuegos.

Teorema 2.5 *Sea* Γ *un juego en forma extensiva de memoria perfecta, entonces tiene al menos un equilibrio perfecto en subjuegos.*

Teorema 2.6 *Sea* Γ *un juego en forma extensiva de información perfecta, entonces tiene al menos un perfil de estrategias puras que es un equilibrio perfecto en subjuegos.*

Una última definición previa, relacionada con los juegos, que será necesaria en la siguiente sección es la Pareto dominancia. Nos permitirá distinguir si unas soluciones son preferibles a otras. Diremos que $x = (x_i)_{i\in N}$ domina, en el sentido de Pareto, a $y = (y_i)_{i\in N}$, con $x, y \in S \subset \mathbb{R}^N$, si $x_i \geq y_i$ para todo $i \in N$ y existe $j \in N$ tal que tal que $x_j > y_j$. Definiremos la frontera de Pareto de un subconjunto $S \subset \mathbb{R}^N$ como

$$PB(S) = \{x \in S : \text{ no existe } y \in S \text{ tal que y domine a } x\}.$$

2.2.2. Juegos no cooperativos asociados a problemas PERT/CPM

Cuando un proyecto es llevado a cabo por un conjunto de empresas, no solo es pertinente el estudio del reparto de las penalizaciones, como ya hemos tratado utilizando juegos cooperativos, sino que será útil, para el planificador y las empresas, predecir el comportamiento del resto de agentes implicados en el proyecto y establecer estrategias de actuación en consecuencia.

Las empresas tendrán unos recursos limitados que tendrán que gestionar entre la actividad que les ha sido adjudicada en el proyecto y otras actividades que se supone estarán haciendo simultáneamente en otros proyectos. Una empresa constructora, por ejemplo, que esté trabajando en la edificación de

dos edificios en el mismo período de tiempo, puede no tener los recursos necesarios para realizar las dos actividades a tiempo y consecuentemente incurrirá en un retraso. En este caso, el problema de la empresa será gestionar que cantidad de recursos destinar a cada proyecto teniendo en cuenta el beneficio que obtendrá por cada uno y el coste que le supondría el retraso de cada proyecto con las respectivas penalizaciones, considerando los retrasos en los que incurrirán o ya hayan incurrido el resto de empresas implicadas en los proyectos.

Por esto, será indiferente hablar de los recursos que dedicará una empresa a su actividad o del retraso en el que incurra, ya que el retraso vendrá propiciado por los recursos que deje de destinar para la correcta realización de la actividad.

A pesar de que las empresas tendrán que decidir el retraso que generarán en su correspondiente actividad en el proyecto, es lógico pensar que este retraso deba estar acotado. Es por ello que supondremos un máximo de retraso generado por cada actividad que implicará un máximo de recursos asignados a otros proyectos.

También será importante el papel del planificador y su decisión de qué penalización fijar. Por ejemplo, si el planificador es la administración pública que desea construir la sede para una Exposición Internacional, el objetivo al establecer la penalización será desincentivar cualquier retraso en las actividades que componen el proyecto. Pues la fecha del evento está fijada con anterioridad y el coste que supondría no tener una sede supera el precio de esta. Por otra parte, puede darse el caso de un proyecto en el que no exista esa necesidad de finalizar en una determinada fecha, por lo que el planificador puede buscar establecer una penalización que provoque que las empresas se retrasen y así obtener una ganancia extra.

Bergantiños y Lorenzo (2017) estudian el problema de retrasos en proyectos, modelando un juego no cooperativo en forma extensiva, para obtener las mejores estrategias del planificador (qué penalización fijar) y de las empresas (cuántos recursos dedicar a la actividad).

Utilizaremos un juego no cooperativo en forma extensiva con 3 etapas para modelar nuestro problema. En la primera etapa el planificador impondrá la penalización. En la segunda las empresas decidirán cuántos recursos dedicar a la actividad (y en consecuencia, el retraso generado). En esta segunda etapa las empresas tomarán las decisiones de forma dinámica, pues no decidirán qué retraso generarán hasta que tengan que iniciar su actividad y sus predecesoras hayan finalizado, por lo que tendrán la información de cuánto se han retrasado para tomar su decisión. Es por esta situación que es necesario el uso de los juegos en forma extensiva para modelar el problema. Por último, se realizarán los pagos y se impondrán las penalizaciones correspondientes.

Un problema de retrasos en proyectos es una tripla (G, c, r) donde G es el grafo, c es el coste por unidad de tiempo en el que incurre el planificador cuando hay retrasos y $r = (r_i)_{i \in N}$ es el vector de ganancias que obtienen las empresas al dedicar menos recursos de los necesarios para finalizar a tiempo a las actividades.

En el caso particular de un problema de retrasos en proyectos, el conjunto de jugadores se corresponderá, igual que en el caso cooperativo, con el conjunto de actividades, añadiendo al planificador que denotaremos como jugador 0.

Cada conjunto de información para el jugador, o empresa, i está caracterizado por el instante t en el cual la actividad tiene que empezar a realizarse, el conjunto de actividades S que ya han finalizado en ese instante y el vector de retrasos $d_S = (d_j)_{j \in S}$ de las actividades ya completadas. El instante t

estará definido en el intervalo

$$\left[\max_{\pi\in\Pi:i\in\pi}\sum_{j\in Pre(i,\pi)}t_j,\max_{\pi\in\Pi:i\in\pi}\sum_{j\in Pre(i,\pi)}(t_j+D_j)\right],$$

definido entre el sumatorio de los tiempos de las actividades predecesoras y el sumatorio de los tiempos y el máximo retraso posible (D_j) de cada actividad predecesora de i. El conjunto S queda definido, formalmente, como

$$S=\left\{j\in N:\max_{\pi\in\Pi:j\in\pi}\sum_{k\in Pre(j,\pi)\cup\{j\}}(t_k+d_k)\leq t\right\}.$$

La función de utilidad dependerá de la penalización establecida y del retraso. Se definirá de forma diferente dependiendo de si es la función de utilidad del planificador, que tendrá una ganancia con la penalización y un coste por el retraso, o de las empresas, que tendrán un coste por la penalización y una ganancia al dedicar recursos a actividades alternativas. Obviamente, tanto planificador como empresas obtendrán una ganancia por completar las actividades y el proyecto, pero podemos simplificar la función de utilidad omitiendo estas ganancias que entenderemos fijas y aseguradas para cada participante en el juego.

Supondremos que la penalización por los retrasos en el proyecto será un valor constante p por unidad de tiempo, en lugar de una función más general. Utilizamos esta penalización lineal con la intención de ajustarla a los criterios de la ley española y, en consecuencia, de los contratos públicos que se realizan en España, que han motivado esta modelización.

Ejemplo 2.3 *Sea un proyecto ilustrado por la red G dada en la Figura 2.7 y sean, para cada actividad $i\in N$, los datos que se muestran en el Cuadro 2.3 de duración (t), retraso máximo permitido (D) y utilidad por dedicar recursos a otras actividades (r). Tenemos que el coste del retraso para el planificador es $c=6$.*

Vemos en la Figura 2.7 que la red tiene una actividad ficticia, que resultará irrelevante en el estudio, por lo que podremos omitirla.

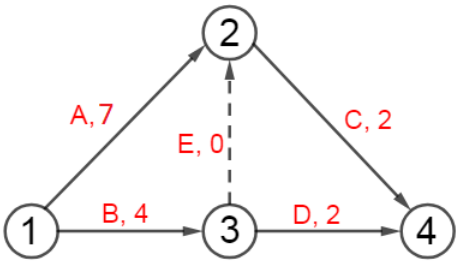

Figura 2.7: Grafo del problema

Estos datos son los necesarios para plantear y estudiar el problema de retrasos en proyectos (G,c,r) con juegos no cooperativos, en el que el planificador deberá plantear una penalización p, y las empresas deberán elegir cuánto retrasan sus actividades.

Actividad	t_i	D_i	r_i
A	7	4	0
B	4	4	5
C	2	2	2
D	2	3	5

Cuadro 2.3: Datos de las actividades.

Los caminos de la red son $\pi_1 = A - C$, $\pi_2 = B - C$ y $\pi_3 = B - D$ y sus duraciones son 9, 6 y 6 respectivamente. Entonces, el tiempo del proyecto es $T = 9$, que viene determinado por el camino π_1. La holgura es 3 para cada uno de los caminos π_2 y π_3.

A continuación estudiaremos dos casos, primero el supuesto en que la penalización p se aplique solo en el caso de que todo el proyecto se vea retrasado, y en segundo lugar el caso en que las penalizaciones se apliquen a todas las actividades retrasadas independientemente de si el proyecto se retrasa o no.

Penalizaciones solo aplicadas cuando el proyecto es retrasado

Como ya hemos adelantado, utilizaremos un juego en 3 etapas para modelar el problema. Formalmente, sea el problema de retrasos en proyectos (G, c, r), asociamos el juego no cooperativo $\Gamma(G, c, r)$, definido como:

- Etapa 1: El planificador decide la penalización $p \in [0, +\infty)$.
- Etapa 2: Las empresas, siguiendo la propia estructura de la red del proyecto, deciden el vector de retrasos $d = (d_i)_{i \in N}$.
- Etapa 3: El planificador paga a las empresas y recauda las penalizaciones, de donde se obtienen las siguientes funciones de utilidad:

$$u_i(p,d) = \begin{cases} 0 & \text{si} \quad i = 0 \text{ y } T(d) \leq T \\ r_i d_i & \text{si} \quad i \in N \text{ y } T(d) \leq T \\ \sum_{i \in N} p d_i - c(T(d) - T) & \text{si} \quad i = 0 \text{ y } T(d) > T \\ (r_i - p) d_i & \text{si} \quad i \in N \text{ y } T(d) > T \end{cases}$$

Observemos que cuando para un determinado $i \in N$, $r_i = p$, y el proyecto se ha retrasado, significa que la empresa i no tiene ni ganancias ni pérdidas al retrasar el proyecto. En estos casos, supondremos

que la empresa no causa retraso y prefiere finalizar su actividad en el tiempo planeado. Resulta lógico asumir que la empresa no retrasará su actividad si no tiene incentivos económicos para ello, aunque solo sea por su propia reputación.

Ejemplo 2.4 *Continuando con el problema de retrasos (G, c, r) presentado en el Ejemplo 2.3, el problema se desarrollaría como sigue:*

- *Etapa 1: El planificador escoge como penalización $p = 4$.*
- *Etapa 2: Las empresas A y B deciden su retraso a la vez. Una vez finalizada la actividad B la empresa D decide su retraso. Cuando finalicen las actividades A y B la empresa C decidirá su retraso. Las empresas C y D pueden escoger simultáneamente o la C después de la D dependiendo de los retrasos de las actividades A y B. Tenemos que el vector de retrasos escogido por las empresas es (0,2,2,2). Debido a las estrategias de las empresas, el proyecto se retrasa 2.*
- *Etapa 3: Calculamos las diferentes utilidades. La utilidad del planificador será 4, mientras que las utilidades de las actividades A, B, C y D serán 0, 2, -4 y 2 respectivamente.*

1. Equilibrio en la etapa 2

Vamos a caracterizar el NE en el subjuego de $\Gamma(G, c, r)$ inducido por la etapa 2, una vez se ha establecido una penalización p por el planificador. Previamente debemos presentar una serie de definiciones que serán necesarias en esta caracterización.

Definición 2.24 *Sea $\Gamma(G, c, r)$ el juego inducido por el problema (G, c, r) y dada una penalización p, definimos la utilidad de mínimo derecho de la empresa i como la utilidad que se puede asegurar cuando el proyecto se ha retrasado y la actividad se retrasa lo máximo posible. La denotaremos por $u_i^{mr}(p)$,*

$$u_i^{mr}(p) = \begin{cases} (r_i - p)D_i & si \quad r_i > p \\ \\ 0 & si \quad r_i \leq p \end{cases}$$

Podemos entender que la utilidad de mínimo derecho es la cantidad que se puede garantizar la empresa i como mínimo aunque el proyecto se retrase.

Definición 2.25 *Sea $\Gamma(G, c, r)$ el juego inducido por el problema (G, c, r) y dada una penalización p, definimos el retraso de mínimo derecho de la empresa i, como el retraso de la actividad i que le asegura obtener su utilidad de mínimo derecho cuando el proyecto acaba en tiempo. La denotaremos por $s_i(p)$,*

$$s_i(p) = \begin{cases} \frac{(r_i - p)D_i}{r_i} & si \quad r_i > p \\ \\ 0 & si \quad r_i \leq p \end{cases}$$

Todas las empresas querrán garantizarse, al menos, su utilidad de mínimo derecho, lo que supondrá que se retrasen como mínimo s_i. Fijada una penalización p, podremos predecir un retraso del proyecto viendo si los sumatorios de los retrasos s_i son mayores que las holguras de los caminos. Si las holguras no pueden cubrir estos retrasos, el proyecto se retrasará.

Definición 2.26 *Sea $\Gamma(G, c, r)$ el juego inducido por el problema (G, c, r) y dada una penalización p, definimos $F(p)$ como el conjunto de retrasos de actividades iguales o mayores a su correspondiente retraso mínimo por derecho y que no generan un retraso en el proyecto. Formalmente,*

$$F(p) = \{x \in \mathbb{R}^N : x_i[s_i(p), D_i] \text{ y } \sum_{i \in \pi} x_i \leq ps_\pi \text{ para todo } \pi \in \Pi\}.$$

Bergantiños y Lorenzo (2017) describen los NE en la etapa 2 diferenciando en dos casos:

1. Si existe un camino $\pi' \in \Pi$ tal que $\sum_{i \in \pi'} s_i(p) > ps_{\pi'}$, entonces el NE es único, y el proyecto se verá retrasado. En este caso, toda actividad i tal que $r_i > p$ se retrasará al máximo (D_i), mientras que el resto no se retrasarán nada.
2. Si $\sum_{i \in \pi} s_i(p) \leq ps_\pi$ para todo $\pi \in \Pi$, entonces cualquier asignación $\{(r_i x_i)_{i \in N} : x \in PB(F(p))\}$ estará asociada a un NE. Existen, por tanto, NE en los que el proyecto no se retrasa.

Ejemplo 2.5 *Siguiendo el problema presentado en el Ejemplo 2.3, calculamos las funciones de utilidad mínima por derecho y de retraso de mínimo derecho que presentamos en el Cuadro 2.4 para una penalización $p = 4$.*

Actividad	$u_i^{mr}(4)$	$s_i(4)$
A	0	0
B	4	0.8
C	0	0
D	3	0.6

Cuadro 2.4: Utilidades mínimas de derecho y retrasos de mínimo derecho.

Para ninguno de los caminos tenemos que la suma de los retrasos de mínimo derecho de sus actividades es mayor que la holgura del camino, por lo que existen NE en los que el proyecto finaliza en el tiempo estipulado.

El conjunto $F(p)$, para $p = 4$, es $\{x \in \mathbb{R}^4 : x_1 = x_3 = 0, x_2 \in [0.8, 4], x_4 \in [0.6, 3], x_2 + x_4 \leq 3\}$. Su frontera de Pareto será el conjunto $\{(0, y, 0, 3 - y) : y \in [0.8, 2.4]\}$.

Si nos situamos en el caso descrito en el segundo punto, cuando existen NE en los que el proyecto no se retrasa, algunos NE pueden llevarnos a un vector de retrasos perteneciente a $F(p)$ que sea dominado en el sentido de Pareto por otro, también perteneciente a $F(p)$. Veamos a continuación un ejemplo.

Ejemplo 2.6 *Sea un proyecto ilustrado por la red G dada en la Figura 2.8 y sean, para cada actividad $i \in N$ no ficticia, los datos que se muestran en el Cuadro 2.5 de duración (t), retraso máximo permitido (D) y utilidad por dedicar recursos a otras actividades (r). Supongamos una penalización por retraso $p = 3$.*

Podemos ver fácilmente que la única actividad que no tendrá incentivos para retrasarse será la actividad D, pues $r_D = 0 \leq 3$.

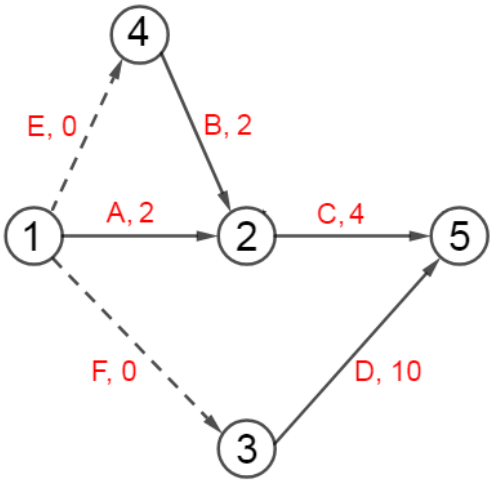

Figura 2.8: Grafo del problema

Actividad	t_i	D_i	r_i
A	2	3	4
B	2	3	4
C	4	5	4
D	10	5	0

Cuadro 2.5: Datos de las actividades.

Los retrasos de mínimo derecho para las actividades A, B y C son $\frac{3}{4}$, $\frac{3}{4}$ $\frac{5}{4}$ *respectivamente, por lo que existen NE en los que el proyecto no se retrasa. Con estos datos, supongamos que las empresas definen los retrasos como siguen, en función de los conjuntos de información I que manejan en el momento de iniciar la actividad:*

- $d_A(I) = 1 \ \forall I.$
- $d_B(I) = 2 \ \forall I.$
- $d_C(I) = \begin{cases} 2 & si \quad d_A(I) = 1 \\ 5 & si \quad d_A(I) \neq 1 \end{cases}$
- $d_D(I) = 0 \ \forall I.$

Podemos ver que si $d_A(I) = 1$ tenemos un NE, ya que los retrasos de las cuatro actividades son mayores, o iguales, a sus respectivos retrasos de mínimo derecho y si la actividad A se retrasa más, aunque esto no genere un retraso de por sí, producirá que la actividad C se retrase 5 y las utilidades serán menores para las 3 empresas, al retrasarse el proyecto.

A pesar de que el vector de retrasos $(1, 2, 2, 0)$ es un NE para el subjuego generado en la etapa 2, no parece la mejor opción para el comportamiento de las empresas. Puesto que el vector de retrasos $(2, 2, 2, 0)$ domina en el sentido de Pareto a $(1, 2, 2, 0)$ y además pertenece a $F(3)$.

Bergantiños y Lorenzo (2017) especifican que, de existir NE que no retrasan el proyecto, el conjunto de estos cuyos pagos no están dominados en el sentido de Pareto es el siguiente:

$$\{(r_i x_i)_{i \in N} : x \in PB(F(p))\}.$$

Algunos de estos NE, a pesar de no estar dominados en el sentido de Pareto, pueden no ser buenas predicciones del comportamiento de las empresas. Esto se debe a que en algunas situaciones las empresas deberán elegir su estrategia mientras otras actividades están siendo realizadas, por lo que las suposiciones sobre el comportamiento de estas puede llevar a retrasos innecesarios que causen un menor beneficio para las empresas. Por ello, será conveniente que las empresas predigan que las actividades que se realizan en paralelo no causarán un retraso innecesario. Ilustramos como algunos NE pueden no ser una buena predicción mediante un ejemplo.

Ejemplo 2.7 *Consideremos el proyecto presentado en el Ejemplo 2.1 con los datos que se muestran en el Cuadro 2.6 y una penalización $p = 1.9$.*

Actividad	t_i	D_i	r_i
A	3	4	0
B	6	4	2
C	5	4	2
D	10	4	0

Cuadro 2.6: Datos de las actividades.

Podemos ver que las actividades A y D tienen como estrategia dominante en cualquier caso el no retrasarse, pues $r_A = r_D = 0$. En consecuencia, sus utilidades serán siempre 0.

Tenemos dos NE para el subjuego en la etapa 2, uno en el que el proyecto finaliza en el tiempo estipulado y otro en el que el proyecto finaliza con retraso. En el primero tenemos el vector de retrasos $(0, 1, 2, 0)$, que generará un vector de utilidades $(0, 2, 4, 0)$. Por lo que las actividades B y C tendrán una ganancia de 2 y 4 respectivamente.

En el segundo NE las dos actividades (B y C) se retrasan lo máximo posible, por lo que tendremos un vector de retrasos $(0, 4, 4, 0)$, que generará un vector de utilidades $(0, 0.4, 0.4, 0)$.

Notemos que el segundo NE estaría dominado en el sentido de Pareto, en términos de de utilidades, por el primero, por lo que podemos pensar que el primer NE sería una mejor predicción para el comportamiento de las empresas.

Parece necesario, visto el anterior ejemplo, presentar un NE cuyos pagos no estén dominados en el sentido de Pareto para los casos en lo que existan NE que no retrasan el proyecto. Definimos entonces el NE optimista bajo la suposición de que cada empresa actuará considerando que el resto no incurrirá en un retraso del proyecto salvo que sea beneficioso para ellas, i.e. salvo que exista un camino $\pi' \in \Pi$ tal que $\sum_{i\in\pi'} s_i(p) > ps'_\pi$.

Sea la empresa $i \in N$ e $I = (t, S, (d_j)_{j\in S})$ su correspondiente conjunto de información. Definimos $Par(I)$ como el conjunto de actividades que se están realizando en paralelo a i en el instante t.

Ahora definimos el vector de creencias optimistas $o(I) \in \mathbb{R}^N$ para la empresa i en el conjunto de información $I = (t, S, d_S)$. Para ello, consideramos varios casos:

- $j \in S$: La empresa i sabe perfectamente el retraso de la empresa j, pues ya ha finalizado. Por lo que,
$$o_j(I) = d_j.$$
- $j \in Par(I)\backslash\{i\}$: La empresa i conoce el tiempo de inicio de la actividad j,pues es una actividad que se realiza en paralelo, y cree que no retrasará el proyecto salvo que sea necesario para obtener su utilidad de mínimo derecho, por lo que esperará un retraso de la actividad j de $s_j(p)$. En el caso de que ese retraso ya hubiese sido superado, la empresa i supondrá que la actividad j finalizará inmediatamente. Formalmente,
$$o_j(I) = \text{máx}\{s_j(p), t - t_j - \max_{j\in\pi\in\Pi} \sum_{k\in Pre(j,\pi)} (t_k + d_k) + \epsilon\}.$$
- $j \in N\backslash\{S \cup Par(I)\}$: En este caso i no tiene ninguna información acerca de la empresa j, ya que ni ha finalizado, ni transcurre en paralelo. Por ello supondrá que esta empresa se retrasará $s_j(p)$. Por lo que,
$$o_j(I) = s_j(p).$$

Realizamos una última definición relacionada con el vector de creencias optimistas. Consideremos una empresa i y su correspondiente conjunto de información I, y sea $x = (x_j)_{j\in N\backslash(Suc(i)\cup\{i\})}$ tal que $x_j = d_j$ para todo $j \in S$, definimos el juego $\Gamma^{I,x}(G, c, r)$ como el juego inducido por $\Gamma(G, c, r)$ en el conjunto de información I. En el juego $\Gamma^{I,x}(G, c, r)$ los jugadores son las empresas $Suc(i) \cup \{i\}$, mientras que el resto de empresas $j \in N\backslash(Suc(i) \cup \{i\})$ tiene una duración fijada de $t_j + x_j$. Obtenemos un juego no cooperativo en forma extensiva en el que la primera decisión le corresponde a la empresa i.

Bergantiños y Lorenzo (2017) prueban que no solo existe un NE optimista para el juego $\Gamma(G, c, r)$ inducido por (G, c, r), sino que además es único.

Este NE optimista vendrá determinado, en el caso de que el proyecto no sufra retraso, por los retrasos de mínimo derecho. Cada actividad i tendrá un retraso de $s_i(p)$, pero esto supondrá, en general, que algunos caminos finalizan antes que el conjunto del proyecto. Ese tiempo podrá ser utilizado por las empresas para aumentar su utilidad sin retrasar el proyecto, por lo que algunas actividades tendrán un retraso mayor que su retraso de mínimo derecho. Realizando inducción hacia atrás, las actividades finales con un $s_i(p) > 0$ se retrasarán tanto como les sea posible sin llegar a causar un retraso en el proyecto, si alcanzan un retraso igual a D_i entonces sus predecesoras inmediatas serán las que aumenten el retraso por encima de $s_i(p)$ en la medida de lo posible, y así sucesivamente hasta

que ningún camino tenga una holgura aprovechable.

En el caso de que al jugarse el NE optimista se produzca un retraso en el proyecto, el NE optimista coincidirá con el único NE del juego, en el que el retraso de las actividades con $s_i(p)$ mayor que cero es igual su retraso máximo D_i y el retraso del resto de actividades será 0.

Ejemplo 2.8 *Continuando con el Ejemplo 2.5 y apoyándonos en los cálculos ahí realizados, vamos a obtener el NE optimista para la etapa 2 del juego no cooperativo asociado al problema de retrasos en proyectos presentado en el Ejemplo 2.3.*

Las actividades A y C tendrán un retraso igual a cero, pues su utilidad de mínimo derecho también es cero. En cambio las actividades B y D tendrán un retraso mayor que cero. Tenemos que $ps_{\pi_3} = 3$ *y que* $s_B(4) + s_C(4) = 0.8 + 0.6 = 1.4$*, por lo que si las dos empresas retrasan sus actividades su correspondiente retraso de mínimo derecho aún habría una holgura de 1.6 en el camino* π_3*.*

Haciendo inducción hacia atrás, la holgura sobrante podría ser aprovechada por la empresa D, que no alcanzaría su retraso máximo permitido, por lo que la actividad B no llegaría a tener un retraso extra. Entonces tenemos que el vector de retrasos de las actividades jugando un NE optimista será (0, 0.8, 0, 2.2). Las utilidades para las empresas A y C y para el planificador serán 0, mientras que para B y D serán 4 y 11.

2. Equilibrio en el juego

En esta sección vamos a estudiar el equilibrio en todo el juego, centrándonos en calcular la penalización que maximice la utilidad del planificador. Para ello supondremos que las empresas jugarán un NE óptimo, que no sea dominado en el sentido de Pareto, no necesariamente un NE optimista.

Asumiendo que las empresas juegan un NE óptimo en la etapa 2 del juego $\Gamma(G, c, r)$ inducido por (G, c, r) para cualquier penalización p, entonces existe una penalización p^*_Γ tal que si y solo si $p < p^*_\Gamma$ el proyecto se verá retrasado.

El resultado principal sobre el equilibrio en el juego $\Gamma(G, c, r)$ nos proporciona la penalización con la que el planificador obtendrá la mayor utilidad. Omitiremos la demostración, que se puede consultar en Bergantiños y Lorenzo (2017).

Proposición 2.7 *Sea* $\Gamma(G, c, r)$ *el juego no cooperativo inducido por* (G, c, r)*. Suponemos que para cada penalización p las empresas juegan un NE* $d(p)$ *sin retrasos (cuando exista) y el único NE con retrasos cuando no exista un NE sin retrasos en la etapa 2. Entonces,* $\sup\{u_0(p, d(p)) : p \geq 0\}$ *se alcanza para un* $p \in \{p^*_\Gamma, \{r_j\}_{j \in N: r_j < p^*_\Gamma}\}$*.*

Este resultado simplifica significativamente el problema de encontrar la penalización óptima para el planificador, pues restringe los potenciales valores a un conjunto formado por las utilidades que obtienen de dedicar recursos a otras actividades de cada empresa y la penalización mínima que evita un retraso del proyecto. Así, a la hora de buscar la penalización óptima bastará con estudiar como varía la utilidad del planificador en un conjunto finito de valores.

Notemos que, a pesar de escribir que el conjunto está conjunto formado por los r_i de las actividades y la penalización mínima que garantiza que el proyecto no se retrasará, la máxima utilidad para el planificador se alcanzaría cuando p converja por la izquierda a un r_i, con $i \in N$, o a p^*_Γ. En la práctica tomaremos un $r_i - \epsilon$, o $p^*_\Gamma - \epsilon$, donde ϵ sea lo suficientemente pequeño.

Ejemplo 2.9 *Continuamos estudiando el problema introducido en el Ejemplo 2.3. Vamos a calcular la mínima penalización que evita un retraso del proyecto. Ya hemos visto que con una penalización de 4 el proyecto no sufre retraso, por lo que el único camino en el que debemos estudiar los posibles retrasos será* π_3 *(si cualquiera de los otros dos caminos se retrasa,* π_3 *también). Entonces, el proyecto se retrasará cuando* $s_B(p) + s_D(p) > ps_{\pi_3}$*. Resolvemos esa ecuación y obtenemos que la penalización mínima que evita el retraso del proyecto es* $p^*_\Gamma = 2.86$*.*

Vamos a calcular ahora la penalización que maximiza la utilidad del planificador. Siguiendo el resultado ofrecido por la Proposición 2.7, la penalización que determina el supremo de las utilidades del planificador se encontrará en el conjunto $\{0, 2, 2.86, 5\}$*, aunque podemos descartar la penalización* $p = 0$*, pues causaría un coste por el retraso y ningún beneficio por la penalización. Estudiamos qué ocurre cuando p converge por la izquierda hacia los valores del conjunto, y vemos que el único caso en el que la utilidad del planificador es mayor que 0 es cuando converge hacia 5, por lo que* $p = 5$ *será la penalización que determine el supremo de las utilidades.*

Penalizaciones aplicadas sobre cualquier actividad retrasada

El estudio del problema en el que las penalizaciones se aplican a cualquier actividad retrasada resulta necesario ya que en la práctica, en determinados proyectos, algunas actividades retrasadas pueden causar un coste por sí mismas. Un ejemplo de esto puede ser una vía de comunicación, como una carretera, en la que el retraso de sus tramos individualizados supone un coste aunque el proyecto finalice a tiempo.

Modelizaremos este problema, igual que el caso donde las penalizaciones se aplican solo cuando el proyecto en su conjunto es retrasado, mediante un juego no cooperativo en forma extensiva en tres etapas. Denotaremos al juego asociado al problema $\Delta(G, c, r)$, donde:

- Etapa 1: El planificador decide la penalización $p \in [0, +\infty)$.
- Etapa 2: Las empresas, siguiendo la propia estructura de la red del proyecto, deciden el vector de retrasos $d = (d_i)_{i \in N}$.
- Etapa 3: El planificador paga a las empresas y recauda las penalizaciones de donde se obtienen las siguientes funciones de utilidad:

$$u_i(p, d) = \begin{cases} \sum_{i \in N} pd_i - c(T(d) - T) & \text{si} \quad i = 0 \\ (r_i - p)d_i & \text{si} \quad i \in N \end{cases}$$

1. Equilibrio en la etapa 2

Fijada una penalización p, analizar cuál será el NE del subjuego definido en la etapa 2 resulta más sencillo que el caso anterior. La estrategia de las empresas tales que $r_i \leq p$ será no retrasar sus actividades, mientras que el resto las retrasarán todo lo posible. Obtenemos así el NE para el subjuego,

en el que para todo $i \in N$ y para cualquier conjunto de información I de la empresa i,

$$d_i(I) = \begin{cases} 0 & \text{si} \quad r_i \leq p \\ \\ D_i & \text{si} \quad r_i > p \end{cases}$$

Ejemplo 2.10 *Sea el problema presentado en el Ejemplo 2.3, el NE para el juego será aquel en el que las actividades A y C no se retrasan y las actividades B y D se retrasan lo máximo posible. Así, el vector de retrasos será* $d = (0, 4, 0, 3)$*. El proyecto finalizará con un retraso de 4, y la utilidad del planificador, para un coste del retraso* $c = 6$ *y una penalización* $p = 4$*, será 4. Las utilidades de las empresas se encuentran en el Cuadro 2.7.*

Actividad	$u_i(p, d)$
A	0
B	4
C	0
D	3

Cuadro 2.7: Utilidades de las empresas para $p = 4$.

Vemos que, para este tipo de penalización, la utilidad que ofrece el único NE del subjuego generado en la etapa 2 coincide con la utilidad de mínimo derecho.

2. Equilibrio en el juego

Los resultados que obtenemos en el equilibrio para el juego en el caso en el que se penaliza toda actividad retrasada son similares a los del caso en el que las penalizaciones se aplican solo cuando el proyecto se retrasa, solamente precisamos asumir que las empresas jugarán el único NE en la etapa 2.

Podemos ver, igual que en el caso anterior, que si en el juego $\Delta(G, c, r)$ asumimos que para toda penalización p las empresas juegan un NE en la etapa 2 entonces, existe una penalización p^*_Δ tal que el proyecto se retrasará si y solo si $p < p^*_\Delta$. Además, tenemos que $p^*_\Delta \in \{r_i\}_{i \in N}$.

El resultado principal sobre el equilibrio en el juego $\Delta(G, c, r)$ nos proporciona la penalización con la que el planificador obtendrá la mayor utilidad. Omitiremos la demostración, que se puede consultar en Bergantiños y Lorenzo (2017).

Proposición 2.8 *Sea* $\Delta(G, c, r)$ *el juego no cooperativo inducido por* (G, c, r)*. Suponemos que para cada penalización* p *las empresas juegan un NE* $d(p)$ *en la etapa 2. Entonces,* $\sup\{u_0(p, d(p)) : p \geq 0\}$ *se alcanza para un* $p \in \{r_i\}_{i \in N}$*.*

Igual que en el caso anterior, gracias a este resultado la búsqueda de la penalización óptima para el planificador se restringirá a un conjunto finito. Notemos que en este caso, el conjunto de potenciales

penalizaciones óptimas no incluye expresamente la penalización mínima que evita un retraso del proyecto, pues esta ya pertenece a $\{r_i\}_{i\in N}$.

Proposición 2.9 *Continuando con el problema del Ejemplo 2.3, y para el que hemos visto en el Ejemplo 2.10 como una penalización $p = 4$ causaba un retraso del proyecto, podemos afirmar que la penalización mínima que evitará el retraso será $p = 5$. Ya que, con dicha penalización, ninguna actividad tendrá incentivos para retrasarse ($r_i \leq 5\ \forall i \in N$) y en consecuencia el proyecto tampoco se retrasará.*

Además, la penalización $p = 5 - \epsilon$, con $\epsilon > 0$, también marcará el supremo de la utilidad del planificador, ya que cuando p converge a 5 por la izquierda $u_0(p, d)$ toma valores cercanos a 9, mientras que al converger a 2 por la izquierda la utilidad es negativa.

Comparación de los diferentes casos

En esta sección vamos a comparar los dos casos presentados y estudiados previamente. Sean $\Gamma(G, c, r)$ y $\Delta(G, c, r)$ los juegos no cooperativos en forma extensiva con los que modelizamos los problemas de retrasos en proyectos cuando la penalización se efectúa si el proyecto se retrasa o si cualquier actividad se retrasa, respectivamente. En primer lugar veamos las diferencias que surgen entre los dos métodos de penalización en los subjuegos de $\Gamma(G, c, r)$ y $\Delta(G, c, r)$ generados en la etapa 2, dada una penalización p, y suponiendo que las empresas siguen un NE:

- La utilidad del planificador no está relacionada entre los dos juegos.
- La utilidad de cada empresa es igual o mayor en el subjuego de $\Gamma(G, c, r)$. Esto es debido a que al aplicar la penalización solo cuando el proyecto se retrasa, nos encontraremos con casos en los que aunque algunas actividades sufran retraso no se aplique ninguna penalización. Al contrario que en el juego $\Delta(G, c, r)$ en el que siempre habrá una penalización si existe cualquier retraso.
- El retraso del proyecto es igual o mayor en el subjuego de $\Delta(G, c, r)$. Aunque resulte contradictorio, ya que en este caso hay mas penalizaciones, que el retraso sea igual o mayor en el subjuego de $\Delta(G, c, r)$ se debe a que cuando una actividad se retrasa, lo hará lo máximo posible. Mientras que en el juego $\Gamma(G, c, r)$ las empresas que decidan retrasar sus actividades pueden obtener el mismo beneficio sin llegar a causar un retraso del proyecto.

Una vez comparada la etapa 2, comparamos el juego completo. Suponemos que en la etapa 2 del juego $\Gamma(G, c, r)$ las firmas juegan cualquier NE óptimo.

- La penalización óptima para el planificador es igual o mayor en $\Delta(G, c, r)$.
- La utilidad del planificador es igual o mayor en $\Delta(G, c, r)$.
- La utilidad de las empresas es igual o mayor en $\Gamma(G, c, r)$.
- El retraso en el proyecto no está relacionado.

De esta comparación podemos concluir que las empresas preferirán una penalización solo en el caso de que el proyecto se retrase, pues su utilidad será potencialmente mayor.

Por otra parte, las preferencias del planificador son más complejas. Si el planificador es una empresa privada, preferirá un sistema que penalice toda actividad retrasada, pero si el planificador es una institución pública, puede tener sentido la penalización solo cuando el proyecto se retrasa en orden de beneficiar a las empresas, que son parte de la sociedad a la que sirve dicha institución pública.

Ejemplo 2.11 *Para finalizar con esta sección, comparamos el problema presentado en el Ejemplo 2.3 y que hemos estudiado para los dos casos. Para una penalización $p = 4$ el NE optimista del subjuego generado por la etapa 2 tiene unas utilidades para las empresas B y D de 4 y 11, mientras que en el subjuego de $\Delta(G, c, r)$ esas utilidades son iguales o menores, concretamente 4 y 3, a pesar de que el proyecto se retrasa, al contrario que en el subjuego de $\Gamma(G, c, r)$, en el que el proyecto finaliza en el tiempo estipulado.*

Además, la penalización óptima para el planificador es igual en el juego $\Delta(G, c, r)$ que en el $\Gamma(G, c, r)$, 5 en ambos casos, pero garantiza una mayor utilidad al planificador en el juego $\Delta(G, c, r)$. Al contrario, las utilidades de las empresas son iguales o mayores en $\Gamma(G, c, r)$, como podemos ver en el Cuadro 2.8, donde están los valores a los que convergen las utilidades cuando p converge a la penalización que determina el supremo de las utilidades del planificador, suponiendo que en el juego $\Gamma(G, c, r)$ las empresas juegan el NE optimista.

Actividad	$\Gamma(G, d, r)$	$\Delta(G, d, r)$
A	0	0
B	8.46	0
C	0	0
D	6.42	0

Cuadro 2.8: Utilidades de las empresas para ambos juegos.

Capítulo 3

Programación

El objetivo de este capítulo es ofrecer un código que calcule, para los dos juegos no cooperativos presentados en el Capítulo 2, las estrategias óptimas de las empresas y las penalizaciones óptimas del planificador (dependiendo de si su objetivo es obtener la máxima utilidad o evitar que el proyecto se retrase). Para el caso en el que solo se aplican las penalizaciones cuando el proyecto se retrasa, calcularemos las estrategias de las empresas que se corresponden con el NE optimista, mientras que para el caso en el que se penaliza toda actividad que se retrase, calcularemos las estrategias del único NE del juego.

Será necesario, en primer lugar, construir un modelo PERT/CPM sobre el que realizaremos los cálculos mostrados en el Capítulo 1:

- De cada actividad: la holgura, el tiempo inicial mínimo y el tiempo final máximo.
- De cada camino: las actividades que lo componen, la duración total y la holgura.
- Del proyecto: la duración.

Todo el código presentado en este capítulo, así como en los apéndices correspondientes, está realizado con el lenguaje y el entorno para la computación estadística R.

3.1. Modelo PERT/CPM

Crearemos una función principal, que denominaremos *CalculoPert*, destinada a calcular y devolver los resultados del método PERT/CPM. En esta función calcularemos, en primer lugar, una matriz que contenga todos los caminos de la red del proyecto, a partir de la cual obtendremos una matriz análoga con los tiempos de las actividades, que nos permitirá realizar los cálculos necesarios. La función *CalculoPert* empleará, a su vez, diversas funciones secundarias que realizarán cálculos concretos.

Dividiremos el proceso en tres partes; en primer lugar explicaremos cómo introducir los datos necesarios, en segundo lugar daremos unas nociones básicas sobre los cálculos realizados y por último trataremos los resultados obtenidos con la función.

3.1.1. Introducción de los datos

Elegimos introducir los datos en R a partir de un archivo *JSON*, ya que resulta un formato estándar rápido y ligero que nos permitirá obtener los datos de forma sencilla a partir de otros lenguajes de

programación y tratamiento de datos. Tenemos que tener en cuenta que los datos introducidos no los podremos leer como una tabla, al no tener un tamaño regular, por lo que utilizar un archivo *JSON* resulta más conveniente que, por ejemplo, un *csv*. Utilizaremos el paquete *jsonlite* para leer los datos en R.

```
fromJSON("../../TFM/programacionR/datos.json")
```

Una vez introducidos los datos a partir del archivo generamos un objeto *list* en el que deberán estar incluidos los dos siguientes elementos:

- Un *data frame* que llamaremos *datosActividades* que como mínimo tendrá una columna con los tiempos de cada actividad.
- Otro objeto *list predecesores*, donde se listará cada actividad con un vector formado por sus actividades predecesoras.

Para una mayor simplificación denominaremos a cada actividad por un número, para facilitar su identificación en la posición de un vector. Aunque no es necesario mientras mantengan el mismo orden.

En el Apéndice A podemos encontrar dos ejemplos de como construir los archivos *JSON* necesarios para introducir los datos. Utilizamos para crear estos archivos los datos del Ejemplo 1.1 y del Ejemplo 2.3.

3.1.2. Cálculos

Procedemos ahora a comentar y explicar los cálculos realizados en la función *CalculoPert* a la que le hemos pasado como argumentos el objeto *list datos*.

Creamos, para facilitar el trabajo, un vector con los datos de tiempos para las actividades, un objeto *list* con la información de las actividades predecesoras y definiremos n como el número de actividades del proyecto.

```
  t<-datos$datosActividades$t_i
  Pre<-datos$predecesores$predecesores
  n=length(t)
```

A partir del objeto *list* de actividades predecesoras obtenemos la matriz A de ceros y unos que contenga la información de las actividades sucesoras para cada actividad. Identificamos cada actividad con cada columna y asignamos 1 a la fila que se corresponda con una actividad sucesora y 0 en caso contrario. Para obtener esta matriz utilizamos la siguiente función:

```
matrizSucesoresInmediatos<-function(Pre,n){
  A=matrix(0,nrow=n,ncol=n) #futura matriz de los sucesores inmediatos de cada actividad (matriz de
       ceros y unos)
  for (i in 1:n){
    A[Pre[[i]],i]=1 #en cada columna ponemos un 1 en la fila que se corresponda a una actividad
         predecesora
  }
  return(A)
}
```

El dato (i, j) de la matriz A será 1 si y solo si la actividad i es predecesora de j, en caso contrario será 0. Teniendo esto en cuenta, podemos ver cuáles son las actividades iniciales y finales fijándonos en las columnas y filas de ceros.

```
  b=which(colSums(A) == 0)  #Actividades iniciales
  e=which(rowSums(A) == 0)  #Actividades finales
```

Para obtener la matriz C de caminos del nodo origen al nodo final utilizaremos otra función auxiliar, que llamaremos *matrizCaminos*. Nos apoyaremos en una matriz B que construiremos iterativamente con los caminos intermedios desde las actividades iniciales. Iremos añadiendo filas a medida que sumemos actividades a los caminos, hasta que lleguemos a una actividad final. Las filas de la matriz C serán las actividades, en orden, que compongan los diferentes caminos de la red. Dado que los caminos, en general, no tendrán la misma longitud, la matriz se completará con ceros.

```
matrizCaminos<-function(A,b,n){
  B=matrix(0,nrow=length(b),ncol=n)  #Futura matriz de los caminos intermedios que comienzan en las
       actividades iniciales
  B[1:length(b),1]=b
  C=matrix(0,nrow=1,ncol=n) #Futura matriz de los caminos

  for (i in 2:n) {  #Posicion i-esima del camino
    for (j in which(B[,i-1] != 0)) {  #j-esima fila de la matriz
      if (sum(A[B[j,i-1],]) == 0){ #Primero comprobamos si la actividad inicial es tambien final
        c=B[j,]
        C=rbind(C,c) #Agregamos el camino que solo contiene la actividad
        break
      }
      for (k in seq(1,n,by=1)[-b]) {  #Vemos las actividades que debemos agregar
        if (A[B[j,i-1],k] == 1 && sum(A[k,]) != 0) { #Comprobamos si la actividad k pertenece al
             camino y es intermedia (no final)
          l=B[j,]
          l[i]=k
          B=rbind(B,l) #Agregamos la actividad a la matriz de caminos intermedios
        } else { if (A[B[j,i-1],k] == 1 && sum(A[k,]) == 0) { #Comprobamos si la actividad k
             pertenece al camino y es final
          c=B[j,]
          c[i]=k
          C=rbind(C,c) #Agregamos el camino c a la matriz de caminos
        }}
      }
    }
  }
  return(C[2:length(C[,1]),which(colSums(C) != 0)]) #cada fila es un camino desde un nodo inicial a
       un nodo final
}
```

Definimos una matriz TC, con las mismas dimensiones que la matriz C, donde cada par (i, j) tengamos el tiempo de la actividad correspondiente al par (i, j) de la matriz C. Calculamos esta matriz con la función auxiliar *matrizTiemposCaminos*.

```
matrizTiemposCaminos<-function(C,t,n){
  taux=c() #vector auxiliar de los tiempos (incluimos un cero al inicio para los elementos que no
       tienen actividades)
  for (i in 1:n) {
    taux[i+1]=t[i]
  }
  taux[1]=0
  if(nrow(C) != 0){ #si la matriz de caminos tiene mas de una fila (hay mas de un camino)
    TC=matrix(taux[C+1],nrow=nrow(C)) #matriz de los tiempos de las actividades de los caminos
  } else {
    TC=taux[C+1] #tiempos de las actividades que forman el camino del camino
  }
  return(TC)
}
```

A partir de la matriz TC resulta sencillo obtener el tiempo del proyecto como el máximo del sumatorio de cada fila y las holguras de los caminos como las diferencias de los sumatorios (de cada fila) con el tiempo del proyecto.

```
TP=max(rowSums(TC))  #tiempo del proyecto
HC=TP-rowSums(TC)  #holguras de los caminos
```

Apoyándonos en la matriz *TC* obtenemos el tiempo más temprano de comienzo y más tardío de final y la holgura para cada actividad. Lo hacemos calculando, para cada actividad, los caminos de mayor longitud desde el inicio del proyecto hasta la actividad y desde la actividad hasta el final del proyecto. Para ello, diferenciamos en primer lugar si la actividad correspondiente es la única del camino y de no serlo si es inicial, final o ninguna de las dos. A continuación presentamos el código implementado recortando los casos en los que la actividad es inicial o final por brevedad.

```
for (i in 1:n) {
  tInicialAux=rep(0,m) #Vector auxiliar de los tiempo iniciales
  tFinalAux=rep(0,m) #Vector auxiliar de los tiempos finales
  f=which(C == i)%%m #Vemos los caminos que contienen la actividad i
  f[f==0]=m #Como hacemos el modulo, si i esta en el camino m nos devolvera 0, por eso lo
      arreglamos

  if (length(which(b == i)) != 0) { #Vemos si i es una actividad inicial
    ...
  } else { if (length(which(e == i)) != 0) { #Vemos si es actividad final
    ...
  } else { #Si i no es actividad inicial ni final
    for (j in f) {
      tInicialAux[j]=sum(TC[j,1:(which(C[j,] == i)-1)]) #Sumamos los tiempos de las actividades
          previas de i en el camino j
      tFinalAux[j]=sum(TC[j,(1+which(C[j,] == i)):length(C[j,])]) #Sumamos los tiempos de las
          actividades sucesoras de i en el camino j
    }
    tInicial[i]=max(tInicialAux)
    tFinal[i]=TP-max(tFinalAux)
  }}
  if (length(f) == 1) {
    Ha[i]=TP-sum(TC[f,])  #Holgura de la actividad si solo esta en un camino (holgura del camino)
  } else {
    Ha[i]=TP-max(rowSums(TC[f,])) #Si pertenece a varios caminos, tomamos el de mayor tiempo
  }
}
```

En el Apéndice B adjuntamos el código completo de la función *CalculoPert* y de las funciones auxiliares correspondientes.

3.1.3. Resultados

Creamos el objeto *list* que devolverá la función. Elegimos devolver un objeto de esta clase, pues resulta la manera más sencilla de agrupar toda la información que hemos calculado.

Creamos un objeto que llamaremos *PERT* con los siguientes elementos:

- Un objeto data frame con la información de los tiempos de las actividades, que llamaremos *Actividades* constituido por las siguientes columnas:
  - *Actividad*: asignamos a cada fila una actividad.
  - *Duracion*: devolvemos el dato de duración de cada actividad que introducimos en la función.
  - *TiempoInicialMin*: tiempo más temprano de comienzo de cada actividad.
  - *TiempoFinalMax*: tiempo más tardío de final de cada actividad.
  - *Holgura*: holgura de cada actividad.
- Un objeto *list* con la información correspondiente a los caminos, que llamaremos *Caminos* que a su vez estará formado por:

  - *matrizCaminos*: matriz C calculada mediante la función *matrizCaminos* que contiene las actividades, en orden, que componen los caminos desde el nodo inicial al final.
  - *tiempoActividadesCaminos*: matriz TC de los tiempos de cada actividad que forma el camino con el que se corresponde cada fila.
  - *tiempoCaminos*: vector con la duración de cada camino.
  - *holgurasCaminos*: vector con la holgura de cada camino.
  - *numeroCaminos*: entero que nos indica cuántos caminos componen la red del proyecto.
- Un objeto al que llamamos *Proyecto* formado únicamente por el valor *tiempoProyecto* que se corresponde con la duración total del proyecto.
- Incluimos también la información con la que construimos el modelo PERT/CPM y que ya detallamos al explicar la introducción de los datos.

Por último, creamos un archivo *JSON* a partir de los datos devueltos por la función *CalculoPert*. Volvemos a utilizar para esto el paquete *jsonlite*.

```
exportJSON<-toJSON(PERT)
write(exportJSON,"../../TFM/programacionR/resultados.R")
```

En el Apéndice C incluimos los archivos *JSON* generados a partir de los ejemplos expuestos en el Apéndice A.

3.2. Juego asociado a un modelo PERT/CPM

Una vez obtenido el modelo PERT/CPM y la matriz C de caminos del nodo origen al nodo final, pasamos a programar los dos juegos no cooperativos asociados, vistos en el Capítulo 2. Construiremos la función *JuegoPERT* a la que le pasaremos como atributo una variable que pueda tomar el valor *TRUE* o *FALSE* y que indique si se penaliza toda actividad retrasada en el caso de ser *TRUE*, o no, en caso de ser *FALSE*. De esta forma indicaremos cuál de los juegos no cooperativos vistos estaremos modelizando.

Con la ayuda de la función *CalculoPERT* calculamos para ambos juegos, las estrategias que deberán seguir las empresas y el planificador dada una penalización que proporcionaremos a la función *JuegoPERT*. Obtendremos también la penalización óptima para el planificador y la penalización mínima que garantiza que el proyecto no resultará retrasado para los dos juegos.

3.2.1. Introducción de los datos

Igual que en la función *CalculoPert*, introduciremos un objeto *list* de datos como argumento de la función *CalculoPERT* constituido, igualmente, por otro objeto *list* con los datos de predecesores de cada actividad y un *data frame* con las siguientes variables para cada actividad:

- t_i: duraciones de las actividades.
- D_i: duraciones máximas permitidas.
- r_i: beneficios obtenidos por las empresas por dedicar recursos a actividades en otros proyectos.

Además del objeto de datos, introduciremos como argumentos en la función *CalculoPERT* tres argumentos más:

- p: la penalización establecida por el planificador.
- *cost*: coste para el planificador, por unidad de tiempo, del retraso del proyecto.
- *penalizarCualquierActividadRetrasada*: argumento lógico que determina si se penalizan las actividades retrasadas, aunque no provoquen un retraso en el proyecto, en caso de ser igual a *TRUE*, o si solo se penalizan cuando el proyecto se retrasa, en el caso de ser igual a *FALSE*.

Como en la sección anterior, introduciremos los datos relativos a las actividades y sus relaciones de precedencias mediante un archivo *JSON* que leeremos haciendo uso del paquete *jsonlite*.

El segundo ejemplo de los presentados en el Apéndice A, que se corresponde con los datos del Ejmplo 2.3, también es válido para presentar cómo construir el archivo *JSON* necesario para introducir los datos. Podemos utilizar el mismo archivo, puesto que en la función *CalculoPert* solo leeremos los datos que necesitamos del data frame de actividades, concretamente la variable de duraciones de las actividades.

3.2.2. Cálculos

En primer lugar, creamos los vectores t, D y r de duraciones, duraciones máximas y ganancias por dedicar recursos a otras actividades y el objeto *list Pre* con los datos de predecesoras para cada actividad. Con estos datos obtenemos los resultados del método PERT/CPM mediante la función *CalculoPert* que ya hemos descrito anteriormente, obteniendo así, los resultados que proporciona y que necesitaremos.

A continuación, definimos una estructura de control que nos permitirá elegir si modelamos el juego $\Gamma(G, c, r)$ o $\Delta(G, c, r)$ según el valor de la variable *penalizarCualquierActividadRetrasada* introducida como argumento en la función.

```
if (penalizarCualquierActividadRetrasada == FALSE) {
  #Aqui realizamos los calculos para el caso en el que solo se penaliza al retrasar el proyecto
}
if (penalizarCualquierActividadRetrasada == TRUE) {
  #Aqui realizamos los calculos para el caso en el que se penaliza toda actividad con retraso
}
```

Diferenciamos entre estos dos casos, pero obtenemos los mismos datos para cada juego; un NE del subjuego generado por la etapa 2 y las utilidades que produce, la penalización óptima para el planificador y la penalización mínima que garantiza que el proyecto no resultará retrasado.

Caso en el que solo se penalizan las actividades en caso de retraso del proyecto

Suponiendo la penalización por parte del planificador que hemos introducido como argumento de la función *JuegoPERT*, nos apoyaremos en una función auxiliar para calcular las estrategias que seguirán las empresas en el supuesto de jugar el único NE optimista del subjuego generado en la etapa 2 del juego $\Gamma(G, c, r)$. Además, obtendremos las utilidades de las empresas y del planificador, así como las duraciones de las actividades y del proyecto.

```
modelo1<-function(r,D,C,tC,p,cost){...}
#tC es el vector con los tiempos de los caminos
```

Como hemos visto en el capítulo anterior, los primeros cálculos necesarios para obtener el NE optimista del juego $\Gamma(G, c, r)$ son los retrasos de mínimo derecho $s_i(p)\ \forall i \in N$. Los obtenemos a partir de los vectores r y D para la penalización p dada.

```
calculo_s<-function(n,p,r,D){
  #Calculamos los retrasos de minimo derecho para cada actividad
  s=c()
  for(i in 1:n){
    if (r[i]>p && r[i]!=0){
      s[i]=((r[i]-p)*D[i])/r[i]
    }else{
      s[i]=0
    }
  }
  return(s)
}
```

A partir del vector s de retrasos de mínimo derecho vemos si estos retrasos de las actividades generan que el proyecto finalice más tarde del tiempo estipulado. De ser así, asignamos a toda actividad $i \in N$ con un r_i mayor p un retraso igual a su retraso máximo, $d_i = D_i$. En caso contrario, los retrasos de las actividades se corresponderán con sus retrasos de mínimo derecho.

```
#ts es el vector de tiempos de los caminos sumando los retrasos por minimo derechos
calculo_d_modelo1<-function(n,p,r,D,s,ts,tC){
  #Calculamos los retrasos de las actividades según hay o no retraso del proyecto
  d=c()
  for(i in 1:n){
    if(max(ts) > max(tC) && r[i] > p){ #vemos si habra retraso
      d[i]=D[i]
    }else{
      d[i]=s[i]
    }
  }
  return(d)
}
```

Una vez asignados los retrasos d a las actividades, si el proyecto no sufre ningún retraso, nos encontraremos, en general, con caminos que tienen una holgura mayor que cero. Entonces, repartimos estas holguras de las que disponen los caminos (y en consecuencia las actividades que los forman) entre las actividades que puedan tener un retraso mayor del que ya les hemos asignado. Por eso, no aumentaremos el retraso de las actividades más de su correspondiente retraso máximo ni de forma que generen un retraso en otro camino al que pertenezcan. Tampoco se le adjudicará un retraso a toda actividad $i \in N$ con $s_i(p) = 0$.

Distribuiremos la holgura de cada camino empleando inducción hacia atrás y reevaluándola a cada paso. Así, empezamos por la actividad final de cada camino y continuamos por su predecesora inmediata hasta llegar al inicio del camino o hasta que el camino tenga una holgura igual a cero.

```
#hC son las holguras de los caminos
#Td es el retraso total del proyecto
  actCriticas=unique(C[which(hC == 0),])
  if(Td == 0){
    camHolgura=which(hC > 0) #Seleccionamos los caminos con una holgura mayor que cero
    for(i in length(camHolgura)){ #Iteramos tantas veces como el numero de caminos con holgura
        mayor que cero
      h=min(hC[which(hC > 0)]) #Tomamos la hogura minima de los caminos con holgura mayor que cero
      cam=which(hC==h)[1] #Escogemos uno de los caminos con la holgura minima
      for(j in rev(C[cam,])){ #Recorremos el camino empezando por el final
        #Escogemos las actividades no criticas con s>0 y les sumamos el retraso extra
        if(length(which(actCriticas==j))==0 && s[j] != 0){
          dExtra=min(h,D[j]-d[j]) #Calculamos el retraso extra para la actividad j
          h=h-dExtra #Recalculamos la holgura del camino
          d[j]=d[j]+dExtra #Sumamos el retraso extra al que ya tenia la actividad
```

```
        }
      }
      #Actualizamos las holguras de los caminos
      tr<-calculo_tr(n,d,tC,C)
      hC=max(tr)-tr #holguras de los caminos
    }
  }
```

Con estos cálculos ya tenemos el NE óptimo del subjuego de $\Gamma(G, c, r)$ generado por la etapa 2. Por lo que, únicamente nos quedarían por calcular las utilidades, tanto del planificador como de las empresas que genera el NE optimista.

```
calculo_utilidad_planificador_modelo1<-function(Td,d,p,cost){
  #Utilidad del planificador, dependiente de si hay retraso o no
  if (Td==0){
    return(0) #si no hay retraso del proyecto
  }else{
    return(sum(d*p)-cost*Td)
  }
}

calculo_utilidades_empresas_modelo1<-function(n,Td,r,d,p){
  #Utilidades de los jugadores, dependientes de si hay retraso o no
  u=c()
  for (i in 1:n){
    if (Td==0){
      u[i]=r[i]*d[i] #si no hay retraso del proyecto
    }else{
      u[i]=(r[i]-p)*d[i]
    }
  }
  return(u)
}
```

Devolvemos con la función *metodo1* los retrasos de las actividades correspondientes a jugar el NE óptimo con la penalización dada y el retraso del proyecto, la utilidad del planificador y las utilidades de las actividades generadas.

Continuamos con los cálculos en la función *JuegoPERT*. Para obtener la penalización mínima que garantiza que el proyecto no se retrasará, recorremos todos los valores que se encuentran entre el mínimo del vector r y el máximo. Calculando con la función *metodo1* el retraso del proyecto detenemos las iteraciones en el momento en que el retraso del proyecto es igual a 0. Restringimos los posibles valores de la penalización entre el mínimo y el máximo del vector r ya que de ser p mayor que el máximo r_i, con $i \in N$, ninguna actividad se retrasaría. Igualmente, de ser p menor que el menor r_i, con $i \in N$, se retrasarían todas las actividades.

```
calculo_pGamma<-function(r,D,C,tC,cost){
  #Calculamos pGamma sabiendo que se encuentra entre los valores minimo y maximo del vector r
  for (i in seq(min(r),max(r),by=0.01)) {
    retraso=modelo1(r,D,C,tC,i,cost)$retrasos$retrasoProyecto
    if (retraso == 0) {
      return(i)
    }
  }
}
```

Por último, calculamos la penalización que garantiza la máxima utilidad al planificador. Para ello hacemos uso de la Proposición 2.7 y construimos un vector que contenga las utilidades para el planificador, devueltas por la función *metodo1* al pasar como argumentos la penalización mínima que garantiza que no se retrasa el proyecto y las diferentes r_i, con $i \in N$. Notemos que no debemos evaluar estas penalizaciones, sino que es necesario restar por un valor ϵ cercano a cero, en nuestro caso optamos

por 0.01 (puesto que supondría la menor unidad monetaria si trabajásemos en euros).

```
calculo_pmax_modelo1<-function(r,D,C,tC,cost,pGamma){
  n=length(r)
  #Calculamos pmax sabiendo que esta en el conjunto formado por los valores del vetor r y pGamma
  u0=c()
  u0[1]=modelo1(r,D,C,tC,pGamma-0.01,cost)$utilidades$planificador
  for (i in 1:n) {
    u0[i+1]=modelo1(r,D,C,tC,r[i]-0.01,cost)$utilidades$planificador
  }
  if (which(u0 == max(u0))[1] != 1) {
    return(min(r[which(u0 == max(u0))-1])) #Restamos 1 ya que el primer elemento del vector u0 no
        se
                                           #corresponde con una actividad, si no con pGamma
  } else {
    return(pGamma)
  }
}
```

Caso en el que toda actividad retrasada es penalizada

La estructura del código para el juego $\Delta(G, c, r)$ será muy similar a la ya vista para el juego $\Gamma(G, c, r)$ en el que solo se penalizan las actividades retrasadas si el proyecto también resulta retrasado. Como en el caso anterior, nos apoyamos en una función auxiliar para calcular el único NE del subjuego generado por la etapa 2.

```
modelo2<-function(r,D,C,tC,p,cost) {...}
```

La función auxiliar *modelo2* resultará más simple que la función análoga *modelo1*, pues en este caso el cálculo de las estrategias de los jugadores, i.e. los retrasos de las actividades, se realizará directamente con la comparación de los diferentes r_i, con $i \in N$, con la penalización dada. Para cada actividad i, si r_i es mayor que la penalización p, asignamos a la actividad i un retraso d_i igual a su máximo retraso permitido D_i. Por el contrario, si $r_i \leq p$, el retraso de la actividad i será 0.

```
calculo_d_modelo2<-function(n,p,r,D){
  #Calculo de los retrasos. Dependen solo de si r>p
  d=c() #vector de los retrasos de las actividades
  for (i in 1:n) {
    if (r[i] > p) { #si la ganancia por el retraso es mayor que la penalizacion la actividad se
        retrasa
      d[i]=D[i]
    } else { #si no es mayor la ganancia, la actividad no se retrasa
      d[i]=0
    }
  }
  return(d)
}
```

Una vez obtenidos los retrasos correspondientes al único NE del subjuego de $\Delta(G, c, r)$ generado por la etapa 2 resulta sencillo, haciendo uso de la matriz de caminos C, saber qué retraso, de existir, generarán las actividades con un $r_i > p$ en el proyecto.

```
#tr es un vector con los tiempos de los caminos sumando los retrasos de cada actividad
  Tr=max(tr) #Tiempo del proyecto con el modelo 2 y la penalizacion dada p.
  Td=Tr-max(tC)  #Retraso total del proyecto
```

Por último, calculamos las utilidades para el planificador y las empresas derivadas de sus respectivos retrasos.

```
  u0=sum(d*p)-cost*Td  #Utilidad para el planificador
  #Calculo de las utilidades de las empresas
  u=c()
  for (i in 1:n) {
    u[i]=(r[i]-p)*d[i] #Utilidad para las empresas
  }
```

La función *modelo2* devuelve los retrasos y las utilidades de las actividades y del proyecto para la penalización que hemos introducido como argumento en la función principal *JuegoPERT*. Donde continuamos con los cálculos de la penalización mínima que garantiza que el proyecto no se retrase y de la penalización óptima para el planificador.

Siguiendo la Proposición 2.8, calculamos la penalización mínima que garantiza que el proyecto no se retrasa, evaluando la función *modelo2* pasando como argumentos penalizaciones iguales a todos los elementos del vector r de menor a mayor. En el momento en el que el retraso del proyecto devuelto por la función *modelo2* sea igual a 0, interrumpimos las iteraciones y tomamos como $p\Delta$ el elemento del vector r correspondiente.

```
calculo_pDelta<-function(r,D,C,tC,cost){
  #Calculamos pDelta sabiendo que se encuentra entre los valores del vector r
  for (i in unique(r[order(r)])) {  #Recorremos los valores de r de menor a mayor
    retraso=modelo2(r,D,C,tC,i,cost)$retrasos$retrasoProyecto
    if (retraso == 0) {
      return(i)
    }
  }
}
```

Por último, calculamos la penalización que maximiza la utilidad del planificador. Para ello nos fijamos en la Proposición 2.9, que restringe los posibles valores de dicha penalización a los diferentes r_i, con $i \in N$. Como en el juego anterior, tomamos los valores r_i, con $i \in N$, restando por una pequeña cantidad, concretamente 0.01, y evaluamos en la función *modelo2* quedándonos con las diferentes utilidades devueltas. Finalmente, escogemos la mayor de dichas utilidades y asignamos el valor de la penalización que la generó como la penalización óptima.

```
calculo_pmax_modelo2<-function(r,D,C,tC,cost,pGamma){
  n=length(r)
  #Calculamos pmax sabiendo que esta en el conjunto formado por los valores del vetor r
  u0=c()
  for (i in 1:n) {
    u0[i]=modelo2(r,D,C,tC,r[i]-0.01,cost)$utilidades$planificador
  }
  return(min(r[which(u0 == max(u0))]))
}
```

En el Apéndice D podemos revisar el código completo de las diferentes funciones utilizadas en esta sección.

3.2.3. Resultados

Continuando con lo hecho hasta el momento, la función *JuegoPERT* devolverá un objeto *list* con toda la información calculada. Este objeto de resultados, tendrá los siguientes elementos:

- Los resultados relativos al juego, $\Gamma(G, c, r)$ o $\Delta(G, c, r)$ según corresponda, para la penalización dada inicialmente como argumento. Construiremos un objeto list que llamaremos *resultadosPenalizacionDada* y que contenga esta información con la siguiente estructura:

  - La penalización con la que se realizan los cálculos.
  - Un objeto *list* con la información de utilidades que contenga el valor de la utilidad para el planificador y un vector con las utilidades de las empresas.
  - Un objeto *list* con la información de retrasos que contenga el vector de retrasos d de las actividades al jugar el NE correspondiente y el retraso generado en el proyecto.

- La penalización mínima que garantiza que el proyecto no se retrasa, que denominaremos *pMinima*.
- La penalización óptima para el planificador, que denominaremos *pOptima*.
- Un objeto llamado *modeloPERT* con los resultados del modelo PERT/CPM obtenidos con la función *CalculoPERT* pero sin incluir los datos de entrada.
- Los datos de entrada necesarios para realizar todos los cálculos de la función *JuegoPERT*.

En el Apéndice E incluimos los archivos *JSON* generados a partir de los datos del Ejemplo 2.3 que incluimos en el Apéndice A.

3.3. API

Creamos un script que funcione a modo de interfaz de programación de aplicaciones (API) y que nos permita llamar a las diferentes funciones, cargar diferentes archivos de datos, etc. Como estamos utilizando scripts diferentes para las funciones, las cargamos mediante la función source del paquete básico de R.

```
source("../../TFM/programacionR/CalculoPERT.R")
```

Además, añadimos cuatro funciones en cada uno de los scripts que contienen las funciones *CalculoPERT* y *JuegoPERT*. Nos permitirán ejecutar la función principal del script de diversos modos, según queramos leer los datos de un archivo o desde el propio R, y generar un archivo con los resultados o conservarlos en R.

En el Apéndice F adjuntamos esta API sencilla que facilita el uso de las funciones para el usuario.

Capítulo 4

Propuestas para futuros estudios

A lo largo de los capítulos previos hemos estudiado el método PERT/CPM así como diversos juegos cooperativos y no cooperativos asociados. Nos hemos centrado principalmente en los juegos $\Gamma(G, c, r)$ y $\Delta(G, c, r)$, definidos como los juegos no cooperativos asociados a problemas PERT/CPM, diferenciando los casos en que las penalizaciones se apliquen solo cuando el proyecto es retrasado o se apliquen ante toda actividad retrasada respectivamente, suponiendo una penalización lineal fijada por el planificador en una primera etapa.

A continuación, vamos a ver diferentes generalizaciones que surgen de forma natural a partir de los juegos $\Gamma(G, c, r)$ y $\Delta(G, c, r)$. Nos fijaremos, en primer lugar, en los costes para el planificador asociados al retraso del proyecto y cómo resultaría conveniente estudiar unas penalizaciones consecuentes. Después, propondremos una generalización de la identificación entre las empresas y las actividades que componen el proyecto que se realiza en Bergantiños y Lorenzo (2017). Por último, nos basaremos en Estévez-Fernández (2012) para proponer una generalización sobre las penalizaciones en la que se incluyan también incentivos por finalizar antes de tiempo.

Futuros estudios sobre el coste para el planificador y diferentes penalizaciones

A pesar de que la motivación inicial del planteamiento y modelización de ambos juegos viene dada por las penalizaciones establecidas por ley para los contratos públicos en España, podemos esperar que los contratos privados, en general, no se atengan a los mismos criterios y necesidades.

Por ejemplo, si pensamos en la construcción de un nuevo estadio de fútbol por parte de un equipo. Podemos pensar en diferentes penalizaciones, dependiendo de las necesidades del propio equipo. Si la fecha de finalización es a principios de verano, es probable que la penalización por retrasar un mes el fin de las actividades no sea muy alta, pues en verano el uso del nuevo estadio no sería muy alto. Pero cuando empiece la liga, la necesidad de un nuevo estadio será mucho más alta ya que, de no estar disponible, el equipo incurrirá en otros gastos como podría ser el alquiler de un estadio a otro equipo de la ciudad. Esto generaría un coste no lineal en el planificador, que al ser una entidad privada, podríamos esperar que impusiese en consecuencia unas penalizaciones no lineales a las empresas.

Resultaría interesante, por ello, estudiar una generalización de las penalizaciones impuestas por el planificador, así como de los costes del retraso del proyecto.

Esta generalización de la función de penalización implicaría un cambio en los cálculos de los dos juegos $\Gamma(G, c, r)$ y $\Delta(G, c, r)$. Aumentaría especialmente la complejidad del cálculo del NE en el subjuego generado en la etapa 2 del juego $\Delta(G, c, r)$, pues las actividades que tuviesen incentivos para

retrasarse no tendrían porque tenerlos para hacerlo hasta el máximo permitido. De igual manera, en el subjuego generado por la etapa 2 del juego $\Gamma(G, c, r)$, se modificarían los cálculos de las utilidades de mínimo derecho para las actividades (y en consecuencia, las utilidades de mínimo derecho), lo que resultaría en un NE diferente.

Siguiendo con el mismo ejemplo del estadio. Los retrasos en las actividades finales del proyecto no supondrán un coste igual para el planificador. Es obvio que si una vez empezada la liga el césped aún no ha sido instalado, el equipo no podrá hacer uso del estadio y el coste generado será mayor. Pero si lo que falta para la finalización completa del proyecto es, por ejemplo, una de las gradas, seguiría habiendo un coste para el planificador (se dejarían de vender entradas para esa grada), aunque sería menor (el estadio podría usarse a pesar de no tener una grada).

En base a esto, podemos esperar que las penalizaciones del planificador puedan ser diferentes dependiendo de qué actividad sea la retrasada. En la modelización de un juego con estas características el planificador no impondría una única penalización, sino tantas como actividades finales tenga el proyecto.

Futuros estudios sobre las empresas y actividades

Los contratos de grandes proyectos urbanísticos, en España y en la mayoría de países, impulsados por la administración o por entidades privadas, suelen estar al alcance exclusivo de un pequeño grupo de empresas. Es probable, entonces, que varias actividades del proyecto puedan ser contratadas con una misma empresa.

Si una misma empresa puede realizar varias actividades de un mismo proyecto, sus conjuntos de información en cada punto serán mayores, pues podrá conocer lo que hará en el siguiente punto de decisión antes que el resto de empresas. Esto, supondría una generalización de los modelos expuestos por Bergantiños y Lorenzo (2017), pues asocian cada actividad con una única empresa.
se Podríamos encontrarnos, por ejemplo, con el caso simple de un camino del proyecto formado por dos actividades encargadas a la misma empresa. Supongamos que las penalizaciones solo son aplicadas cuando el proyecto es retrasa, que existe un NE sin retraso y que el camino planteado posee una holgura mayor que cero. Si la ganancia de la empresa por dedicar recursos a otras actividades es mayor para una de sus actividades que para la otra, podría decidir no retrasar una de las actividades y así poder retrasar al máximo la actividad que le reportará unos mayores beneficios, al no dedicar todos los recursos para su realización.

Notemos que esta generalización solo afectaría al caso en el que las actividades son penalizadas únicamente si el proyecto es retrasado. En el caso de que toda actividad retrasada fuese penalizada, el equilibrio de la etapa 2 del juego seguiría siendo el mismo, pues cada actividad se retrasará el máximo permitido o nada, dependiendo solamente de su ganancia al dedicar recursos a actividades no relacionadas con el proyecto.

Futuros estudios sobre incentivos

En el Capítulo 2 ya mencionamos que Estévez-Fernández et al. (2007) estudian también la perspectiva de un juego cooperativo en el que no se penalice a los jugadores que provocan un retraso del proyecto, sino que se premia a aquellos que finalizan antes del tiempo establecido. Posteriormente, en Estévez-Fernández (2012) se profundiza en esta idea y se amplía a juegos con penalizaciones e incentivos según se finalice después o antes del tiempo establecido.

Podemos tomar esta idea y aplicarla a los juegos no cooperativos estudiados en el Capítulo 3. Así convendría ampliar las variables que intervienen en el modelo, incluyendo un coste para cada empresa por dedicar más recursos a su actividad y un incentivo por la finalización antes del tiempo estipulado, que vendría fijado por el planificador en función del beneficio que este obtuviese con el adelanto del proyecto. También resulta natural definir, al igual que un tiempo máximo, un tiempo mínimo para la realización de cada actividad.

Sería adecuado, igual que en el caso de los juegos con penalizaciones exclusivamente, modelar dos juegos no cooperativos diferentes: uno en el que las actividades adelantadas solo serán premiadas si el proyecto finaliza antes de tiempo (igualmente, las actividades retrasadas solo serían penalizadas cuando el proyecto también es retrasado) y otro en el que toda actividad adelantada fuese premiada (igualmente, toda actividad retrasada sería penalizada).

Apéndice A

Ejemplos de lectura de datos

A continuación damos dos ejemplos para la construcción de los archivos *JSON* utilizados para introducir los datos en R. Concretamente, utilizamos los datos del Ejemplo 1.1 y del Ejemplo 2.3. Notemos, que con los datos de Ejemplo 1.1 no podemos ejecutar la función *JuegoPERT*, únicamente podremos calcular el método PERT/CPM con la función *CalculoPERT*.

- Datos del Ejemplo 1.1:

```
{"datosActividades":[
  {
    "actividad":1,
    "t_i":2
  },
  {
    "actividad":2,
    "t_i":4
  },
  {
    "actividad":3,
    "t_i":3
  },
  {
    "actividad":4,
    "t_i":2
  },
  {
    "actividad":5,
    "t_i":10
  },
  {
    "actividad":6,
    "t_i":4
  },
  {
    "actividad":7,
    "t_i":2
  },
  {
    "actividad":8,
    "t_i":1
  }],
"predecesores":[
  {
    "actividad":[1]
  },
  {
    "actividad":[2],
    "predecesores":[1]
  },
  {
    "actividad":[3],
    "predecesores":[2]
```

```
  },
  {
    "actividad":[4],
    "predecesores":[2]
  },
  {
    "actividad":[5],
    "predecesores":[3,4]
  },
  {
    "actividad":[6],
    "predecesores":[2]
  },
  {
    "actividad":[7],
    "predecesores":[6]
  },
  {
    "actividad":[8],
    "predecesores":[5,7]
  }]
}
```

- Datos del Ejemplo 2.3:

```
{"datosActividades":[
  {
    "actividad":1,
    "t_i":7,
    "D_i":4,
    "r_i":0
  },
  {
    "actividad":2,
    "t_i":4,
    "D_i":4,
    "r_i":5
  },
  {
    "actividad":3,
    "t_i":2,
    "D_i":2,
    "r_i":2
  },
  {
    "actividad":4,
    "t_i":2,
    "D_i":3,
    "r_i":5
  }],
"predecesores":[
  {
    "actividad":[1]
  },
  {
    "actividad":[2]
  },
  {
    "actividad":[3],
    "predecesores":[1,2]
  },
  {
    "actividad":[4],
    "predecesores":[2]
  }]
}
```

Apéndice B

Código para el modelo PERT/CPM

Presentamos en este apéndice el código necesario para los cálculos relativos al método PERT/CPM.

```
#### Funciones utilizadas en la API ####

CalculoPERT_API_read_write<-function(ficheroDatos,ficheroResultados){
  require(jsonlite)
  #Leemos el archivo
  datos<-fromJSON(ficheroDatos)
  #Llamamos a la funcion
  PERT<-CalculoPERT(datos)
  #Construimos el archivo de resultados
  exportJSON<-toJSON(PERT)
  write(exportJSON,ficheroResultados)
}

CalculoPERT_API_read<-function(ficheroDatos){
  require(jsonlite)
  #Leemos el archivo
  datos<-fromJSON(ficheroDatos)
  #Llamamos a la funcion
  return(CalculoPERT(datos))
}

CalculoPERT_API_write<-function(datos,ficheroResultados){
  require(jsonlite)
  #Llamamos a la funcion
  PERT<-CalculoPERT(datos)
  #Construimos el archivo de resultados
  exportJSON<-toJSON(PERT)
  write(exportJSON,ficheroResultados)
}

CalculoPERT_API<-function(datos){
  #Llamamos a la funcion
  return(CalculoPERT(datos))
}


#### Funcion principal ####

CalculoPERT<-function(datos){
  t<-datos$datosActividades$t_i
  Pre<-datos$predecesores$predecesores
  n=length(t)

  A<-matrizSucesoresInmediatos(Pre,n)

  b=which(colSums(A) == 0)  #Actividades iniciales
  e=which(rowSums(A) == 0)  #Actividades finales

  C<-matrizCaminos(A,b,n) #Matriz que tiene por cada fila un camino del proyecto
  TC<-matrizTiemposCaminos(C,t,n) #Matriz de iguales dimensiones que C pero con los tiempos de cada
        actividad
```

```
  m=nrow(C)  #numero de caminos
  TP=max(rowSums(TC))  #tiempo del proyecto
  HC=TP-rowSums(TC)  #holguras de los caminos

  #Calculamos ahora el minimo tiempo de inicio (si las actividades precedentes no se retrasan) para
       cada
  #actividad y el maximo tiempo en el que pueden finalizar sin causar retraso en el proyecto
  tInicial=c() #tiempo inicial minimo para cada actividad
  tFinal=c() #tiempo final maximo para cada actividad
  Ha=c() #holgura de cada actividad
  for (i in 1:n) {
    tInicialAux=rep(0,m) #Vector auxiliar de los tiempo iniciales
    tFinalAux=rep(0,m) #Vector auxiliar de los tiempos finales
    f=which(C == i)%%m #Vemos los caminos que contienen la actividad i
    f[f==0]=m #Como hacemos el modulo, si i esta en el camino m nos devolvera 0, por eso lo
         arreglamos
    if (length(which(b == i)) != 0) { #Vemos si i es una actividad inicial
      tInicial[i]=0 #Obviamente el tiempo inicial sera 0
      for (j in f) {
        tFinalAux[j]=sum(TC[j,(1+which(C[j,] == i)):length(C[j,])])
      }
      tFinal[i]=TP-max(tFinalAux)
    } else { if (length(which(e == i)) != 0) { #Vemos si es actividad final
      for (j in f) {
        tInicialAux[j]=sum(TC[j,1:(which(C[j,] == i)-1)])
      }
      tInicial[i]=max(tInicialAux)
      tFinal[i]=TP #El tiempo final sera el tiempo total del proyecto
    } else { #Si i no es actividad inicial ni final
      for (j in f) {
        tInicialAux[j]=sum(TC[j,1:(which(C[j,] == i)-1)]) #Sumamos los tiempos de las actividades
             previas de i en el camino j
        tFinalAux[j]=sum(TC[j,(1+which(C[j,] == i)):length(C[j,])]) #Sumamos los tiempos de las
             actividades sucesoras de i en el camino j
      }
      tInicial[i]=max(tInicialAux)
      tFinal[i]=TP-max(tFinalAux)
    }}
    if (length(f) == 1) {
      Ha[i]=TP-sum(TC[f,])  #Holgura de la actividad si solo esta en un camino (holgura del camino)
    } else {
      Ha[i]=TP-max(rowSums(TC[f,])) #Si pertenece a varios caminos, tomamos el de mayor tiempo
    }
  }
  return(construirResultadosPERT(t,tInicial,tFinal,Ha,C,TC,HC,TP,datos))
}


#### Funciones auxiliares ####

matrizSucesoresInmediatos<-function(Pre,n){
  A=matrix(0,nrow=n,ncol=n) #futura matriz de los sucesores inmediatos de cada actividad (matriz de
       ceros y unos)
  for (i in 1:n){
    A[Pre[[i]],i]=1 #en cada columna ponemos un 1 en la fila que se corresponda a una actividad
         predecesora
  }
  return(A)
}

matrizCaminos<-function(A,b,n){
  B=matrix(0,nrow=length(b),ncol=n)  #Futura matriz de los caminos intermedios que comienzan en las
       actividades iniciales
  B[1:length(b),1]=b
  C=matrix(0,nrow=1,ncol=n) #Futura matriz de los caminos

  for (i in 2:n) {  #Posicion i-esima del camino
    for (j in which(B[,i-1] != 0)) {  #j-esima fila de la matriz
      if (sum(A[B[j,i-1],]) == 0){ #Primero comprobamos si la actividad inicial es tambien final
        c=B[j,]
        C=rbind(C,c) #Agregamos el camino que solo contiene la actividad
        break
      }
      for (k in seq(1,n,by=1)[-b]) {  #Vemos las actividades que debemos agregar
        if (A[B[j,i-1],k] == 1 && sum(A[k,]) != 0) { #Comprobamos si la actividad k pertenece al
             camino y es intermedia (no final)
```

```
          l=B[j,]
          l[i]=k
          B=rbind(B,l) #Agregamos la actividad a la matriz de caminos intermedios
        } else { if (A[B[j,i-1],k] == 1 && sum(A[k,]) == 0) { #Comprobamos si la actividad k
            pertenece al camino y es final
          c=B[j,]
          c[i]=k
          C=rbind(C,c) #Agregamos el camino c a la matriz de caminos
        }}
      }
    }
  }
  return(C[2:length(C[,1]),which(colSums(C) != 0)]) #cada fila es un camino desde un nodo inicial a
       un nodo final
}

matrizTiemposCaminos<-function(C,t,n){
  taux=c() #vector auxiliar de los tiempos (incluimos un cero al inicio para los elementos que no
      tienen actividades)
  for (i in 1:n) {
    taux[i+1]=t[i]
  }
  taux[1]=0
  if(nrow(C) != 0){ #si la matriz de caminos tiene mas de una fila (hay mas de un camino)
    TC=matrix(taux[C+1],nrow=nrow(C)) #matriz de los tiempos de las actividades de los caminos
  } else {
    TC=taux[C+1] #tiempos de las actividades que forman el camino del camino
  }
  return(TC)
}

construirResultadosPERT<-function(t,tInicial,tFinal,Ha,C,TC,HC,TP,datos){
  #Construimos el objeto que contendra los resultados
  actividad=seq(1,length(t),by=1)
  tiemposActividades=data.frame(Actividad=actividad,Duracion=t,TiempoInicialMin=tInicial,
                                TiempoFinalMax=tFinal,Holgura=Ha)

  Caminos=list()
  Caminos$matrizCaminos<-C
  Caminos$tiempoActividadesCaminos<-TC
  Caminos$tiempoCaminos<-rowSums(TC)
  Caminos$holgurasCaminos<-HC
  Caminos$numeroCaminos<-nrow(C)

  PERT=list()
  PERT$Actividades<-tiemposActividades
  PERT$Caminos<-Caminos
  PERT$Proyecto$tiempoProyecto<-TP
  PERT$datosEntrada$datosActividades$duraciones<-datos$datosActividades$t_i
  PERT$datosEntrada$datosPredecesores<-datos$predecesores
  return(PERT)
}
```

Apéndice C

Resultados del método PERT/CPM

Presentamos en este apéndice los dos archivos *JSON* resultantes de ejecutar la función *CalculoPert* con los datos del Ejemplo 1.1 y del Ejemplo 2.3.

- Resultados con los datos del Ejemplo 1.1:


```
{"Actividades":[
  {
    "Actividad":1,
    "Duracion":2,
    "TiempoInicialMin":0,
    "TiempoFinalMax":2,
    "Holgura":0
  },
  {
    "Actividad":2,
    "Duracion":4,
    "TiempoInicialMin":2,
    "TiempoFinalMax":6,
    "Holgura":0
  },
  {
    "Actividad":3,
    "Duracion":3,
    "TiempoInicialMin":6,
    "TiempoFinalMax":9,
    "Holgura":0
  },
  {
    "Actividad":4,
    "Duracion":2,
    "TiempoInicialMin":6,
    "TiempoFinalMax":9,
    "Holgura":1
  },
  {
    "Actividad":5,
    "Duracion":10,
    "TiempoInicialMin":9,
    "TiempoFinalMax":19,
    "Holgura":0
  },
  {
    "Actividad":6,
    "Duracion":4,
    "TiempoInicialMin":6,
    "TiempoFinalMax":17,
    "Holgura":7
  },
  {
    "Actividad":7,
    "Duracion":2,
```

```
    "TiempoInicialMin":10,
    "TiempoFinalMax":19,
    "Holgura":7
  },
  {
    "Actividad":8,
    "Duracion":1,
    "TiempoInicialMin":19,
    "TiempoFinalMax":20,
    "Holgura":0
  }],
"Caminos":
  {
    "matrizCaminos":[[1,2,3,5,8],[1,2,4,5,8],[1,2,6,7,8]],
    "tiempoActividadesCaminos":[[2,4,3,10,1],[2,4,2,10,1],[2,4,4,2,1]],
    "tiempoCaminos":[20,19,13],
    "holgurasCaminos":[0,1,7],
    "numeroCaminos":[3]
  },
"Proyecto":
  {
    "tiempoProyecto":[20]
  },
"datosEntrada":
  {
    "datosActividades":
      {
        "duraciones":[2,4,3,2,10,4,2,1]
      },
    "datosPredecesores":[
      {
        "actividad":[1],
        "predecesores":{}
      },
      {
        "actividad":[2],
        "predecesores":[1]
      },
      {
        "actividad":[3],
        "predecesores":[2]
      },
      {
        "actividad":[4],
        "predecesores":[2]
      },
      {
        "actividad":[5],
        "predecesores":[3,4]
      },
      {
        "actividad":[6],
        "predecesores":[2]
      },
      {
        "actividad":[7],
        "predecesores":[6]
      },
      {
        "actividad":[8],
        "predecesores":[5,7]
      }
    ]
  }
}
```


- Resultados con los datos del Ejemplo 2.3:


```
{"Actividades":[
  {
    "Actividad":1,
    "Duracion":7,
    "TiempoInicialMin":0,
    "TiempoFinalMax":7,
```

```
    "Holgura":0
  },
  {
    "Actividad":2,
    "Duracion":4,
    "TiempoInicialMin":0,
    "TiempoFinalMax":7,
    "Holgura":3
  },
  {
    "Actividad":3,
    "Duracion":2,
    "TiempoInicialMin":7,
    "TiempoFinalMax":9,
    "Holgura":0
  },
  {
    "Actividad":4,
    "Duracion":2,
    "TiempoInicialMin":4,
    "TiempoFinalMax":9,
    "Holgura":3
  }],
"Caminos":
  {
    "matrizCaminos":[[1,3],[2,3],[2,4]],
    "tiempoActividadesCaminos":[[7,2],[4,2],[4,2]],
    "tiempoCaminos":[9,6,6],
    "holgurasCaminos":[0,3,3],
    "numeroCaminos":[3]
  },
"Proyecto":
  {
    "tiempoProyecto":[9]
  },
"datosEntrada":
  {
    "datosActividades":
      {
        "duraciones":[7,4,2,2]
      },
    "datosPredecesores":[
      {
        "actividad":[1],
        "predecesores":{}
      },
      {
        "actividad":[2],
        "predecesores":{}
      },
      {
        "actividad":[3],
        "predecesores":[1,2]
      },
      {
        "actividad":[4],
        "predecesores":[2]
      }
    ]
  }
}
```

Apéndice D

Código para los juegos no cooperativos asociados a problemas PERT/CPM

Presentamos en este apéndice el código necesario para los cálculos de las estrategias óptimas de las empresas en los juegos $\Gamma(G, c, r)$ y $\Delta(G, c, r)$, así como las penalizaciones óptimas para el planificador y las penalizaciones mínimas que garantizan que el proyecto no se retrasa.

```
#### Funciones utilizadas en la API ####

JuegoPERT_API_read_write<-function(ficheroDatos,ficheroResultados,p,cost,
    penalizarCualquierActividadRetrasada){
  require(jsonlite)
  #Leemos el archivo
  datos<-fromJSON(ficheroDatos)
  #Llamamos a la funcion
  resultadosJuego<-JuegoPERT(datos,p,cost,penalizarCualquierActividadRetrasada)
  #Construimos el archivo de resultados
  exportJSON<-toJSON(resultadosJuego)
  write(exportJSON,ficheroResultados)
}

JuegoPERT_API_read<-function(ficheroDatos,p,cost,penalizarCualquierActividadRetrasada){
  require(jsonlite)
  #Leemos el archivo
  datos<-fromJSON(ficheroDatos)
  #Llamamos a la funcion
  return(JuegoPERT(datos,p,cost,penalizarCualquierActividadRetrasada))
}

JuegoPERT_API_write<-function(datos,ficheroResultados,p,cost,penalizarCualquierActividadRetrasada){
  require(jsonlite)
  #Llamamos a la funcion
  resultadosJuego<-JuegoPERT(datos,p,cost,penalizarCualquierActividadRetrasada)
  #Construimos el archivo de resultados
  exportJSON<-toJSON(resultadosJuego)
  write(exportJSON,ficheroResultados)
}

JuegoPERT_API<-function(datos,p,cost,penalizarCualquierActividadRetrasada){
  #Llamamos a la funcion
  return(JuegoPERT(datos,p,cost,penalizarCualquierActividadRetrasada))
}


#### Funcion principal ####

JuegoPERT<-function(datos,p,cost,penalizarCualquierActividadRetrasada) {
```

```
  #Lectura de datos y llamada a la funcion CalculoPERT
  t<-datos$datosActividades$t_i #tiempos de las actividades
  r<-datos$datosActividades$r_i #ganancias de las empresas al desviar recursos a terceras
      actividades
  D<-datos$datosActividades$D_i #maximo retraso posible de cada actividad
  PERT<-CalculoPERT(datos) #Calculamos el modelo PERT
  C<-PERT$Caminos$matrizCaminos #matriz de caminos
  tC<-PERT$Caminos$tiempoCaminos #tiempo de cada camino
  n=length(t) #numero de actividades

  if (penalizarCualquierActividadRetrasada == FALSE) {
    #Obtenemos el resultado generado por el NE optimista para el modelo con la penalizacion dada
    modelo1p=modelo1(r,D,C,tC,p,cost)
    #Calculo de la penalizacion minima que garantiza que el proyecto no se retrasa
    pGamma<-calculo_pGamma(r,D,C,tC,cost)
    #Calculo de la penalizacion que maximiza la utilidad del planificador
    pmax<-calculo_pmax_modelo1(r,D,C,tC,cost,pGamma)
    return(construirResultados_JuegoPert(modelo1p,pGamma,pmax,PERT,datos,p,cost))
  }
  if (penalizarCualquierActividadRetrasada == TRUE) {
    #Obtenemos el resultado para el modelo con la penalizacion dada
    modelo2p=modelo2(r,D,C,tC,p,cost)
    #Calculo de la penalizacion minima que garantiza que el proyecto no se retrasa
    pDelta<-calculo_pDelta(r,D,C,tC,cost)
    #Calculo de la penalizacion que maximiza la utilidad del planificador
    pmax<-calculo_pmax_modelo2(r,D,C,tC,cost,pGamma)
    return(construirResultados_JuegoPert(modelo2p,pDelta,pmax,PERT,datos,p,cost))
  }
}

calculo_pGamma<-function(r,D,C,tC,cost){
  #Calculamos pGamma sabiendo que se encuentra entre los valores minimo y maximo del vector r
  for (i in seq(min(r),max(r),by=0.01)) {
    retraso=modelo1(r,D,C,tC,i,cost)$retrasos$retrasoProyecto
    if (retraso == 0) {
      return(i)
    }
  }
}

calculo_pmax_modelo1<-function(r,D,C,tC,cost,pGamma){
  n=length(r)
  #Calculamos pmax sabiendo que esta en el conjunto formado por los valores del vetor r y pGamma
  u0=c()
  u0[1]=modelo1(r,D,C,tC,pGamma-0.01,cost)$utilidades$planificador
  for (i in 1:n) {
    u0[i+1]=modelo1(r,D,C,tC,r[i]-0.01,cost)$utilidades$planificador
  }
  if (which(u0 == max(u0))[1] != 1) {
    return(min(r[which(u0 == max(u0))-1])) #Restamos 1 ya que el primer elemento del vector u0 no
        se corresponde con una actividad, si no con pGamma
  } else {
    return(pGamma)
  }
}

calculo_pDelta<-function(r,D,C,tC,cost){
  #Calculamos pDelta sabiendo que se encuentra entre los valores del vector r
  for (i in unique(r[order(r)])) {  #Recorremos los valores de r de menor a mayor
    retraso=modelo2(r,D,C,tC,i,cost)$retrasos$retrasoProyecto
    if (retraso == 0) {
      return(i)
    }
  }
}

calculo_pmax_modelo2<-function(r,D,C,tC,cost,pGamma){
  n=length(r)
  #Calculamos pmax sabiendo que esta en el conjunto formado por los valores del vetor r
  u0=c()
  for (i in 1:n) {
    u0[i]=modelo2(r,D,C,tC,r[i]-0.01,cost)$utilidades$planificador
  }
  return(min(r[which(u0 == max(u0))]))
}

construirResultados_JuegoPert<-function(modelop,pMinima,pOptima,PERT,datos,p,cost){
```

```
  #Construimos el objeto que devolvemos en la funcion JuegoPERT
  resultadosJuego=list()
  resultadosJuego$resultadosPenalizacionDada<-modelop
  resultadosJuego$pMinima<-pMinima
  resultadosJuego$pOptima<-pOptima
  resultadosJuego$modeloPERT$Actividades<-PERT$Actividades
  resultadosJuego$modeloPERT$Caminos<-PERT$Caminos
  resultadosJuego$modeloPERT$Proyecto$tiempoProyecto<-PERT$Proyecto$tiempoProyecto
  resultadosJuego$datosEntrada$datosActividades<-datos$datosActividades
  resultadosJuego$datosEntrada$datosPredecesores<-datos$predecesores
  resultadosJuego$datosEntrada$penalizacion<-p
  resultadosJuego$datosEntrada$costePlanificador<-cost
  return(resultadosJuego)
}


#### Modelo con penalizaciones solo si se retrasa el proyecto ####

modelo1<-function(r,D,C,tC,p,cost){
  n=length(D) #Numero de actividades

  s<-calculo_s(n,p,r,D) #Vector de retrasos de minimo derecho para cada actividad
  sC<-calculo_matriz_sC(n,s,C) #Matriz auxiliar de los retrasos de minimo derecho por camino
  ts=tC+rowSums(sC)  #Tiempos de los caminos sumando los retrasos de minimo derecho

  #Calculamos los retrasos para cada actividad
  d<-calculo_d_modelo1(n,p,r,D,s,ts,tC)
  #Calculamos los tiempos de los caminos sumando los retrasos
  tr<-calculo_tr(n,d,tC,C)

  hC=max(tr)-tr #Holguras de los caminos
  Tr=max(tr) #Tiempo del proyecto con el modelo 1 y la penalizacion dada p.
  Td=Tr-max(tC) #Retraso total del proyecto

  #Repartimos las holguras sobrantes en cada camino empezando por las actividades finales
  actCriticas=unique(C[which(hC == 0),])
  if(Td == 0){
    camHolgura=which(hC > 0) #Seleccionamos los caminos con una holgura mayor que cero
    for(i in length(camHolgura)){ #Iteramos tantas veces como el numero de caminos con holgura
        mayor que cero
      h=min(hC[which(hC > 0)]) #Tomamos la hogura minima de los caminos con holgura mayor que cero
      cam=which(hC==h)[1] #Escogemos uno de los caminos con la holgura minima
      for(j in rev(C[cam,])){ #Recorremos el camino empezando por el final
        #Escogemos las actividades no criticas con s>0 y les sumamos el retraso extra
        if(length(which(actCriticas==j))==0 && s[j] != 0){
          dExtra=min(h,D[j]-d[j]) #Calculamos el retraso extra para la actividad j
          h=h-dExtra #Recalculamos la holgura del camino
          d[j]=d[j]+dExtra #Sumamos el retraso extra al que ya tenia la actividad
        }
      }
      #Actualizamos las holguras de los caminos
      tr<-calculo_tr(n,d,tC,C)
      hC=max(tr)-tr #holguras de los caminos
    }
  }

  #Calculamos las utilidades
  u0<-calculo_utilidad_planificador_modelo1(Td,d,p,cost)
  u<-calculo_utilidades_empresas_modelo1(n,Td,r,d,p)

  return(construirResultados_modelo(u0,u,Td,d))
}

calculo_s<-function(n,p,r,D){
  #Calculamos los retrasos de minimo derecho para cada actividad
  s=c()
  for(i in 1:n){
    if (r[i]>p && r[i]!=0){
      s[i]=((r[i]-p)*D[i])/r[i]
    }else{
      s[i]=0
    }
  }
  return(s)
}

calculo_matriz_sC<-function(n,s,C){
```

```
  saux=c() #vector auxiliar del vector de retrasos de minimo derecho
  for(i in 1:n){
    saux[i+1]=s[i]
  }
  saux[1]=0
  #Devolvemos la matriz auxiliar de los retrasos de minimo derecho por camino
  return(matrix(saux[C+1],nrow=nrow(C)))
}

calculo_d_modelo1<-function(n,p,r,D,s,ts,tC){
  #Calculamos los retrasos de las actividades según hay o no retraso del proyecto
  d=c()
  for(i in 1:n){
    if(max(ts) > max(tC) && r[i] > p){ #vemos si habra retraso
      d[i]=D[i]
    }else{
      d[i]=s[i]
    }
  }
  return(d)
}

calculo_utilidad_planificador_modelo1<-function(Td,d,p,cost){
  #Utilidad del planificador, dependiente de si hay retraso o no
  if (Td==0){
    return(0) #si no hay retraso del proyecto
  }else{
    return(sum(d*p)-cost*Td)
  }
}

calculo_utilidades_empresas_modelo1<-function(n,Td,r,d,p){
  #Utilidades de los jugadores, dependientes de si hay retraso o no
  u=c()
  for (i in 1:n){
    if (Td==0){
      u[i]=r[i]*d[i] #si no hay retraso del proyecto
    }else{
      u[i]=(r[i]-p)*d[i]
    }
  }
  return(u)
}



#### Modelo con penalizaciones sobre cualquier actividad retrasada ####

modelo2<-function(r,D,C,tC,p,cost) {
  n=length(D) #Numero de actividades

  #Calculamos los retrasos para cada actividad
  d<-calculo_d_modelo2(n,p,r,D)
  #Calculamos los tiempos de los caminos sumando los retrasos
  tr<-calculo_tr(n,d,tC,C)

  Tr=max(tr) #Tiempo del proyecto con el modelo 2 y la penalizacion dada p.
  Td=Tr-max(tC)  #Retraso total del proyecto

  #Calculamos las utilidades
  u0=sum(d*p)-cost*Td  #Utilidad para el planificador
  u<-calculo_utilidades_empresas_modelo2(n,r,d,p)

  return(construirResultados_modelo(u0,u,Td,d))
}

calculo_d_modelo2<-function(n,p,r,D){
  #Calculo de los retrasos. Dependen solo de si r>p
  d=c() #vector de los retrasos de las actividades
  for (i in 1:n) {
    if (r[i] > p) { #si la ganancia por el retraso es mayor que la penalizacion la actividad se
        retrasa
      d[i]=D[i]
    } else { #si no es mayor la ganancia, la actividad no se retrasa
      d[i]=0
    }
  }
```

```
  return(d)
}

calculo_utilidades_empresas_modelo2<-function(n,r,d,p){
  #Calculo de las utilidades de las empresas
  u=c()
  for (i in 1:n) {
    u[i]=(r[i]-p)*d[i] #Utilidad para las empresas
  }
  return(u)
}


#### Funciones comunes a ambos modelos ####

calculo_tr<-function(n,d,tC,C){
  #Calculo de los tiempos de los caminos con el retraso d
  daux=c() #vector auxiliar
  for(i in 1:n) {
    daux[i+1]=d[i]
  }
  daux[1]=0
  dC=matrix(daux[C+1],nrow=nrow(C)) #matriz auxiliar
  return(tC+rowSums(dC))  #tiempos de los caminos con los retrasos
}

construirResultados_modelo<-function(u0,u,Td,d){
  #Construimos el objeto que devolvemos en la funcion modelo1 o modelo2
  modelop=list()
  modelop$utilidades$planificador=u0
  modelop$utilidades$empresas=u
  modelop$retrasos$retrasoProyecto=Td
  modelop$retrasos$retrasoActividades=d
  return(modelop)
}
```

Apéndice E

Resultados para los juegos no cooperativos

Incluimos en este apéndice los dos archivos *JSON* generados con la función *JuegoPERT* al pasar como argumento los datos del Apéndice A correspondientes al Ejemplo 2.3. El primero de los archivos se corresponde con el juego en el que las penalizaciones solo se aplican cuando el proyecto se retrasa. El segundo de los archivos se corresponde con el juego en el que toda actividad que se retrase resulta penalizada. Al igual que en el Ejemplo 2.4 y en el Ejemplo 2.10, en los que se estudia la etapa 2 para ambos juegos, utilizamos una penalización de 4 y un coste para el planificador de 6

- Resultados con los datos del Ejemplo 2.4:


```
{"resultadosPenalizacionDada":
  {
    "utilidades":
      {
        "planificador":[0],
        "empresas":[0,4,0,11]
      },
    "retrasos":
      {
        "retrasoProyecto":[0],
        "retrasoActividades":[0,0.8,0,2.2]
      }
  },
"pMinima":[2.86],
"pOptima":[5],
"modeloPERT":
  {
    "Actividades":[
      {
        "Actividad":1,
        "Duracion":7,
        "TiempoInicialMin":0,
        "TiempoFinalMax":7,
        "Holgura":0
      },
      {
        "Actividad":2,
        "Duracion":4,
        "TiempoInicialMin":0,
        "TiempoFinalMax":7,
        "Holgura":3
      },
      {
        "Actividad":3,
        "Duracion":2,
        "TiempoInicialMin":7,
        "TiempoFinalMax":9,
```

```
        "Holgura":0
      },
      {
        "Actividad":4,
        "Duracion":2,
        "TiempoInicialMin":4,
        "TiempoFinalMax":9,
        "Holgura":3
      }],
    "Caminos":
      {
        "matrizCaminos":[[1,3],[2,3],[2,4]],
        "tiempoActividadesCaminos":[[7,2],[4,2],[4,2]],
        "tiempoCaminos":[9,6,6],
        "holgurasCaminos":[0,3,3],
        "numeroCaminos":[3]
      },
    "Proyecto":
      {
        "tiempoProyecto":[9]
      }
  },
"datosEntrada":
  {
    "datosActividades":[
      {
        "actividad":1,
        "t_i":7,
        "D_i":4,
        "r_i":0
      },
      {
        "actividad":2,
        "t_i":4,
        "D_i":4,
        "r_i":5
      },
      {
        "actividad":3,
        "t_i":2,
        "D_i":2,
        "r_i":2
      },
      {
        "actividad":4,
        "t_i":2,
        "D_i":3,
        "r_i":5
      }],
    "datosPredecesores":[
      {
        "actividad":[1],
        "predecesores":{}
      },
      {
        "actividad":[2],
        "predecesores":{}
      },
      {
        "actividad":[3],
        "predecesores":[1,2]
      },
      {
        "actividad":[4],
        "predecesores":[2]
      }],
    "penalizacion":[4],
    "costePlanificador":[6]
  }
}
```


- Resultados con los datos del Ejemplo 2.10:


```
{"resultadosPenalizacionDada":
```

```
  {
    "utilidades":
      {
        "planificador":[4],
        "empresas":[0,4,0,3]
      },
    "retrasos":
      {
        "retrasoProyecto":[4],
        "retrasoActividades":[0,4,0,3]
      }
  },
"pMinima":[5],
"pOptima":[5],
"modeloPERT":
  {
    "Actividades":[
      {
        "Actividad":1,
        "Duracion":7,
        "TiempoInicialMin":0,
        "TiempoFinalMax":7,
        "Holgura":0
      },
      {
        "Actividad":2,
        "Duracion":4,
        "TiempoInicialMin":0,
        "TiempoFinalMax":7,
        "Holgura":3
      },
      {
        "Actividad":3,
        "Duracion":2,
        "TiempoInicialMin":7,
        "TiempoFinalMax":9,
        "Holgura":0
      },
      {
        "Actividad":4,
        "Duracion":2,
        "TiempoInicialMin":4,
        "TiempoFinalMax":9,
        "Holgura":3
      }],
    "Caminos":
      {
        "matrizCaminos":[[1,3],[2,3],[2,4]],
        "tiempoActividadesCaminos":[[7,2],[4,2],[4,2]],
        "tiempoCaminos":[9,6,6],
        "holgurasCaminos":[0,3,3],
        "numeroCaminos":[3]
      },
    "Proyecto":
      {
        "tiempoProyecto":[9]
      }
  },
"datosEntrada":
  {
    "datosActividades":[
      {
        "actividad":1,
        "t_i":7,
        "D_i":4,
        "r_i":0
      },
      {
        "actividad":2,
        "t_i":4,
        "D_i":4,
        "r_i":5
      },
      {
        "actividad":3,
        "t_i":2,
        "D_i":2,
```

```
        "r_i":2
      },
      {
        "actividad":4,
        "t_i":2,
        "D_i":3,
        "r_i":5
      }],
    "datosPredecesores":[
      {
        "actividad":[1],
        "predecesores":{}
      },
      {
        "actividad":[2],
        "predecesores":{}
      },
      {
        "actividad":[3],
        "predecesores":[1,2]
      },
      {
        "actividad":[4],
        "predecesores":[2]
      }]
    "penalizacion":[4],
    "costePlanificador":[6]
  }
}
```

Apéndice F

Código para la API

En este apéndice, presentamos el código relativo al script en R que utilizaremos para controlar las diferentes funciones implementadas.

```
#### Primero cargamos las funciones ####
source("C:/Users/Alfredo/Desktop/TFM/Programación en R/CalculoPERT.R")
source("C:/Users/Alfredo/Desktop/TFM/Programación en R/JuegoPERT.R")


##### Proyecto PERT #####
#Ruta del archivo con los datos que introduciremos en la funcion
ficheroDatos<-"../../TFM/ProgramacionR/datos.json"
#Escogemos la ruta donde guardar el archivo resultante
ficheroResultadosPERT<-"../../TFM/ProgramacionR/resultadosMetodoPERT.json"

#Funcion si queremos generar un archivo json a partir de otro archivo json
CalculoPERT_API_read_write(ficheroDatos,ficheroResultadosPERT)
#Funcion si queremos guardar los datos en R a partir de un archivo json
resultadoMetodo<-CalculoPERT_API_read(ficheroDatos)
#Funcion si queremos generar un archivo json a partir de datos en R
CalculoPERT_API_write(datos,ficheroResultadosPERT)
#Funcion si queremos guardar los datos en R a partir de datos en R
resultadoMetodo<-CalculoPERT_API(datos)


##### Juego PERT #####
#Definimos y leemos los datos que introduciremos en la funcion
penalizacion=4; costePlanificador=6; penalizarCualquierActividadRetrasada=FALSE
ficheroDatos<-"../../TFM/ProgramacionR/datos.json"
#Escogemos la ruta donde guardar el archivo resultante
ficheroResultadosJuegoPERT<-"../../TFM/ProgramacionR/resultadosJuegoPERT.json"

#Funcion si queremos generar un archivo json a partir de un archivo json
JuegoPERT_API_read_write(ficheroDatos,ficheroResultadosJuegoPERT,penalizacion,costePlanificador,
    penalizarCualquierActividadRetrasada)
#Funcion si queremos guardar los datos en R a partir de un archivo json
resultadosJuego<-JuegoPERT_API_read(ficheroDatos,penalizacion,costePlanificador,
    penalizarCualquierActividadRetrasada)
#Funcion si queremos generar un archivo json a partir de datos en R
JuegoPERT_API_write(datos,ficheroResultadosJuegoPERT,penalizacion,costePlanificador,
    penalizarCualquierActividadRetrasada)
#Funcion si queremos guardar los datos en R a partir de datos en R
resultadosJuego<-JuegoPERT_API(datos,penalizacion,costePlanificador,
    penalizarCualquierActividadRetrasada)
```

Bibliografía

[1] Bergantiños G., Lorenzo L. (2017) How to apply penalties to avoid delays in projects.

[2] Bergantiños G.,Sánchez E. (2002) How to Distribute Costs Associated with a Delayed Project. Annals of Operations Research 109, 159-174.

[3] Branzei R., Ferrari G., Fragnelli V., Tijs S. (2002) Two Approaches to the Problem of Sharing Delay Costs in Joint Projects. Annals of Operations Research 109, 359-374.

[4] Estévez-Fernández A. (2012) A game theoretical approach to sharing penalties and rewards in projects. European Journal of Operational Research 216 647-657.

[5] Estévez-Fernández A., Borm P., Hamers H. (2007) Project games. Int J Game Theory 36:149-176.

[6] Félix F. (22 de abril de 2018) El retraso del Corredor Mediterráneo tiene un alto coste para España. ABC. Recuperado de https://www.abc.es/

[7] González M. (11 de febrero de 2018) El retraso del submarino S-80 cuesta 130 millones de euros. El País. Recuperado de https://www.elpais.com/

[8] Malcolm D. G., Roseboom J. H., Clark C. E., Fazar W. (1959) Application of a Technique for Research and Development Program Evaluation. Operations Research Vol. 7, No. 5, 646-669.

[9] Palop J. (7 de agosto de 2014) Retrasos y sobrecostes en las obras públicas, una lacra... de Alemania. El Mundo. Recuperado de https://www.elmundo.es/

[10] Real Decreto Legislativo 3/2011. Boletín Oficial del Estado, núm. 276, de 16 de noviembre de 2011.

[11] Vélez A.M. (28 de junio de 2017) El retraso del almacén nuclear obliga a pagar a Francia más de 67.000 euros al día desde este sábado. eldiario.es. Recuperado de https://www.eldiario.es/

[12] Von Neumann J., Morgenstern O. (1944) Theory of Games and Economic Behavior. Princeton, NJ, US: Princeton University Press.