FACULTADE DE MATEMÁTICAS

Traballo Fin de Grao

Técnicas matemáticas de planificación e programación de proxectos.

Mario Picáns Rey

2019/2020

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

GRAO DE MATEMÁTICAS

Traballo Fin de Grao

Técnicas matemáticas de planificación e programación de proxectos.

Mario Picáns Rey

Xullo de 2020

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Traballo proposto

Área de Coñecemento: Estatística e Investigación Operativa
Título: Técnicas matemáticas de planificación e programación de proxectos.
Breve descrición do contido
As técnicas matemáticas de programación de proxectos nacen a mediados do século XX, a raíz da necesidade dun medio eficaz de planificación para a dirección de proxectos, e por outra banda, ao desenvolvemento das matemáticas coas que podemos resolver problemas que anteriormente eran moito máis complexos. Neste traballo pretenderemos facer un estudo das principais técnicas de programación de proxectos, comezando no método PERT, pasando por outras que cobren carencias que van xurdindo, como o método de programación a mínimo custo, ou a programación con recursos limitados. Estudaremos as técnicas das que dispoñemos para afrontalos, e finalmente aplicaremos parte do exposto a un caso práctico con datos reais.

Índice xeral

Resumo **VIII**

Introdución **XI**

0.1. Historia XI

0.2. Obxectivos XII

1. PERT-CPM **1**

1.1. Nocións básicas de grafos e redes 1

1.2. Actividades sobre Arcos (AoA) 3

1.3. Construción dun grafo PERT 5

1.4. Asignación dos tempos ás actividades 7

1.5. Cálculo tempos «last» e «early» 8

1.5.1. Matriz de cálculo para os tempos *early e last*: Matriz de Zaderenko . 11

1.6. Folguras e camiño crítico 13

1.6.1. Folgura libre 14

1.7. O calendario do proxecto 15

2. Programación de proxectos a mínimo custo **19**

2.1. Relación entre a duración e o custo. 19

2.2. O método MCE 20

2.2.1. Optimización da duración das actividades coma un problema de programación lineal. 22

2.3. Aplicación a un caso na práctica 24

2.4. Algoritmo de Ackoff-Sasieni 26

3. Programación de proxectos con recursos limitados **29**

3.1. Nivelación de recursos 30

3.1.1. A nivelación de recursos nun modelo de programación matemática . 32

3.2. Asignación de recursos 34
3.2.1. A asignación de recursos nun modelo de programación matemática . 34
3.3. Algoritmos heurísticos 35
3.3.1. Algoritmo de Burgess-Killebrew 35
3.3.2. Algoritmo de Wiest-Levy 38

4. Aplicación do método PERT a un caso real **41**
4.1. Tempos *early* e *last* 45
4.1.1. Tempos *early* 45
4.1.2. Tempos *last* 46
4.2. Folguras 46
4.3. Calendario do proxecto 47

Bibliografía **51**

Código de R **53**

Resumo

As técnicas de programación e planificación de proxectos son métodos que axudan á dirección dun proxecto, relacionando todos os factores que interveñen, e presentando a situación dende unha perspectiva máis clara, facilitando a toma de decisións e a análise dun proxecto. Os principais obxectivos que teñen as diferentes técnicas de planificación, son a optimización da duración e do custo das actividades que conforman o proxecto, a distribución óptima dos recursos, e a estimación da duración total do proxecto.

Neste traballo abordaremos este tema estudando as distintas fases dun proxecto, e empregando as distintas ferramentas matemáticas das que dispoñemos para enfocar os problemas de optimización que xorden, como os algoritmos heurísticos.

Abstract

Project scheduling is the discipline that helps the management of projects, studying the factors that are involved, and presenting the situation from a clearer perspective, in order to simplify decision-making and the analysis of projects. The main objectives of the different project scheduling techniques are, to optimize the duration and cost of the activities that make up the project, to find the optimal distribution of resources, and to estimate the total duration of the project.

In this work, we approach this issue by studying the different stages of a project, and using the mathematical tools that we have to focus on the optimization problems that appear, such as heuristic algorithms.

Introdución

O obxectivo deste traballo é o estudo das técnicas deseñadas para axudar na planificación, programación e control de proxectos. Que é un proxecto? Definimos un proxecto como o conxunto de tarefas ou actividades que deben realizarse ao longo do tempo para crear un produto ou servizo único. Chamamos xestión de proxectos á disciplina de planificación, organización e administración de recursos para conseguir de forma óptima o obxectivo fixado.

0.1. Historia

O método PERT (Program Evaluation and Review Technique),[1] e o método CPM (Critical Path Method)[2] xorden aproximadamente á mesma vez (1958), e pese a que comezaron a partir de investigacións independentes, esencialmente son iguais, con lixeiras diferencias tanto en notación, como en aspectos formais. O desenvolvemento do PERT comeza no 1957, no contexto da Guerra Fría, a raíz dos graves problemas de coordinación e control da Mariña dos EEUU no proxecto Polaris, un proxecto de submarinos atómicos armados con proxectís "Polaris". Debido á dificultade de programar dito proxecto, xa que os Estados Unidos tiñan que manter relación con "250 contratistas directos, con máis de 9.000 subcontratistas, e cunha gran cantidade de axencias gobernamentais", a Armada estadounidense decidiu comezar unha investigación co obxecto de obter novas técnicas máis perfeccionadas de programación e control de proxectos. En colaboración coa compañía de Enxeñeiros Consultores Booz, Allen e Hamilton, iniciáronse os conceptos básicos do sistema PERT, co seu primeiro nome, Program Evaluation and Research Task. Xa no primeiro informe da Mariña establécese o que será o seu nome actual, *Program Evaluation and Review Technique*. O resultado da aplicación desta nova técnica foi a redución en dous anos, dun proxecto cuxa duración total estaba estimada en cinco. Ante a boa acollida do método, comezou a utilizarse rapidamente na industria e comercio.

[1] Evaluación de Programas e Investigación de Tarefas

[2] Método do Camiño Crítico

Por outra banda, no 1957, a empresa E. I. DuPont, tentou desenvolver un sistema que puidese mellorar os métodos de planificación e programación que controlaban os proxectos nas súas plantas de fabricación. Baixo a dirección de James E. Kelley, e Morgan R. Walker, da división de Enxeñería da Dupont, crearíase a técnica CPM (Critical Path Method).

Como comentamos anteriormente, estes dous métodos son similares en moitos aspectos. Aínda que máis adiante se xustificará con detemento, a principal diferencia consiste en que o PERT supón que as duracións das tarefas son aleatorias, mentres que pola outra banda, o CPM considera que os tempos son unha cantidade fixa.

0.2. Obxectivos

Estos métodos, como o PERT ou o CPM, son utilizados para planificar un proxecto optimizando os tempos e os medios dos que se dispoñen. Poden evitar problemas, expoñendo as circunstancias do proxecto. Unha das grandes dificultades coas que se pode atopar a dirección dun proxecto, maiormente cando se está ante proxectos complexos, é o feito de coordinar as diferentes actividades ou tarefas de cada proxecto. O problema habitual nestes casos é que os distintos grupos de traballo do proxecto teñen os seus propios plans, o que adoita desembocar nunha falta de coordinación no proxecto como conxunto. Unha virtude das técnicas de programación de proxectos das que vimos falando, é a preparación do plan de forma gráfica, xuntando e coordinando todas as actividades que interveñen.

De maneira xeral, os principais obxectivos das técnicas que imos estudar neste traballo, segundo Escudero (1977) poderían resumirse en tres puntos:

a) Optimización dende un punto de vista económico, segundo duración das distintas actividades que o compoñen, custo de execución das mesmas e os recursos dispoñibles;

b) Información á dirección sobre as actividades críticas no proxecto, facilitando a vixilancia e control do mesmo e distribución óptima dos recursos dispoñibles;

c) Información á dirección sobre a duración total estimada do proxecto, así como sobre os límites entre os que pode oscilar a devandita duración debido a unha posible aleatoriedade nos tempos de execución das actividades a realizar;

d) Información sobre o custo total mínimo para cada unidade de tempo en que se reduza a duración do proxecto, apuntando entón para que actividades e en que contía habería que incrementar os recursos produtivos.

Entre as principais vantaxes que poden proporcionar estas técnicas á dirección dun proxecto están, por exemplo, que traballos hai e cantos serán necesarios en cada momento; cales son as actividades que, de atrasarse, atrasen a duración total do proxecto; que traballos será necesario realizar primeiro, e cales poden atrasarse ou realizar máis máis tarde.

No segundo capítulo, tentaremos dar unha solución ao problema que xorde cando queremos planificar un proxecto con custe total mínimo, e duración óptima. Afondaremos na cuestión dándolle un enfoque de problema de programación matemática, e resolvéndoo mediante o algoritmo de Ackoff-Sasieni, un algoritmo heurístico.

No terceiro capítulo estudaremos, que ocorre cando estamos nunha situación real, na que non dispoñemos de recursos ilimitados. Explicaremos o proceso de construción dunha planificación dun proxecto, pero coa dispoñibilidade limitada de recursos. Como veremos máis adiante, as principais problemáticas que aparecen son a nivelación e a asignación de recursos. Tamén veremos métodos heurísticos que nos axudarán a resolver os problemas que comentamos antes, nuns tempos de computación razoables.

Finalmente, aplicaremos parte dos contidos expostos para estudar a situación dun caso real e avalialo.

Capítulo 1

PERT-CPM

Como explicamos anteriormente, o obxecto de estudo deste traballo son os métodos de control de proxectos, e unha ferramenta que nos facilita esta análise é a representación gráfica dos proxectos.

O método PERT descompón o proxecto en obras parciais ou actividades, entendendo por actividade a execución dunha tarefa e os recursos que precisa para a súa realización (man de obra, máquinas, materiais...). Por exemplo, quitar a maleza, colocar baldosas, ou levantar un muro, son actividades nun proxecto de construción dun parque de xogos escolar. Logo está o concepto de suceso, que é un punto no tempo, unha data no calendario que só indica o principio ou fin dunha actividade, ou dun conxunto de actividades.

Antes de adentrarnos na construcción de redes das actividades dun proxecto, daremos unhas nocións básicas de Teoría de Grafos.

1.1. Nocións básicas de grafos e redes

Segundo González Díaz (2017), un grafo é un par (N,M) consistente nun conxunto N de elementos chamados nodos ou vértices, e un conxunto M cuxos elementos representan arcos. Unha rede é un grafo cun ou máis números asociados con cada arco ou nodo, que poden representar distancias, custos, duración...
Segundo como sexan os elementos de M podemos distinguir entre dous tipos de grafos:

- **Grafos dirixidos:** Un grafo dirixido ou orientado é aquel no que $M \subset N \times N$, i.e., os arcos son pares ordenados; o arco (i, j) empeza no nodo i e termina no nodo j.
- **Grafos non dirixidos:** En un grafo non dirixido M está composto por subconxuntos

de N de dous elementos. Neste caso, como $\{i, j\}$ e $\{j, i\}$ son o mesmo conxunto, representarán o mesmo arco.

De forma gráfica, o carácter dirixido representámolo con frechas, e no caso non dirixido, esta condición nola dá a ausencia delas. Cabe destacar que non consideraremos grafos nos que haxa "lazos", que son arcos da forma (i, i) ou $\{i, i\}$. Indicamos co m e co n o número total de nodos e arcos respectivamente.

Un **subgrafo** dun grafo G é un grafo $G' = (N', M')$ que ten todos seus vértices e arcos en G, i.e, $N' \subseteq N$ e $M' \subseteq M$.

Se o arco (i, j) está presente nun grafo G, dicimos que os nodos i e j son adxacentes e tamén que son incidentes co arco (i, j), e de xeito equivalente, o arco (i, j) é incidente cos nodos i e j. O grado dun nodo calquera dun grafo G é o número de arcos incidentes con el.

Sexa G un grafo non dirixido e sexa $(a_1, a_2, \dots, a_r)$ unha secuencia de arcos en G. Se existen vértices $(v_0, v_1, \dots, v_r)$ tales que, para $l \in \{1, 2, \dots, r\}, a_l = (v_{l-1}, v_l)$, dicimos que a secuencia é unha **cadea**. Á hora de referirnos a unha cadea usaremos indistintamente a secuencia de arcos ou de nodos que a forma.

Un tipo de cadea que é un concepto fundamental para poder establecer a estrutura dos métodos modernos de programación e control de proxectos é o camiño. Nun grafo, un camiño se define como unha cadea na que tódolos vértices son distintos, é dicir, unha sucesión de arcos tales que se un arco incide no vértice i, o arco seguinte da sucesión sae dese mesmo vértice. Véxase a Figura 1.1.

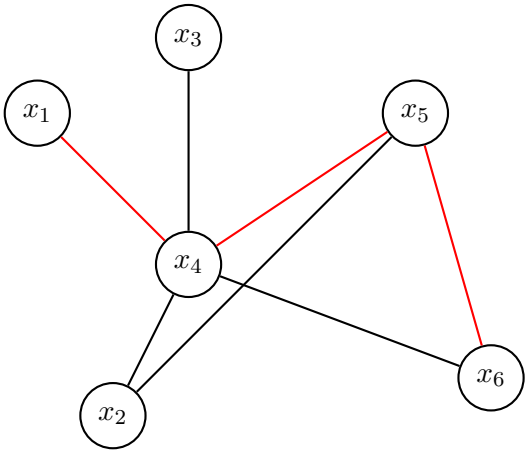


Figura 1.1: Exemplo de camiño

Un caso particular de camiño é aquel no que o vértice inicial coincide co vértice final.

Nesta situación, o camiño recibe o nome de circuíto.

Cando falamos de problemas reais, é de moita utilidade asignarlle aos diferentes arcos ou nodos parámetros (valores numéricos), que representan, ou ben un custo, unha distancia, un tempo de execución... Outro caso semellante é se lle asociamos o número aos nodos. Neste caso interesaríanos ter os tempos de execución asociados a cada actividade, e falaríamos da "**rede**" do proxecto.

O conxunto de actividades dun proxecto e as súas prelacións, poden representarse usando dous formatos: como Actividades sobre os Nodos (AoN[1]), ou como Actividades sobre os Arcos (AoA[2]). No método PERT usaremos o formato AoA, isto é, cada arco (i, j) do grafo representará unha actividade do proxecto.

1.2. Actividades sobre Arcos (AoA)

No formato AoA, as actividades móstranse mediante os arcos na rede, mentres que os nodos son os eventos (ou sucesos), que denotan o inicio ou/e final do conxunto de actividades do proxecto. Desta forma, o vértice 1 da Figura indica o suceso inicio da actividade A, e o vértice 2 a fin de dita actividade. É importante dicir que a lonxitude dos arcos non teñen que ver co tempo previsto de execución da actividade á que representan. No grafo que imos construír asociaremos a cada actividade (arco) a súa duración.

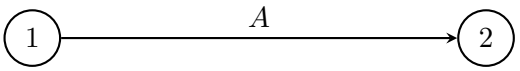


Figura 1.2: Exemplo

Unha vez o proxecto está dividido en actividades, temos que establecer as prelacións que hai, que nos indican a orde na que deben ser realizadas. Hai casos que ben por razóns de tipo técnico, económico ou xurídico. Por exemplo, na construción dun edificio temos que nivelar o terreo antes de poñer os cimientos, ou para poder comezar unhas obras, temos que ter todos os permisos administrativos correspondentes.

O caso máis sinxelo son as prelacións lineais, nas que é necesario que rematara a actividade precedente para poder iniciar a seguinte.

[1] Activity-on-the-node
[2] Activity-on-the-Arc

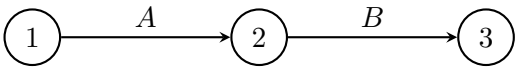


Figura 1.3: Exemplo

De forma similar, temos os casos de converxencia, diverxencia, e converxencia-diverxencia. Son os casos nos que para que comece un conxunto de actividades, ten que ter rematado máis dunha actividade precedente

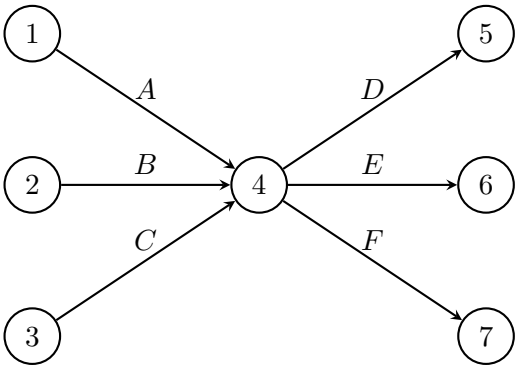


Figura 1.4

Agora ben, poden darse problemas cando nun grafo PERT se presentan casos de prelacións lineais, e de converxencia (ou diverxencia), simultaneamente. Por exemplo, no caso da Figura 1.5 queremos representar as prelacións seguintes: A e B preceden a C, e A precede a D.

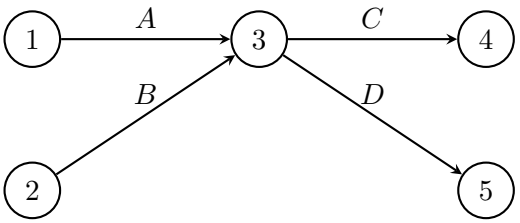


Figura 1.5

Como podemos observar, non reflexa correctamente o que queremos expresar. Para solucionar este problema, co cal nos atoparemos na maioría de escenarios, temos que recorrer ás actividades ficticias. Introducimos un arco "virtual" entre as actividades que considera-

mos, que non consumen tempo nin recursos, e que nos permitirán reflectir as prelacións existentes do exemplo anterior. Esta actividade, evidentemente, non altera nin a duración nin o custo total do proxecto.

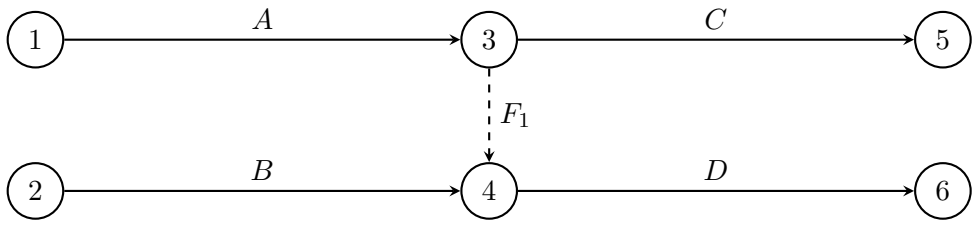


Figura 1.6

1.3. Construción dun grafo PERT

Para comenzar a construción dun grafo PERT, introduciremos os conceptos de suceso inicio do proxecto, e suceso fin do proxecto.

Defínese suceso inicio (de xeito equivalente fin), como o suceso que representa o comezo (fin) dunha actividade, pero que non é a fin (comezo) de ningunha outra. De forma gráfica, no grafo observaremos que, por exemplo, o suceso inicio é o único vértice do que saen arcos, pero non chega ningún. Cabe destacar, antes de comezar o desenvolvemento do proceso, que a única condición que ten que cumprir a numeración dos vértices, é que $i < j$, é dicir, o número do suceso comezo dunha actividade ten que ser menor que o número do vértice do suceso fin da devandita actividade.

Antes de dar un exemplo de grafo PERT, é conveniente ter claras unha serie de "condicións" que ten que cumprir un grafo deste tipo.

- Todo grafo PERT debe ter un nodo inicio e un nodo fin.
- Excepto os sucesos inicio e fin, todo suceso terá unha actividade que a preceda, e outra que a suceda.
- Os arcos representan precedencia lóxica.
- O número de identificación dun nodo sucesor non pode ser inferior ao número de identificación do nodo predecesor, polo que a numeración dos nodos do proxecto faise de esquerda a dereita en orde ascendente.

Sabendo esto, só nos queda coñecer todas as actividades que conforman o proxecto, así como as prelacións entre elas. Para recoller de forma sistematizada a información contida no conxunto de prelacións, existen dous procedementos: a matriz de encadeamentos e o cadro de prelacións.

A **matriz de encadeamentos** é unha matriz cadrada de dimensión igual ao número de actividades nas que se descompón o proxecto. Cando un elemento da táboa está marcado, significa que para poder comezar a actividade que corresponde á fila, ten que ter terminado a actividade correspondente á columna.

O **cadro de prelacións** consiste en dous columnas, onde na primeira se representan todas as actividades nas que se descompón o proxecto, e na segunda, onde aparecen as actividades precedentes.

Exemplo 1.1. Para esclarecer estas explicacións, engadiremos un exemplo. Sexa un proxecto con seis actividades, sendo as súas prelacións:

- A precede a C,D,E
- B precede a E
- C,D,E preceden a F

Agora podemos debuxar a súa matriz de encadeamentos e o cadro de prelacións, para poder observar esta información de forma máis visual:

	A	B	C	D	E	F
A						
B						
C	X					
D	X					
E	X	X				
F			X	X	X	

Cadro 1.1: Matriz de encadeamentos

Actividades	**Precedentes**
A	-
B	-
C	A
D	A
E	A,B
F	C,D,E

Cadro 1.2: Cadro de prelacións

Como podemos observar, as actividades inicio, non teñen ningún precedente.
Agora debuxaremos o grafo PERT deste exemplo:

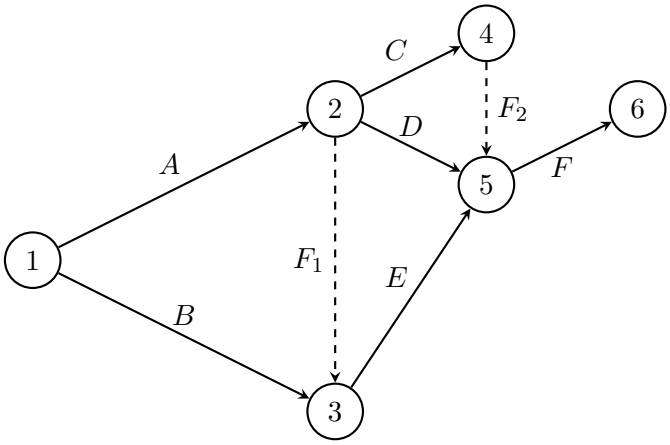


Figura 1.7: Grafo PERT do exemplo

E vemos que tendo a información mostrada nas táboas, é sinxelo debuxar o grafo PERT do proxecto co cal estamos traballando.

Unha vez determinado o diagrama do proxecto, o seguinte paso é a programación deste. Na maior parte dos casos non podemos determinar a duración dunha actividade, xa que depende de circunstancias aleatorias que poden alterar os tempos. A nosa misión será obter os datos que nos permitan traballar. Como se comentou con anterioridade, a principal diferencia do método PERT co CPM é a estimación das duracións das actividades.

1.4. Asignación dos tempos ás actividades

Cando traballamos co PERT, enfocamos o problema da aleatoriedade dos tempos facendo tres estimacións: a estimación optimista (asignámoslle a letra a), que sería o tempo mínimo no que se pode realizar a actividade, con probabilidade de finalizarse nesta estimación inferior ao 0,01; a estimación máis probable (m), representa o tempo no que normalmente se realiza a actividade; e a estimación pesimista (b), que representa o tempo máximo no que realizar a actividade, se todos os contratempos posibles se deran, con probabilidade de finalizarse nesta estimación inferior ao 0,01. Este enfoque asume que a estimación da duración das actividades están feitas por alguén que está familiarizado con elas, e ten coñecemento abondo nas características de cada unha.

Logo de facer as tres estimacións, a media da duración da actividade calcúlase mediante a fórmula:

$$t = \frac{a + 4m + b}{6} \tag{1.1}$$

A distribución que supoñemos no PERT para as duracións das actividades, é unha variación da distribución beta. O cambio consiste en asumir que a esperanza ten a forma:

$$E[X] = \frac{a + 4m + b}{6}$$

Sendo a, m, e b os valores mínimo, máis probable, e máximo, da variable. Esta suposición fíxoa por primeira vez Charles Edwin Clark no seu traballo *"The PERT Model for the Distribution of an Activity Time"*, para poder estimar o efecto da incerteza das duracións das actividades no calendario final do proxecto que estamos a avaliar co método PERT.

Cabe destacar, que na práctica, podemos atoparnos con distribucións doutro tipo, como a triangular, a normal, ou mesmo outro tipo de distribución beta. Por eso mesmo, moitos autores critican que se use este suposto. Outros investigadores, como MacCrimmon e Ryavec, demostraron que usando a distribución triangular, os erros cometidos son moi semellantes aos que obtemos ao usar unha distribución beta.

Así é todo, no contexto determinista do PERT, que é no que estamos traballando, úsase esta distribución dende os anos 70 na maioría dos casos.

1.5. Cálculo tempos «last» e «early»

Unha das aplicacións que xustifica a utilidade do método PERT, é que podemos dar unha data aproximada de cando agardamos que finalice o proxecto en cuestión, e para iso necesitamos introducir varios conceptos.

Comezamos explicando os **tempos *early***[3], que son os que tratan de medir o tempo mínimo necesario para chegar a un determinado suceso, i.e., o máis pronto posible que podemos chegar a el. Procedemos a iteración partindo dun grafo PERT, de esquerda a dereita, asignándolle ao suceso inicio un tempo *early* de 0. Seguidamente, calculamos os tempos *early* dos sucesos representados polos seguintes vértices. Expresado formalmente, para o suceso j, quedaría:

$$t_j = \begin{cases} 0 & se \quad j = 1 \\ max\,[t_i + t_{ij}], \forall i & \text{noutro caso} \end{cases} \tag{1.2}$$

[3]Podería traducirse como "tempo máis pronto posible"

Onde t_{ij} é a duración da actividade que ten por suceso inicial i, e por suceso final j. Podemos observar de maneira intuitiva, que no grafo dun proxecto calquera, o tempo *early* dun suceso daríanos a lonxitude do camiño máis longo dende o suceso inicio ata el. Observemos esta definición cun exemplo fácil, sobre a rutina de todas as mañás despois de despertarnos:

Exemplo 1.2.
Primeiro definimos as actividades correspondentes a todo o proxecto, que logo usaremos tamén para explicar os tempos *last*:

- A - Ducharse
- B - Vestirse e prepararse
- C - Almorzar
- D - Facer a cama e recoller o cuarto
- E - Coller as chaves, a carteira e os apuntamentos.
- F - Asegurarse de que todas as luces quedan apagadas
- G - Pechar a porta e baixar no ascensor.

Agora vemos as prelacións que existen:

- A precede a B.
- B precede a C.
- B e D preceden a E.
- E precede a F.
- C e F preceden a G.

A súa táboa de prelacións, e os seus tempos PERT (en minutos), serían:

Actividade	**Precedente**
A	-
B	A
C	B
D	-
E	B,D
F	E
G	C,F

Cadro 1.3: Táboa de prelacións

	A	B	C	D	E	F	G
Tempo PERT	10	3	15	5	2	1	2

Cadro 1.4: Táboa cos tempos de execución

Agora construímos o seu grafo correspondente:

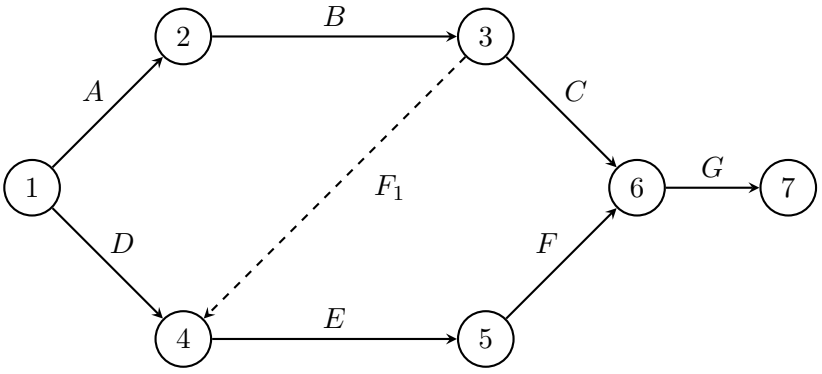


Figura 1.8: Grafo PERT do exemplo

Calculamos o tempo *early* dun suceso j, elixindo o máximo entre as sumas dos tempos *early* dos sucesos que rematan en j e a duración das devanditas actividades. No noso exemplo, é trivial que o tempo *early* do suceso 2 é 10

$$t_2 = max[t_1 = 0 + t_{12} = 10] = 10$$

No nodo 3 só teríamos que sumarlle ao tempo *early* do nodo 2, a duración da actividade que hai entre 2 e 3, o que nos daría que $t_3 = 10 + 3 = 13$. No caso do nodo 4 non é tan evidente, vexamos que:

$$t_4 = max[(t_1 = 0 + t_{14} = 5) = 5, (t_3 = 13 + t_{34} = 0) = 13] = 13$$

Como podemos observar, coincide coa propiedade que demos anteriormente de que o tempo *early* coincide co camiño máis longo dende o suceso inicio ata 4. Aplicando de forma iterativa a fórmula (1.2) a todos os sucesos do proxecto, obtemos os seus tempos

early, chegando a obter o do suceso final. É de especial interese o tempo «máis pronto posible» do suceso final do proxecto, xa que nos indica o tempo mínimo necesario para poder finalizar o proxecto. Este tempo será o obxectivo a conseguir, o que chamamos a **duración do proxecto**. No noso caso, a duración total sería de 30 minutos.

Con isto definido, podemos pasar agora aos **tempos *last***[4]. En contraste cos tempos *early*, aquí tratamos de obter o máis tarde que poden terminar as actividades que converxen nun nodo, para que non se atrasen as que parten de el, e por ende, o proxecto en total. Segue sendo un proceso iterativo, pero agora de dereita a esquerda do grafo, de maneira que comezamos polo suceso fin do proxecto, ao que se lle asigna un tempo *last* igual ao tempo *early* que se lle asignou anteriormente, xa que é a duración do proxecto. Seguidamente, procedemos cos seguintes vértices, que expresado formalmente, para un certo suceso i, quedaría:

$$t_j^* = \begin{cases} t_i & se \quad j = n \\ \\ min\,[t_j^* - t_{ij}], \forall j & \text{noutro caso} \end{cases} \tag{1.3}$$

Onde t_{ij} segue sendo a duración da actividade que ten por suceso inicial i, e por suceso final j. Igual que antes, isto ten unha interpretación gráfica. O tempo *last* dun suceso dado i, é igual á lonxitude do camiño máis longo dende o suceso inicio ata o fin, menos a do camiño máis longo dende i ata o fin. Agora podemos desenvolver esta definición seguindo o grafo do Exemplo 1.2, pero agora facendo as iteracións de dereita a esquerda coa fórmula (1.3). Por exemplo, o tempo *last* do noso vértice 3, sería igual a:

$$t_4^* = min[(t_3^* = 13 - t_{34} = 0) = 13, (t_5^* = 27 - t_{45} = 2) = 25] = 13$$

De forma iterativa, seguimos calculando os tempos *last* de todo todo o proxecto. Como apunte, é interesante saber que deben coincidir os tempos *last* e *early* do primeiro suceso, no noso caso é igual a cero. De non ser así, é porque houbo erros nos cálculos.

1.5.1. Matriz de cálculo para os tempos *early e last*: Matriz de Zaderenko

Cando nos enfrontamos a proxectos moi grandes, é moi difícil facer este procedemento de cálculo. Ante esta problemática, o Doutor Sergio Gregory Zaderenko en "Sistemas de programación por camiño crítico" propuxo un método matricial de cálculo para os tempos *early e last*, que pode ser usado tanto en problemas sinxelos, como en proxectos con grafos

[4]Podería traducirse como "tempo máis tarde permisible"

PERT máis grandes. Máis adiante, veremos como este algoritmo tamén nos facilita a resolución axudándonos dun ordenador, se enfocamos a optimización do proxecto como un problema de programación lineal.

O método de Zaderenko comeza construíndo unha matriz $n \times n$, sendo n o número de nodos que ten o grafo. No interior da matriz escribiremos os tempos PERT que corresponden ás actividades que saen do nodo que ten por número o da fila da súa posición na matriz, e que chegan ao nodo que ten por número o da columna. Nesta matriz tamén calcularemos os tempos *early e last*, agregando unha columna e unha fila adicional respectivamente. Para os tempos *early*, comezamos asignándolle o cero ao primeiro elemento da columna, xa que é o tempo *early* do suceso inicial. Para os seguintes tempos operamos da seguinte forma, sumamos os elementos da columna que corresponde ao nodo cuxo tempo *early* estamos calculando, aos elementos da columna que engadimos que representen nodos nos que nacen actividades que finalizan no suceso do que estamos calculando o seu tempo *early*. De todas as sumas posibles, só nos quedamos co máximo, que será o tempo *early* que estamos buscando.
Para os tempos last, procedemos á inversa, como o fixemos no punto anterior. Comezamos asignándolle ao primeiro elemento da nova fila engadida comezando pola dereita, o tempo *early* do último suceso no cálculo anterior. Para os seguintes tempos do resto de sucesos, faremos da forma: restamos os elementos da fila que corresponde ao nodo cuxo tempo *last* estamos calculando, aos elementos da fila adicional que representan os nodos nos que finaliza o arco que comeza no nodo do que estamos calculando o seu tempo *last*.
Para ilustrar estes métodos de cálculo, acharemos a matriz de Zaderenko para o exemplo 1.2:

t_i	i \ j	1	2	3	4	5	6	7
0	1		10		5			
10	2			3				
13	3				0		15	
13	4					2		
15	5						1	
28	6							2
30	7							
	t_i^*	0	10	13	25	27	28	30

1.6. Folguras e camiño crítico

Unha vez comprendidos estes conceptos, vemos que en calquera proxecto pode haber actividades, que aínda que se atrasen no seu comezo ou finalización, pode non atrasar a duración do proxecto. Pola contra, temos actividades que de demorarse, poden aprazar a data de finalización do proxecto. Coñecemos estas actividades como "críticas", e formando o camiño que resulta da unión de todas elas, obtemos o que chamamos **"camiño crítico"**. Pero agora ben, como sabemos cales son as actividades críticas? Xa que se lles puxo a restrición de non permitir ningún atraso, serán os arcos tales que tanto no seu nodo inicial, como no seu nodo final, os tempos *early* e *last* coinciden. O camiño crítico sería a cadea que forman os nodos nos que se dá esa condición.

A pesares de que estas tarefas requiren unha especial vixilancia, é importante saber que non se poden desatender as actividades non críticas. A propiedade que posúen este tipo de actividades, é a "folgura" que teñen para a execución da súa tarefa. Aínda que os tempos que definimos na sección 1.5 poidan parecer pouco interesantes máis aló dos tempos *early* e *last* do suceso final do proxecto, agora son ideas de vital importancia para o desenvolvemento do concepto de folgura. Defínese **folgura total** dunha actividade H_{ij}^T, como o tempo que resulta de restar ao tempo *last* do suceso final, o tempo *early* do suceso inicial e a duración da actividade. Formalmente:

$$H_{ij}^T = t_j^* - t_i - t_{ij} \tag{1.4}$$

Esta fórmula indícanos canto podemos aprazar a realización dunha actividade co tempo PERT previsto, de forma que a duración total do proxecto non experimente atrasos. Outra propiedade das actividades críticas é que a súa folgura é igual a cero. Podemos velo na actividade do exemplo 1.2:

$$H_{67}^T = t_7^* - t_6 - t_{67} = 30 - 28 - 2 = 0$$

Como podemos observar, se facemos en máis de dous minutos a actividade G, produciríamos un atraso na duración prevista do proxecto. Non así ocorre coa actividade D, xa que:

$$H_{14}^T = t_4^* - t_1 - t_{14} = 25 - 0 - 5 = 20$$

A realización de D podería demorarse vinte minutos con respecto ao tempo PERT, e ese atraso non influiría á duración prevista do proxecto. Da mesma forma poderíamos obter as folguras correspondentes a todas as actividades.
Tamén cabe remarcar, que un atraso nunha actividade, consumindo parte da súa folgura

total, pode facer que se reduza a folgura total das actividades seguintes. Podemos comprobalo facendo que a actividade E consuma toda a súa folgura, que é de 12 minutos. O tempo *early* do nodo 5 pasaría de 15 a 25 minutos, e o tempo *last* do nodo 4 pasa de 25 a 13. Isto acaba implicando que as folguras das actividades E e F, que orixinalmente eran de 12 minutos cada unha, agora sexan de 0 e 2 minutos respectivamente. Con este exemplo búscase remarcar a importancia de que aínda que as actividades críticas teñan fundamental interese, grandes atrasos nas actividades non críticas poden ter consecuencias graves nos tempos do proxecto.

Observación 1.3. A nivel de notación, indicamos o camiño crítico cunha dobre liña, en lugar da liña simple que usamos inicialmente. O grafo do exemplo 1.2, con esta notación para o camiño crítico, quedaría así:

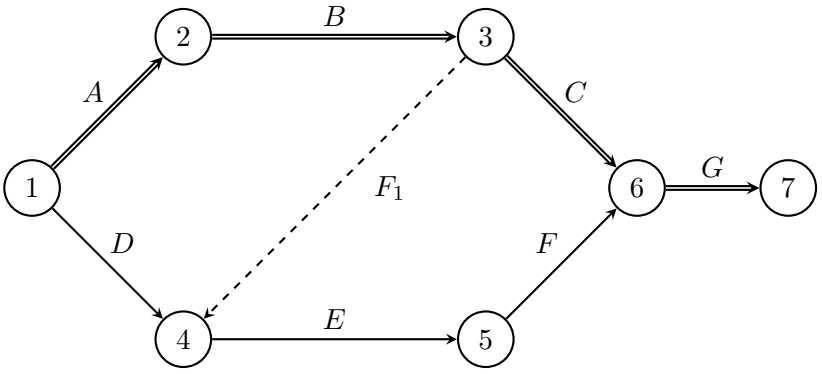


Figura 1.9: Exemplo gráfico do camiño crítico.

Observación 1.4. Se a duración de cada actividade (arco) é o que definimos como "a lonxitude do arco correspondente", entón, é interesante observar que o camiño crítico é o camiño de lonxitude máxima que vai dende o vértice do suceso inicio inicio ata o vértice do suceso final, entendendo por lonxitude de camiño a suma das duracións das actividades que o conforman.

1.6.1. Folgura libre

Unha vez definidos os conceptos do punto anterior, podemos pasar a outro tipo de folgura que ten o seu grao de interese, xa que é coa cal operan os ordenadores electrónicos. Falamos da **folgura libre**, que como define Escudero en "Asignación Óptima de Recursos", é o tempo ou marxe libre que lle queda a unha actividade despois de terse executado «o antes posible» e antes de que comecen as actividades que parten do seu nudo final se estas comezan a súa execución tamén «o antes posible». Con outras palabras, representa o que

podemos consumir da folgura total sen prexudicar ás actividades seguintes. Formalmente, a folgura libre dunha actividade ij defínese:

$$H_{ij}^{L} = t_j - t_i - t_{ij} \tag{1.5}$$

Seguindo co exemplo co que vimos traballando durante os últimos apartados, a folgura libre da actividade D é de 8 minutos (13-0-5), e a súa folgura total era de 20 minutos. A interpretación que lle damos, é que deses 20 minutos, poderemos consumir 8, e aínda así, as seguintes actividades seguir comezando nos seus tempos *early*. Podemos observar que aínda que actividade non sexa crítica, como pode ser o caso de E, pode darse o caso de que a folgura libre sexa cero (15-13-2), polo que non poderíamos atrasarnos ningún minuto sen que iso afectase ás actividades seguintes.

Observación 1.5. A folgura libre dunha actividade sempre será menor ou igual que a súa folgura total, xa que $t_j^* \leq t_j$.

No apéndice "Código de R", temos todos os resultados. Axudándonos do paquete *"ProjectManagement"* construímos un programa que nos calcula os tempos *early*, *last*, e as folguras do noso exemplo.

1.7. O calendario do proxecto

Toda esta información sobre os tempos que vimos definindo dende o punto 1.4, sérvenos para construír un calendario de execución do proxecto co que esteamos traballando, que será fundamental para o control do proxecto.

Para cada actividade definiremos catro datas. Comezamos coa **data de comenzo máis temperá**. Indícanos o máis pronto que pode comezarse unha actividade. Definido para unha certa actividade ij sería:

$$\Delta_{ij} = t_i \tag{1.6}$$

Como podemos observar, esta data é a mesma que a que obtivemos co tempo *early* do suceso inicio da actividade.

A seguinte que definiremos é a **data de comezo máis tardía**, a cal nos indica o máis tarde que pode comezarse a actividade en cuestión, de forma que a duración estimada do proxecto no se demore. A data de comezo máis tardía será igual a:

$$\Delta_{ij}^{*} = t_i + H_{ij}^{T} \quad =^{5} \quad t_j^* - t_{ij} \tag{1.7}$$

[5] Tendo en conta a fórmula da folgura total

A terceira data que explicaremos, é a **data de finalización máis temperá**, a que nos indica o antes que pode finalizarse a execución da actividade ij:

$$\nabla_{ij} = t_i + t_{ij} \tag{1.8}$$

E por último temos a **data de finalización máis tardía**, que nos dá a data tope na que podemos finalizar a actividade ij, de xeito que a duración estimada do proxecto non se postergue. Esta data será igual a:

$$\nabla^*_{ij} = t^*_j \tag{1.9}$$

Observación 1.6. Tanto as datas de comezo, como as de finalización máis temperá e máis tardía dunha actividade crítica, coinciden. É consecuencia directa de que a súa folgura total sexa cero.

Para reforzar as definicións desta sección, daremos como exemplo as catro datas da actividade E. Se comezamos a rutina ás 8.00 a.m, as datas para esta actividade serán:

$$\Delta_{45} = 13 \quad \text{(Serían as 08 : 13 da mañá)}$$
$$\Delta^*_{45} = 27 - 2 = 25 \quad \text{(08:25)}$$
$$\nabla_{45} = 13 + 2 = 15 \quad \text{(08:15)}$$
$$\nabla^*_{45} = 27 \quad \text{(08:27)}$$

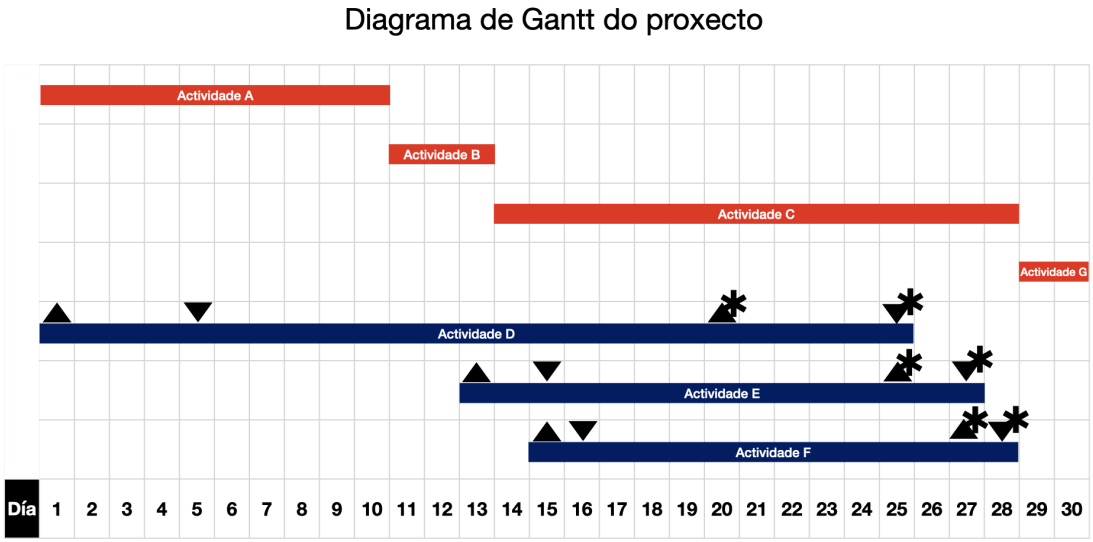


Figura 1.10: Diagrama de Gantt do Exemplo 1.8

Observación 1.7. En vermello marcamos as actividades críticas, en azul o resto. Para as datas de comezo e fin máis temperá e máis tardía, usamos a notación exposta no apartado 1.7.

Capítulo 2

Programación de proxectos a mínimo custo

2.1. Relación entre a duración e o custo.

Como xa vimos no primeiro capítulo, técnicas como o PERT ou o CPM, apórtannos información moi valiosa dun proxecto, como pode ser as actividades primordiais deste, a flexibilidade que poidan ter as datas de execución das tarefas, ou a data media de realización. Sen embargo, non se aborda o tema dos custos das actividades, e é moi importante á hora de comprender as desviacións existentes entre os tempos estimados, e os tempos que se dan na práctica. Se o que se pretende é acelerar algunha actividade para reducir a duración do proxecto, é indubidable que isto pode ocasionar un aumento do custo, e pola contra, unha ampliación na duración dunha actividade produce unha diminución no custo. En moitas ocasións, á hora de encargarnos un proxecto, pódennos impoñer condicións como que se non se termina o plan nun determinado tempo, poden esixirnos unha indemnización, ou concedernos unha prima se adiantamos a data de finalización. Para poder comparar cal pode sernos máis conveniente, é necesario un criterio que elixa a combinación de duración-custo óptima entre varias combinacións.

O método que nos proporciona un estudo da relación destas dúas variables xorde como unha prolongación do método CPM. De feito, J. E. Kelly, un dos desenvolvedores do Método do Camiño Crítico, decidiu incluílo como unha extensión deste. Deste xeito nace a programación de proxectos a mínimo custo, que nós denominaremos como fan algúns autores como C. Romero, método MCE (Minimum Cost Expediting[1]). De forma resumida,

[1] Aceleración do proxecto a mínimo custo.

o desenvolvemento deste tipo de métodos consiste en partir de diferentes posibles tempos do proxecto $(\lambda_1, \lambda_2, ..., \lambda_n)$ segundo sexan os niveis elixidos de utilización dos medios, e para cada un destes niveis, o MCE fixa o tempo de execución das diferentes actividades, minimizando o custo global da realización do proxecto. A continuación, desenvolveremos os métodos MCE.

2.2. O método MCE

Neste novo método que imos definir, consideramos para cada actividade na que se divide o proxecto, dous tempos distintos: o normal e o tempo tope, cos seus correspondentes custos de execución distintos. Formalmente, e xeneralizando para unha actividade (i, j) dun grafo PERT temos, usando a notación do noso libro de referencia "Técnicas de programación e control de proyectos". Sexa (i, j) unha actividade calquera do proxecto (un arco do grafo PERT).

- T_{ij}: É o **tempo normal da execución de** (i, j). Coincide cos tempos máximos asignados no método PERT, e corresponde ao nivel inicial de uso de recursos.

- C_{ijT}: **Custo relativo á execución da actividade** (i, j) **correspondente ao tempo** T_{ij} definido antes (custo mínimo).

- t_{ij}: **Duración correspondente ao nivel máximo de uso dos recursos**, tempo mínimo de realización da actividade (i, j). Definímolo como *tempo tope*.

- C_{ijt}: É o **custo relativo á execución da actividade** (i, j) **no tempo** t_{ij} (custo máximo).

- x_{ij}: As **variables** no método MCE, cada x_{ij} é a duración da actividade (i, j).

De forma gráfica, situamos no eixo de abscisas as duracións das actividades, e nas ordenadas os custos relativos á realización destas.

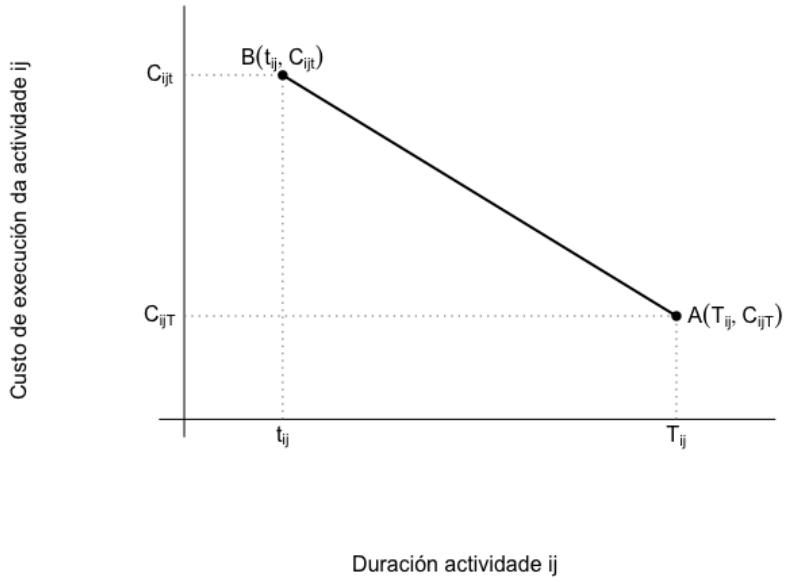


Figura 2.1: Duracións das actividades cos custos inherentes á súa execución

Como podemos observar, temos dous puntos marcados na gráfica. O $A(T_{ij}, C_{ijT})$ é o punto normal, correspondente ao tempo normal (T_{ij}) e ao custo mínimo (C_{ijT}). O outro punto, o $B(t_{ij}, C_{ijt})$ é o que chamamos punto tope, que está asociado ao tempo mínimo de execución (t_{ij}) e ao custo máximo (C_{ijt}). A curva que une estes dous puntos, $C_{ij} = f(x_{ij})$, é a chamada curva custo-duración.

Unha vez visto isto, chegamos a un punto clave no razoamento do método MCE, que se basea na relación entre a diminución dos tempos de execución das actividades e seus correspondentes custos suplementarios, isto é, hai unha proporcionalidade directa entre a redución nos tempos, e os custos derivados destas rebaixas. Así é todo, aínda que non se cumpre esta afirmación na maioría dos casos na práctica, sérvenos como unha achegamento á realidade, e permítenos enfocar o problema da redución de tempos de execución. Isto débese, por exemplo, a que o custo da hora extra por facer un traballo, adoita ser moito máis alto que o da hora normal. Polo tanto, non sería o mesmo o que nos custaría reducir o tempo dunha actividade dentro do horario estipulado, que se precisamos ese mesmo tempo en forma de horas extras.

Observación 2.1. Como dicimos, aínda que unha afirmación tan forte non se cumpre na realidade, a utilidade desta é a de darlle un enfoque que nos axude a resolver a cuestión. De

feito, veremos que nos permite converter o problema de programación de proxectos a mínimo custo, nun problema de programación lineal paramétrica. Doutra forma, teriámonos que enfrontarnos a problemas de programación matemática máis complexos.

O incremento no custo que se dá por reducir o tempo de execución de (i, j) dende o tempo máximo T_{ij} ao tempo mínimo t_{ij}. Temos, entón:

$$S_{ijt} = C_{ijt} - C_{ijT} \tag{2.1}$$

Que chamamos custo asociado á redución da actividade (i, j) de tempo máximo a tempo mínimo.
En conclusión, a recta que relaciona o custo "engadido" S_{ij} con duración x_{ij} de cada actividade (i, j) é:

$$S_{ij} = S_{ijt} - \frac{S_{ijt}}{T_{ij} - t_{ij}}(x_{ij} - t_{ij}) \tag{2.2}$$

A pendente desta recta, representa o custo suplementario no que incorremos cando acurtamos a duración do proxecto nunha unidade de tempo. Denominarémolo a partir de agora como "custo unitario de redución".

2.2.1. Optimización da duración das actividades coma un problema de programación lineal.

Para poder representar o problema que queremos resolver mediante un modelo de programación matemática, o primeiro que debemos facer é fixar a función obxectivo, comezando por calcular o custo engadido total que asumimos cando reducimos os tempos de execución. Obtemos este custo, representado por ϕ, sumando os custo suplementarios correspondentes ás diferentes actividades:

$$\phi = \sum_{\forall(i,j)} S_{ijt} - \sum_{\forall(i,j)} \frac{S_{ijt}}{T_{ij} - t_{ij}}(x_{ij} - t_{ij}) \tag{2.3}$$

Transformamos (2.3) facendo os cambios:

$$\left.\begin{array}{ll} \sum_{\forall(i,j)} S_{ijt} & = K \\ \frac{S_{ijt}}{T_{ij}-t_{ij}} & = Q_{ij} \end{array}\right\} \tag{2.4}$$

E quedaría da forma:

$$\phi = K - \sum_{\forall(i,j)} Q_{ij}(x_{ij} - t_{ij}) \tag{2.5}$$

E se lle engadimos o feito de que $\sum Q_{ij}t_{ij}$ é unha constante que podemos representar por K', facemos o cambio de variables e achamos que:

$$\phi = (K + K') - \sum_{\forall(i,j)} Q_{ij}x_{ij} \tag{2.6}$$

Retomando o obxectivo do que falamos ao comezo deste punto, temos que minimizar a función (2.6), ou o que é o mesmo, maximizar a función oposta. Como $(K + K')$ é unha constante, a función obxectivo do noso modelo de programación dun proxecto a custo mínimo, quedaría:

$$\max \sum_{\forall(i,j)} Q_{ij}xij. \tag{2.7}$$

Agora só nos quedaría obter o conxunto factible, isto é, as restricións que deben verificar as variables x_{ij} que se están a programar.

Para o conxunto factible temos que ter en conta unha serie de condicións. Primeiramente, a duración x_{ij} non pode ser superior ao tempo máximo de execución que definimos anteriormente como T_{ij}, nin inferior ao tempo mínimo, tamén definido antes como t_{ij}. Entón este primeiro conxunto de restricións quedaría como:

$$t_{ij} \leq x_{ij} \leq T_{ij}, \quad \forall(i,j) \tag{2.8}$$

Por outra banda, a segunda restrición que lle impoñeremos á función obxectivo, virá dada pola duración do proxecto. Chamarémoslle λ a esta duración, e sabemos que coincidirá coa lonxitude xeneralizada do camiño crítico. Por este mesmo motivo, sabemos que a lonxitude xeneralizada de calquera camiño que una o suceso inicio do proxecto, co suceso fin, terá que ser menor ou igual que a duración do proxecto λ. Entón, o segundo conxunto de restricións será:

$$\sum_{\forall(i,j)\in W_p} x_{ij} \leq \lambda \quad \forall p \tag{2.9}$$

Observación 2.2. O que temos escrito baixo o signo suma, "$\forall(i,j) \in W_p$", indícanos que a suma estenderase a todos os arcos pertencentes ao conxunto (W_p) formado polos arcos do camiño $p-$ésimo, unindo o suceso inicio co suceso fin do proxecto.

En definitiva, o método MCE conclúe no seguinte problema de programación matemática, xuntando todo o anterior:

$$\begin{gathered} max \quad \textstyle\sum_{\forall(i,j)} Q_{ij}x_{ij} \\ \text{suxeito a :} \quad t_{ij} \leq x_{ij} \leq T_{ij}, \quad \forall(i,j) \\ \textstyle\sum_{\forall(i,j)\in W_p} x_{ij} \leq \lambda, \quad \forall p \end{gathered} \tag{2.10}$$

É dicir, para cada λ obteremos o tempo óptimo de execución de todas as actividades. Polo tanto, conseguimos expresar o problema de minimizar o custo dun proxecto como un problema de programación lineal paramétrica.

Para este tipo de problemas que estamos tratando, os de programar un proxecto a mínimo custo, foron elaborados algoritmos que teñen en conta a estrutura do programa lineal paramétrico como (2.10). Existen, entre outros, algoritmos como os de Kelley ou o de Fulkerson[2], pero nós centrarémonos no algoritmo heurístico de Ackoff-Sasieni.

2.3. Aplicación a un caso na práctica

Antes de comezar co algoritmo, veremos un caso práctico. A partires da seguinte táboa dun proxecto de exemplo, aplicaremos o exposto para explicar o proceso polo cal podemos expresar un problema de minimización de custos, como un problema de programación lineal.

Actividade	Tempo normal de execución (T_{ij})	Tempo tope de execución (t_{ij})	Custo unitario de redución
1 - 2	9	4	2
1 - 3	10	5	4
2 - 5	7	4	3
3 - 4	8	5	4
4 - 5	12	8	5
4 - 6	15	9	2
5 - 6	7	5	1

Cadro 2.1: Problema de apoio para a explicación do método MCE

O grafo calculado de acordo aos tempos normais de execución sería:

[2]Ver Kaufmann, A., Desbazeille, G., *Método do camiño crítico, aplicación aos programas de execución de traballos do método PERT e á optimización dos seus custos*, páx. 131

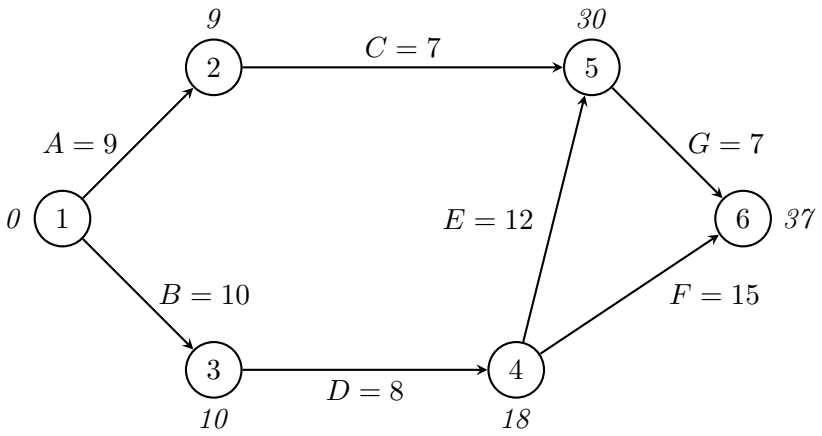


Figura 2.2: Grafo cos tempos normais

E agora representaremos o grafo, pero nesta ocasión atendendo aos tempos tope de execución.

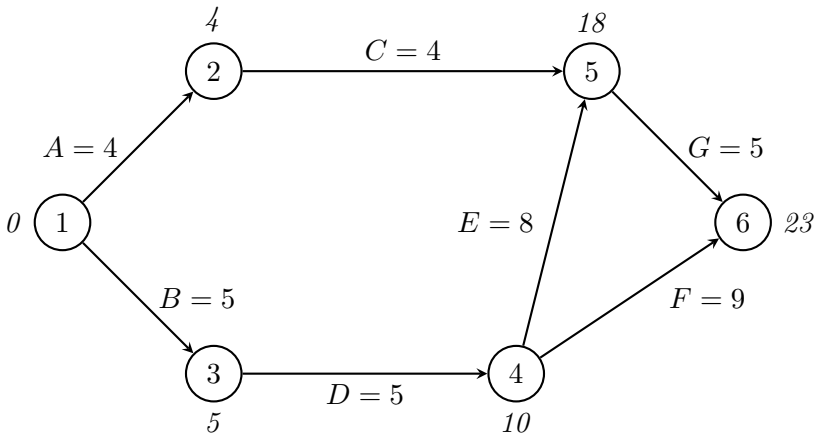


Figura 2.3: Grafo cos tempos tope

Observación 2.3. Ao carón de cada nodo, nos dous casos, escribimos en cursiva o tempo *early* de cada suceso.

Con estes datos xa sabemos que a duración de calquera proxecto ten que estar comprendida entre 23 (tempo *early* do suceso fin do proxecto cos tempos tope), e 37 (tempo *early* do suceso fin do proxecto cos tempos normais).
A continuación, usando os datos da última columna da táboa 2.1, tentaremos expresar a función obxectivo ψ do noso problema de programación lineal. Esta función deberá determinar o tempo de cada actividade, minimizando o custo suplementario inherente a esta.

Obtémola substituíndo os Q_{ij} na función obxectivo que definimos en (2.10), daquela:

$$max\ \psi = 2x_{12} + 4x_{13} + 3x_{25} + 4x_{34} + 5x_{45} + 2x_{46} + x_{56} \tag{2.11}$$

Agora, a primeiro conxunto de restricións resultará de substituír na expresión 2.8, os datos dos tempos normal e tope, logo:

$$\begin{array}{ll} 4 \leq x_{12} \leq 9 & 8 \leq x_{45} \leq 12 \\ 5 \leq x_{13} \leq 10 & 9 \leq x_{46} \leq 15 \\ 4 \leq x_{25} \leq 7 & 5 \leq x_{56} \leq 7 \\ 5 \leq x_{34} \leq 8 & \end{array} \tag{2.12}$$

E por último, aplicando o termo 2.9, obteremos o segundo conxunto de restricións:

$$\begin{array}{rll} x_{12} + x_{25} + x_{56} & \leq \lambda & \\ x_{13} + x_{34} + x_{46} & \leq \lambda & 23 \leq \lambda \leq 37 \\ x_{13} + x_{34} + x_{45} + x_{56} & \leq \lambda & \end{array} \tag{2.13}$$

Observación 2.4. É interesante observar que os primeiros membros da desigualdade 2.13, simbolizan as lonxitudes dos diferentes camiños que van dende o suceso inicio ata o suceso final do grafo.

A finalidade do programa que acabamos de definir, é achar a duración das actividades, finalizando o proxecto nun determinado tempo λ, e minimizando o sobrecusto que resulta de reducir os tempos de execución das actividades. O método MCE guíanos ao problema formado polas expresións(2.11), (2.12), e (2.13), polo que o noso obxectivo agora será resolver un problema desta natureza.

2.4. Algoritmo de Ackoff-Sasieni

O que obtivemos logo de desenvolver a idea do MCE, foi un problema de programación lineal paramétrico, polo que teremos que tentar entender como podemos resolver este tipo de problemas na práctica. Como avanzamos ao final do punto 2.3, a forma máis común de determinar unha solución para un problema desta natureza, é mediante algoritmos específicos, xa que son máis sinxelos de aplicar que os algoritmos típicos da programación lineal paramétrica. Desenvolveremos a continuación, o algoritmo de Ackoff e Sasieni, que é moi doado de aplicar en proxectos dun número non moi grande de actividades. Farémolo apoiando a explicación no exemplo exposto na sección anterior.

O primeiro paso no proceso deste algoritmo é construír unha táboa, onde na primeira columna aparecen os distintos camiños que unen o suceso inicio co suceso fin do proxecto,

as seguintes sete para representar as actividades nas que descompoñemos o proxecto, e debaixo delas aparecen os custos unitarios de redución. Finalmente completamos engadindo unha columna marcada cun **1**, coas lonxitudes dos camiños cos tempos normais, e unha fila coas reducións posibles das actividades. Para o cálculo das reducións facemos tempo normal menos tempo tope.

No noso caso, observamos a columna **1** da táboa 2.2, e deducimos que a duración do proxecto é 37, polo que se queremos reducir a duración total, temos que reducir a duración do camiño III, xa que sería o camiño crítico. E en consecuencia, para diminuír o tempo de execución do camiño, teremos que reducir o dalgunha das actividades. Que criterio temos para escoller que tempo reducimos? Aquela actividade cuxo custo unitario de redución sexa máis pequeno. Neste exemplo, sería a actividade $5-6$. O sobrecusto pola redución da duración calcúlase multiplicando o custo unitario de redución pola redución de tempo que se faga. Neste paso sería $2\times1 = 2$ unidades monetarias de sobrecusto.

Na seguinte columna, sinalada cun **2**, escribimos as novas duracións despois de reducir 2 días nos camiños nos que se executa $5-6$, e na fila **2**, as posibles reducións que teríamos neste paso. Nesta etapa vemos que a duración do proxecto é agora de 35, e que o camiño crítico segue sendo o III. A actividade que menor custo ten é a $5-6$, pero como vemos na fila **2**, non podemos reducila. O seguinte custo máis baixo, neste caso coincide en dous actividades, que son de 4. Como podemos reducir 5 días a actividade $1-3$, fronte aos 3 da $3-4$, escollemos a que máis días nos permita reducir. Procedemos de xeito análogo ao paso anterior, e obtemos que o sobrecusto nesta redución é de 20 unidades monetarias.

A forma de proceder é a mesma no terceiro paso, xa que ao ser 30 a duración total, o camiño crítico é o III. Reducimos a actividade $3-4$, e engadimos unha columna coas novas duracións, e unha fila coas novas posibles reducións. O sobrecusto neste caso é de 12 unidades monetarias.

No seguinte paso seguimos tendo o camiño III como camiño crítico, e a actividade a reducir é a $4-5$. Non obstante, temos que ter en conta, que calquera redución no tempo de execución desta actividade superior a 2, non repercutirá no tempo de execución total do proxecto, xa que a lonxitude do camiño II é de 25, dúas unidades menos que o III. O sobrecusto agora sería de 10. As novas lonxitudes dos camiños están na columna **5**, e como podemos ver na nova fila **5**, non sería posible facer ningunha redución máis, polo que xa chegamos á fin do algoritmo.

Cando rematamos de aplicar o algoritmo, vendo a táboa 2.2, podemos interpretar a información para facer a programación do proxecto a mínimo custo, que é o obxectivo principal. Por exemplo, se queremos finalizar o proxecto en 10 días menos, o sobrecusto mínimo para conseguilo será de 34 unidades monetarias. Podemos ver isto na Figura 2.4 que representa a relación entre sobrecusto por redución do tempo de execución, e o tempo de execució

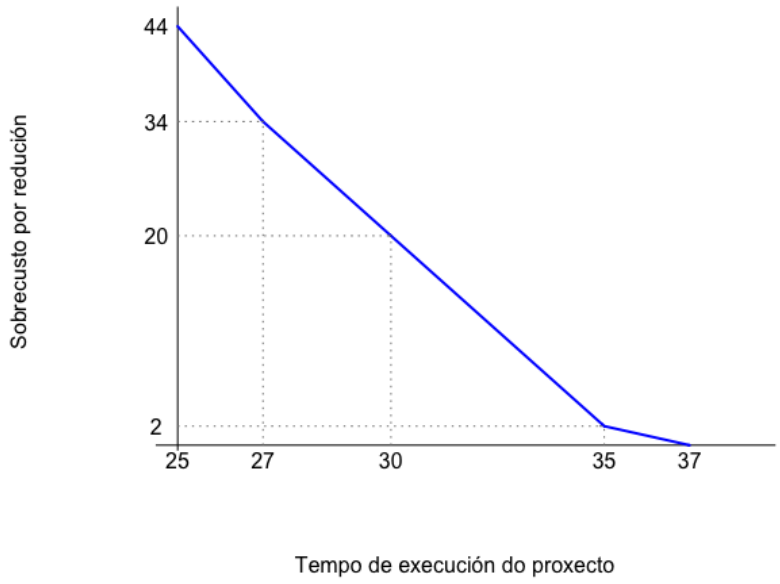


Figura 2.4: Relación sobrecustos con tempo execución

	1 - 2	1 - 3	2 - 5	3 - 4	4 - 5	4 - 6	5 -6	**1**	**2**	**3**	**4**	**5**
I = 1, 2, 5, 6	2		3				1	23	21	21	21	21
II = 1, 3, 4, 6		4		4		2		33	33	28	25	25
III = 1, 3, 4, 5, 6		4		4	5		1	37	35	30	27	25
1	5	5	3	3	4	6	2					
2	5	5	3	3	4	6	0					
3	5	0	3	3	4	6	0					
4	5	0	3	0	4	6	0					
5	5	0	3	0	2	6	0					

Cadro 2.2: Cadro do algoritmo de Ackoff e Sasieni

Capítulo 3

Programación de proxectos con recursos limitados

A programación de proxectos con recursos limitados é un tema que ten moita relevancia, tanto no ámbito académico como na práctica. Debido á complexidade de programar un proxecto con esas restricións, hai moitos traballos nos que se investiga a mellor forma de resolvelos, ou a busca dun algoritmo que faga planificacións factibles tendo en conta a limitación dos recursos.

Ata agora, asumimos que os diferentes recursos que necesitamos (normalmente man de obra, pero tamén materia prima, cartos, materiais...) eran ilimitados, pero a realidade é que na práctica, a súa dispoñibilidade non é infinita. Entón, entendemos por programación de proxectos con este limitante, ao proceso de construción dun calendario do proxecto tendo en conta os medios dispoñibles. Teremos que examinar os posibles desequilibrios á hora do uso dos recursos, para evitar o que coñecemos como "sobreasignacións", situacións que se dan cando requirimos de máis recursos dos posibles. Cando se dá unha situación como a que acabamos de comentar, teremos que reprogramar as actividades afectadas para resolver estas eivas.

Un dos maiores problemas que presentan os métodos que presentamos, como o PERT ou o CPM, é que non se adaptan a esta situación, xa que parten do suposto de que os recursos son ilimitados. A partir dos anos 60, comezan a aparecer publicacións nas que sitúan a estes métodos modernos de programación e control, nun contexto con esas restricións[1]. A continuación presentaremos dous dos principais problemas aos que se chegaron

[1] Ver Burgess, A.R. e Killebrew, J.B., *Variation in Activity Level on a Cyclic Arrow Diagram.*

na investigación no ámbito dos recursos limitados, a nivelación e a asignación de recursos.

En primeiro lugar, falaremos da nivelación de recursos. Os seus principais obxectivos son: que a duración do proxecto non supere a planificada (duración do camiño crítico), e que os consumos dos diferentes medios sexan uniformes. Isto último quere dicir que o contexto ideal é aquel no que o consumo de cada un dos recursos coincide en cada unha das fases do proxecto.

Pola outra banda, os métodos de asignación de recursos tentan reducir o tempo total do proxecto, pero sen que en ningunha etapa deste o uso dos medios supere a dispoñibilidade dos mesmos.

3.1. Nivelación de recursos

Para a explicación dos apartados de nivelación e asignación de recursos, usaremos un exemplo sinxelo para acompañar á exposición. Temos o seguinte grafo PERT:

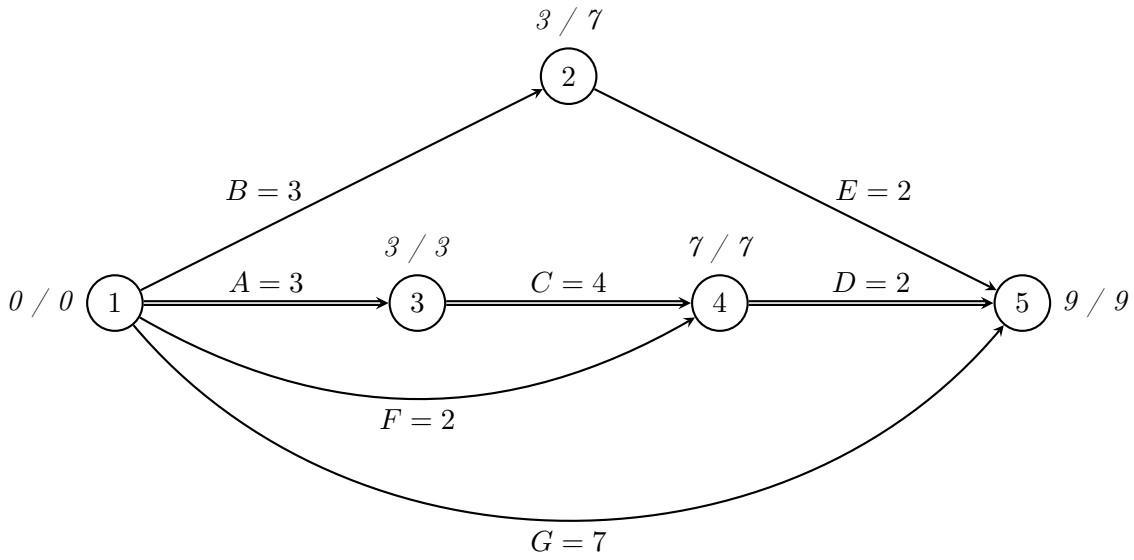


Figura 3.1

Observación 3.1. Ao lado de cada nodo temos escrito o seu tempo *early*, e separado por unha barra, o seu tempo *last*. Ademais, os tempos veñen dados en días.

Vendo a Figura 3.1, observamos que as actividades A, C e D son críticas, xa que a súa

folgura total é nula. As actividades B e E teñen unha folgura total de 4, a F de 5, e a G de 2. Agora calculamos as folguras libres, que recordando a sección 1.6.1, sérvennos para saber canto podemos consumir da folgura total sen prexudicar ás actividades seguintes. As actividades A, B, C, e D teñen folgura libre nula. As actividades E, F, e G, teñen folguras libres de 4, 5, e 2 respectivamente.

Con estes datos poderemos calcular as datas de comezo e finalización máis temperás, construíndo un calendario de execución do proxecto con estas datas. Usando as fórmulas (1.6) e (1.8) obtemos o seguinte calendario:

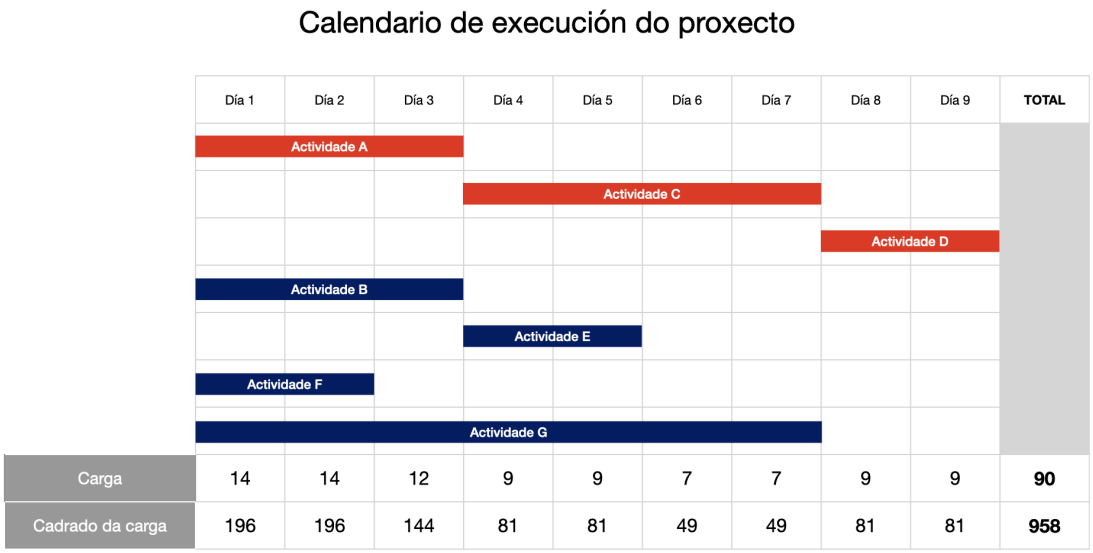


Figura 3.2: Calendario de execución do proxecto

Observación 3.2. En vermello sinalamos as actividades críticas.

Agora falaremos da man de obra que require cada actividade. As actividades A e G requiren 5 obreiros para poder realizarse nos tempos previstos, as actividades B, C, E, e F de 2 obreiros, mentres que a D necesitaría 9 traballadores. Como podemos observar na Figura 3.2, ata o día 2 precisaremos de 14 obreiros, o día 3 requiriríamos de 12, e así obtemos todos os datos na penúltima fila da Figura 3.2, onde podemos ver a carga diaria de traballo.

No seguinte gráfico, podemos ver o consumo dos recursos por día que estamos estudando. No exemplo do noso proxecto, vemos que o uso dos medios é moi dispar. Como sinala

a Figura 3.3, os dous primeiros días necesitamos 14 obreiros, pero entre o día 6 e 7, só de 7 traballadores. Como pode semellar, esta distribución dos recursos é mellorable, e dificulta moito a organización do proxecto. Cos métodos de nivelación que presentaremos, tentaremos conseguir un diagrama de carga o máis plano posible. No exemplo 3.1, debuxamos o que sería o óptimo cunha liña descontinua (90 obreiros / 9 días = 10 obreiros por día). O que temos que conseguir é que a varianza das cargas sexa cero, xa que eso significaría que o consumo por día coincide coa media, pero como esta situación non se dá na gran maioría dos casos, o noso obxectivo será reducir ao mínimo a devandita varianza. O método que empregaremos a continuación, baséase en atrasar ás actividades que non son críticas, de forma que o atraso non supere ás folguras, xa que nesa situación converteríanse en críticas.

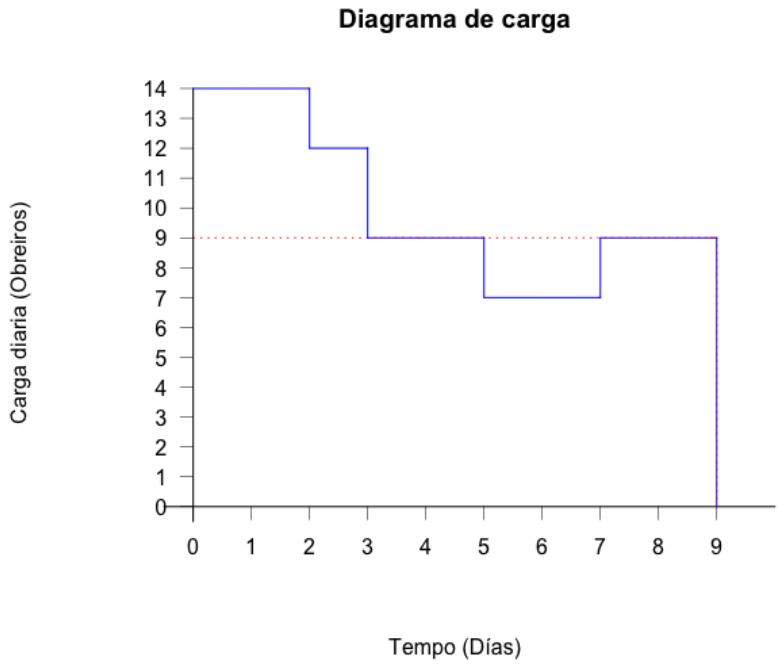


Figura 3.3: Diagrama de cargas do exemplo

Observación 3.3. A liña vermella descontinua é a nivelación óptima

3.1.1. A nivelación de recursos nun modelo de programación matemática

Grazas ás técnicas de programación matemática das que dispoñemos, temos ferramentas para poder resolver de forma precisa os problemas que expuxemos anteriormente. Pero como veremos neste capítulo, cando temos diante un proxecto de grandes dimensións, non é asumible facelo nun tempo de computación razoable, e que como veremos no punto 3.3,

temos outros métodos que nos poden ofrecer unha solución aproximada.

Para explicar a estrutura do noso modelo, comezaremos por definir os parámetros e variables que van intervir:

- i: **Índice** que nos mostra os **recursos** que estamos a estudar. Nun proxecto de m recursos, teremos i = 1,2,... , m.
- j: **Índice** que nos mostra as **actividades** nas que descompuxemos o proxecto. Por exemplo, se temos n actividades, será: $j = 1, 2, \dots, n$.
- k: **Índice** que nos di en cantas **etapas** está divida a execución do proxecto.
- C_{ij}: **Necesidades** do recurso i na actividade j.
- D_j: **Duración** da actividade j.
- x_{jk}: Son as **variables** que estamos programando, impoñéndolles que tomen o valor 0 ou 1. Cando toma 0, significa que non se vai executar na etapa k, e cando toma o valor 1, si se executaría.

A continuación, daremos unha serie de restricións que deberán verificar as variables x_{jk}: A primeira é a que xa avanzamos antes, que as variables deben tomar ou o valor 1 ou o valor 0. Formalmente:

$$\begin{aligned} 0 \leq x_{jk} \leq 1 \qquad & j = 1, 2, \dots, n \\ x_{jk} = \text{número enteiro} \quad & k = 1, 2, \dots, p \end{aligned} \tag{3.1}$$

Observación 3.4. Como temos que usar números enteiros, atoparémonos con cálculos moito máis complexos que na programación continua.

A segunda restrición é que as actividades nas que se descompón o proxecto deben ser realizadas no prazo das p etapas nas que dividimos o proxecto. Logo sería:

$$\sum_{k=1}^{p} x_{jk} = D_j \quad j = 1, 2, \dots, n \tag{3.2}$$

A seguinte restrición que definiremos é a que fixa que non se poida comezar a realización de ningunha actividade antes de que rematen as súas precedentes. Deste xeito asegurámonos de que o programa vai satisfacer as prelacións existentes. Expresamos esta limitación da forma:

$$\sum_{l=1}^{k-1} x_{ql} \geq D_q x_{jk}, \forall q \quad j = 1, 2, \dots, n;\ k = 2, 3, \dots, p \tag{3.3}$$

Observación 3.5. A letra q denota unha actividade anterior á j

E por último, a cuarta restrición é que non se poida dividir a execución de ningunha tarefa, ie, cando comezamos temos que finalizala sen interrupcións. Formalmente:

$$D_j x_{jk} - D_j x_{j(k+1)} + \sum_{l=k+2}^{p} x_{jl} \leq D_j \quad j = 1, 2, \ldots, n;\ k = 1, 2, \ldots, p-2 \tag{3.4}$$

Agora que xa están todos os conxuntos de restricións definidos, podemos expoñer a función obxectivo do modelo. A idea, como indicamos antes, é reducir ao mínimo a varianza da carga total, deste xeito podemos facer que o consumo dos diferentes medios sexa o máis uniforme posible. Como a carga media é unha cantidade fixa, para minimizar a varianza, teremos que minimizar a suma dos cadrados das cargas. A función obxectivo terá a forma:

$$\min \sum_{k=1}^{p} \left[\sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} xjk \right]^2 \tag{3.5}$$

Cando o proxecto é grande, obter a solución deste problema ten un alto custo computacional, polo que se usan algoritmos heurísticos, que dan solucións aceptables en tempos razoables. Presentaremos un exemplo onde faremos uso dun algoritmo heurístico para o problema de nivelación de recursos.

3.2. Asignación de recursos

A diferencia da nivelación de recursos, neste caso o noso obxectivo será que o consumo diario do recursos non supere as reservas existentes do mesmo, sempre tentando que o tempo de execución sexa mínimo. Da mesma forma que cando explicamos a nivelación de recursos, nesta situación o método tamén consistirá en atrasar as actividades non críticas, todo o que nos permitan as súas folguras. No exemplo 3.1, podemos ver que se supoñemos as dispoñibilidades de man de obra en 11 obreiros diarios, non poderíamos cumprir co calendario estipulado, xa que superaríamos o límite durante os tres primeiros días. E de forma análoga ao caso da nivelación, para proxectos máis complexos, na maioría dos casos só podemos obter solucións aproximadas mediante métodos heurísticos que explicaremos con máis detalle no apartado 3.3.2.

3.2.1. A asignación de recursos nun modelo de programación matemática

Neste caso, o problema de programación matemática terá as mesmas restricións que no apartado 3.1.1, e ademais, un novo conxunto de restricións que represente a condición de

que os consumos dos recursos non poden superar á dispoñibilidade destes. Se representamos por a_{ik} a dispoñibilidade do recurso i-ésimo, podemos escribir este conxunto da forma:

$$\sum_{j=1}^{n} C_{ij}x_{jk} \leq a_{ik} \quad i = 1, 2, ..., m;\ k = 1, 2, ..., P \tag{3.6}$$

O que cambia respecto do modelo que presentamos na sección de nivelación de recursos, é a estrutura da función obxectivo. Nesta situación o obxectivo é o de atopar un calendario que cumpra coas restricións que nos impoñen os recursos, e minimice a duración do proxecto. Formalmente:

$$\alpha_0 \sum_{j=1}^{n} x_{j1} + \alpha_1 \sum_{j=1}^{n} x_{j2} + \cdots + \alpha_{P-1} \sum_{j=1}^{n} x_{jP} \tag{3.7}$$

sendo: $\alpha_0 < \alpha_1 < ... < \alpha_{P-1}$.

Coa función (2.6) asegurámonos que minimizamos a duración do proxecto, xa que se o modelo ten que elixir entre executar unha actividade j nunha etapa s ou t, escollerá a etapa s (é dicir, na notación que presentamos antes, $x_{js} = 1$ e $x_{jt}t = 0$) se $\alpha_s < \alpha_t$. Do mesmo xeito que antes, son modelos que teñen un número moi elevado de restricións e variables, aínda que, como no caso da nivelación de recursos, hai moitas variables que toman valores determinados (no caso da actividade E non comeza ata o terceiro día, ie, $x_{5k} = 0,\ k = 1, 2, 3$), o custo computacional na súa resolución é alto. É un dos motivos para achar algoritmos que nos faciliten a resolución deste tipo de problemas, como os algoritmos heurísticos.

3.3. Algoritmos heurísticos

A principal problemática que temos á hora de enfrontarnos a un problema de nivelación ou asignación de recursos, é que se queremos aplicar métodos exactos, teremos que asumir tempos computacionais que en moitos casos, non poderemos permitirnos. Para resolver esta cuestión, foron xurdindo algoritmos heurísticos que solucionan de forma aproximada estes problemas, e nuns tempos moito máis curtos.

3.3.1. Algoritmo de Burgess-Killebrew

O método que presentaremos para a nivelación de recursos é o algoritmo de Burgess-Killebrew, un dos primeiros algoritmos neste campo. Explicaremos o funcionamento desta técnica mediante as fases que a compoñen:

Primeira fase: Co calendario de execución do proxecto diante, buscamos a actividade non crítica que teña a data temperá de finalización máis avanzada. Atrasamos a súa fin unidade por unidade de tempo, todo o que nos permita a súa folgura. Escolleremos como nova data de finalización da actividade a que faga mínima a suma dos cadrados das cargas. Para esta explicación, apoiarémonos no exemplo da figura 3.1, pero facendo unha serie de cambios para que se vexa máis claramente a disparidade da carga dos recursos. Neste novo suposto, para a realización de A e B precísanse 6 traballadores, para C, D, E, e G, 2 traballadores, e para realizar F, 3 traballadores. Vexamos o novo diagrama de cargas:

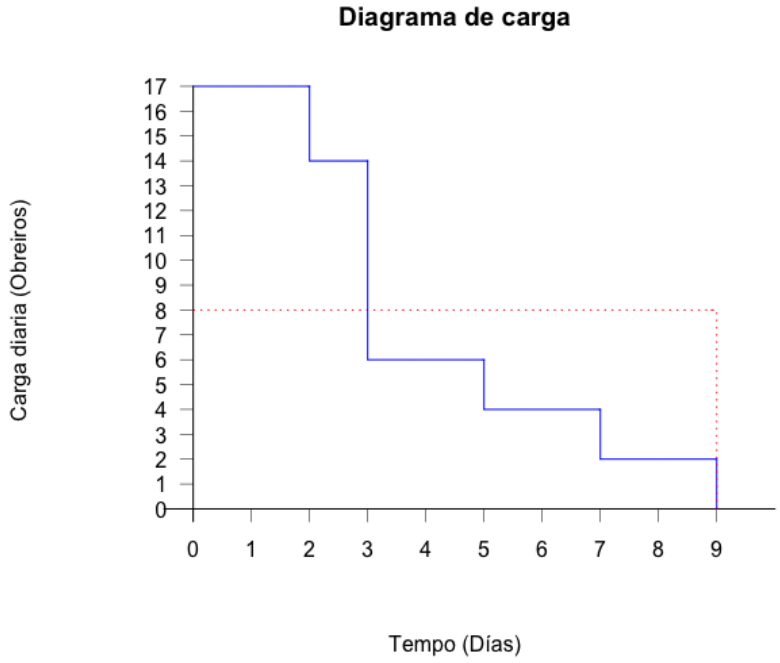


Figura 3.4: Novo diagrama de carga

Agora, na Figura 3.2 vemos que a actividade que ten unha data de finalización temperá máis avanzada é G, e esta actividade ten unha folgura de 2 días. Polo tanto, comezamos atrasando un día esta actividade, e vexamos se ten algún tipo de repercusión na suma dos cadrados das cargas (Neste caso, coa modificación que fixemos, esta suma vale 886). Ao atrasar un día obtemos o novo valor de 834, o que supón unha diminución na varianza total das cargas. Como a folgura é de 2, aínda podemos atrasar un día máis, obtendo o valor de 782, o que implica un descenso de case 100 unidades do orixinal.

Segunda fase: Agora, entre todas as actividades non críticas, pero excluíndo a actividade que xa estudamos no punto anterior, volvemos a elixir a que ten a data de finalización máis avanzada. Unha vez achada, aplicamos outra vez o mesmo paso da etapa 1. Con esta forma de proceder, continuamos ata que chegamos á actividade que ten a data de finalización temperá máis atrasada. Se nos atopamos no caso de que dúas ou máis actividades teñen a mesma data temperá de finalización, priorizamos a que teña unha folgura maior. Deste xeito, no noso exemplo, pasamos á actividade E, que ten unha folgura de 4. Atrasámola os 4 días, sen cambios na suma dos cadrados da carga. Agora pasamos á actividade B, cunha folgura de 4. Se a atrasamos os 4 días, chegamos a unha situación na que o valor que se toma é 594, case 200 menos que na anterior iteración. Seguimos co proceso, e chegamos á última actividade, a F, cunha folgura de 5. Agora observamos que facendo o cambio que fagamos, non imos obter ningunha mellora, polo tanto, freamos neste paso.

Terceira fase: Unha vez chegamos á actividade con data temperá de finalización máis atrasada, volvemos a iniciar un novo ciclo de iteracións. Deteremos o proceso cando rematado un ciclo, non poidamos diminuír a suma dos cadrados das cargas. Neste caso, xa remataríamos no primeiro ciclo. Como podemos observar na seguinte figura, hai unha melloría na nivelación dos recursos con respecto a 3.4, pero non chegamos ao .ªlisado"óptimo ao que aspiramos.

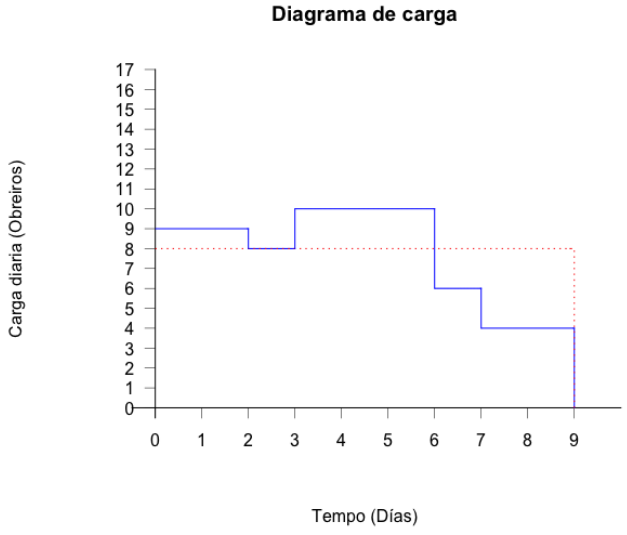


Figura 3.5: Novo diagrama de carga

Hai que ter en conta, que o algoritmo de Burgess-Killebrew, ao ser un método heurístico,

non nos indica se chegamos a unha solución óptima, exceptuando os casos nos que obtemos o alisado óptimo (un diagrama de cargas rectangular). A principal vantaxe desta técnica, é a súa facilidade de procesar nun ordenador, e o seu principal inconveniente, que está pensada para a nivelación dun único recurso. No Apéndice Código de R, temos outra solución alternativa empregando o paquete de R "ProjectManagement". Como comentamos antes, ao estar aplicando un algoritmo heurístico para a resolución do problema, é razoable que obteñamos outra solución distinta á que achamos a man, pero non por iso deixa de ser correcta.

3.3.2. Algoritmo de Wiest-Levy

No caso de asignación de recursos, presentaremos o algoritmo de Wiest-Levy. Apoiarémonos no mesmo exemplo que empregamos na explicación do apartado 3.2. A idea principal deste algoritmo é ir asignando os recursos etapa a etapa. Empezamos por programar todas as actividades que se poidan executar na primeira etapa, sempre que non superen as dispoñibilidades dos recursos, e a continuación procedemos do mesmo xeito coas demais etapas de forma sucesiva. Cando a carga supera a dispoñibilidade dun recurso, daremos prioridade de realización ás actividades coa menor folgura.

Comezamos a aplicación do algoritmo programando a execución de actividades no primeiro día do proxecto. Nesta etapa, temos que executar catro actividades: A, B, F e G, e a carga deste día (14) supera o límite de 11 obreiros, polo que non poderemos executar as catro á mesma vez. Desas actividades, a que maior folgura total ten é a F (5), polo que será a que atrasaremos en primeiro lugar. Se a atrasamos os cinco días, obtemos unha carga diaria nos dous primeiros días de 11, pero seguimos tendo no terceiro día, unha carga de 12, superior ao límite. Se a atrasamos menos deses cinco días, faría que se superase o límite nos días 4 e 5, o que non solucionaría o problema.

Seguimos aplicando o algoritmo, asignando o recurso da man de obra no terceiro día, que é de 12. Están programadas para este día as actividades A, B e G. Como a actividade A é crítica, e a G ten 2 días de folgura, atrasaremos a actividade B, sendo a que máis marxe ten. Aprazándoa 3 días, xa obtemos unha solución na que non superaríamos o límite de 11 obreiros en ningún dos días do proxecto. Podemos observar como quedaría o novo calendario de execución na Figura 3.6:

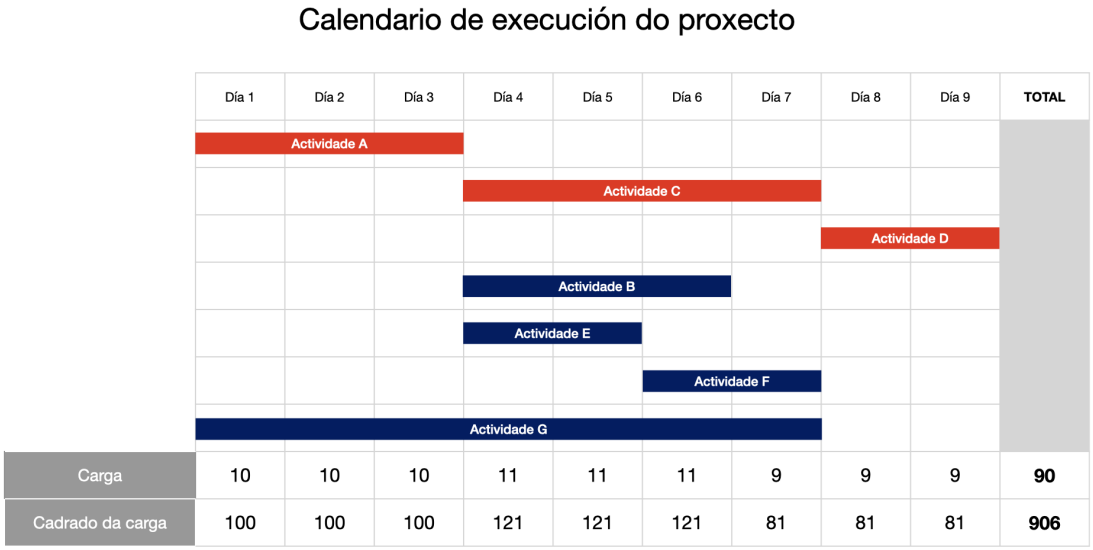


Figura 3.6: Novo calendario de execución

Agora ben, podería darse o caso, no que para poder cumprir co límite que nos impoñen os recursos limitados, incorramos en atrasos na duración total do proxecto. Esta situación pode darse no caso no que, para reducir a carga diaria, tiveramos que aprazar actividades máis aló da súa folgura. Cabe dicir que o algoritmo exposto neste apartado non contempla o fraccionamento na execución das actividades, o que quere dicir que toda actividade iniciada ten que ser completada.

Como comentamos anteriormente, o que estamos facendo é aplicar un algoritmo heurístico, o cal non nos asegura que a solución á que cheguemos é a óptima. Obtemos unha solución razoablemente boa nun menor tempo de computación, pero poden existir outras solucións que sexan correctas. No Apéndice temos o código de R dun programa que dá unha solución ao problema de asignación de recursos usando o paquete "ProjectManagement". Como no caso anterior, se as solucións non son iguais, é debido a que non resolvemos o problema cun algoritmo exacto. Co noso algoritmo heurístico, obtemos unha solución aceptable, pero non quere dicir que sexa única.

Capítulo 4

Aplicación do método PERT a un caso real

Como peche a este traballo, e coa intención de xustificar a utilidade das técnicas que fomos presentando ao longo do mesmo, neste capítulo aplicarémolas a un caso real. Estudaremos e avaliaremos o caso do proceso de obtención da uva nunha bodega D.O. Rías Baixas, no período que comprende dende o primeiro día despois da vendima do ano anterior, ata o día antes do comezo da vendima do ano seguinte. Os datos das actividades e dos tempos foron facilitados pola propia bodega. Con este capítulo tentaremos exemplificar a aplicación dos métodos expostos, e ver como enfocar a formulación do problema.

En primeiro lugar definimos as actividades correspondentes a todo o proxecto:

- A - Picas as ramas que non se poden aproveitar o ano seguinte.
- B - Baixar as vides.
- C - Podar as vides.
- D - Atar as viñas á condución.
- E - Retirar as cepas mortas.
- F - Repoñer con cepas novas.
- G - Esperar a que broten as parras.
- H - Atar as cepas novas ao poste para endereitalas.
- I - Subir as ramas á condución.

- J - Despuntar e limpar os corredores.
- K - Poda en verde da planta.
- L - Últimos tratamentos da uva.
- M - Esperar a que a uva estea lista para a súa recollida.
- N - Tratar a vide de enfermidades.
- O - "Fertirego".

Desta lista de actividades, despois de analizar cada caso particular, decidimos non incluír as actividades N e O no estudo, xa que a súa realización é simultánea ao resto de actividades, e os seus tempos non afectan en ningún caso á duración do proxecto.

A táboa cos tempos de execución das actividades é:

Actividade	A	B	C	D	E	F	G	H	I	J	K	L	M
Tempos	25	25	30	20	5	5	50	7	5	15	15	15	21

Observación 4.1. O tempo de execución vén expresado en días.

E o seu cadro de prelacións é:

Actividade	**Precedente**
A*	-
B*	A
C*	B
D*	C
E	D
F*	E
G	F
H	G
I*	H
J	I
K	J
L	K
M	L

Cadro 4.1: Táboa de prelacións

Actividade	**Demora**
A	-
B	15 días despois de A
C	10 días despois de B
D	15 días despois de C
E	-
F	2 días despois de E
G	-
H	-
I	3 días despois de H
J	-
K	-
L	-
M	-

Cadro 4.2: Táboa coas demoras

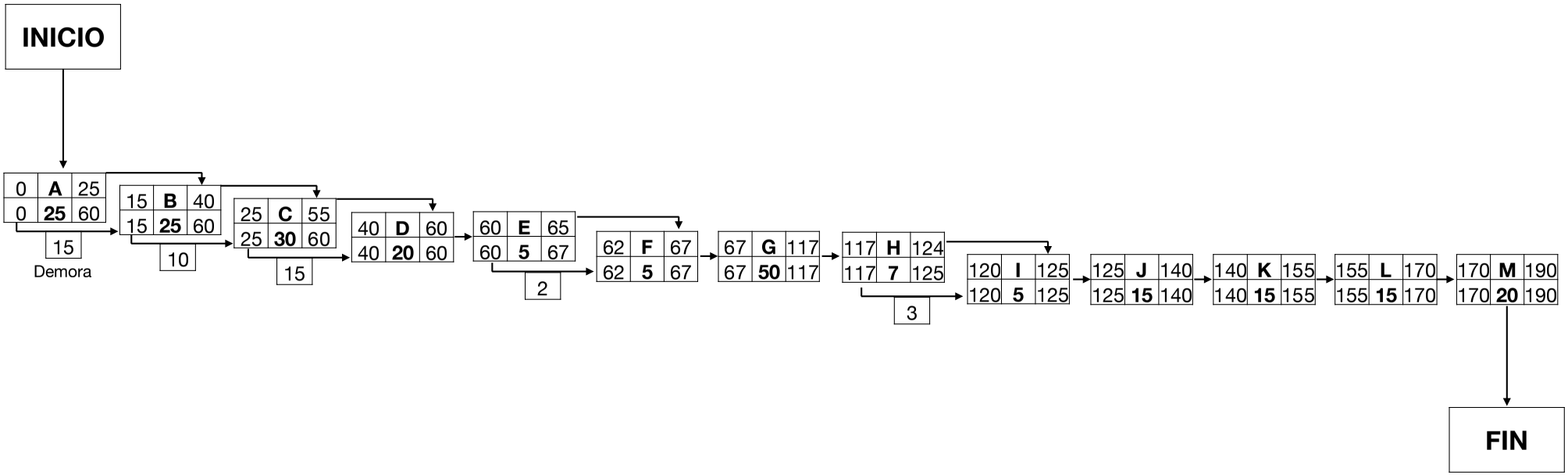
Observación 4.2. As actividades sinaladas cun asterisco, son tarefas que seguen o orde de precedencia indicado, pero que poden coincidir no tempo. Isto quere dicir, que logo dunha demora, pode comezar a actividade posterior antes de que remate a precedente. Por exemplo, non ten que rematar A para que comece B. No cadro da dereita sinalamos os casos nos que existe este solapamento.

Agora teríamos que construír o seu grafo PERT. As técnicas que estudamos neste traballo son métodos no formato de "actividades sobre arcos" (AoA), como presentamos no apartado 1.1. É un formato que funciona moi ben nun contexto de relación de precedencia, i.e., unha actividade ten que terminar antes de que comecen as seguintes, pero é conservador noutras situacións que se dan na práctica como a deste caso. O problema que xorde nesta situación en concreto, é que temos varias actividades que aínda que teñen certa orde de execución, simultanéanse no tempo nalgún intre, e polo tanto, fai complicada a representación no formato AoA. Entón, a solución máis axeitada que atopamos, foi intentar enfocalo cun método do formato de "actividades sobre nodos" (AoN).

Recordamos do primeiro capítulo, que a principal diferencia entre o formato AoA e AoN, é a construción do grafo. No formato AoN, ou tamén chamado método ROY[1], os nodos representan as actividades, e os arcos indican a orde na que deben executarse. A principal vantaxe de empregar o método ROY é a de poder expresar as prelacións dun xeito que se pode adaptar a un contexto como é o deste caso, o de solapamento de actividades. Aínda que poderíamos representalo cun grafo PERT, esta é unha boa situación para mostrar o formato ROY, xa que a representación é máis sinxela.

Representemos agora o diagrama en formato AoN do proxecto:

[1] Véxase Romero, C., *Técnicas de programación y control de proyectos* páx. 105



LENDA:

Early Start	**Actividad**	Early Finish
Last Start	**Duración**	Last Finish

Como podemos observar na lenda da Figura **??**, representamos os nodos como unha "caixa" con 6 espazos. En letra máis grosa, nos ocos do medio, escribimos a actividade, e debaixo a súa duración. A continuación explicaremos o cálculo dos marxes que ten cada tarefa para realizarse.

4.1. Tempos *early* e *last*

Nesta sección explicaremos o algoritmo mediante o cal obtemos os tempos "máis pronto posible" e "máis tarde permisible", que teñen un papel moi semellante ao que xogan no caso do PERT. Como xa comentamos, os nodos están divididos en 6 ocos. Só nos quedaría por asignarlle a información aos espazos das esquinas, posto que xa explicamos os do medio. Na fila de arriba escribiremos os tempos *early*, e na fila de abaixo os tempos *last*.

4.1.1. Tempos *early*

Comezamos calculando os tempos *early* mediante o proceso que se coñece como *Forward Pass Calculation*[2], xa que procederemos cara adiante dende a primeira actividade.

En primeiro lugar, asignámoslle á primeira tarefa un ES (*Early Start*) de 0. Para o cálculo do EF (*Early Finish*) sumamos a duración da actividade, que no caso de A é 25, obtendo o primeiro EF igual a 25. A partir de agora teremos que distinguir dous casos. Se a actividade seguinte comeza cun retardo e de forma simultánea, ou se comeza cando remate a anterior. No caso de B, como podemos observar, comeza cunha demora de 15 días, polo tanto o seu ES será 15 días despois de que comece A, ou, formalmente:

$$ES_B = ES_A + \text{DEMORA} = 0 + 15 = 15$$

Para o cálculo de EF_B sumamos a duración da actividade ao seu Early Start como no caso anterior:

$$EF_B = ES_B + DURACIÓN\ B = 15 + 25 = 40$$

Procedemos da mesma forma no caso de que exista solapamento. Cando non, é moito máis sinxelo, como no caso de D e E. Ata que remate D, non comeza D, polo tanto, podemos escribilo formalmente como:

$$EF_D = ES_E$$

[2]Véxase *Construction Equipment Management for Engineers, Estimators, and Owners* Gransberg, D., Cap. 7

Imos operando así ata chegar á última actividade, M. Obtemos un tempo $EF = 190$, e este será o obxectivo a conseguir, xa que é a duración do proxecto.

4.1.2. Tempos *last*

Para o cálculo dos tempos last, seguiremos o *Backward Pass Calculation*, no que procederemos dende a última actividade, cara atrás.

O primeiro paso é calcular o Last Finish (LF) de M, asignándolle o EF do proceso anterior, porque como dixemos, é a duración do proxecto. Procedemos cara atrás, restándolle ao LF a duración da actividade en cuestión e obtendo o Late Start (LS). Escribimos:

$$LS_M = LF_M - \text{DURACIÓN DE M} = 190 - 20 = 170$$

Como antes, nos casos nos que non hai simultaneidade, o cálculo é sinxelo. Á actividade precedente asignámoslle un LF igual ao LS da seguinte, neste caso, $LF_L = LS_M$, e procedemos do mesmo xeito ata que nos atopamos co caso das actividades H e I. Este sería o primeiro caso que nos atopamos con demora. O tempo LF_H será igual ao LF_I, e para o cálculo de LS_H temos que escoller o mínimo entre:

$$LS_H = min \begin{cases} LF_I - \text{DURACIÓN DE H} & = 125 - 7 = 118 \\ LS_I - \text{DEMORA} & = 120 - 3 = \mathbf{117} \end{cases}$$

E desta forma obtemos o Late Start de H. Para o resto de actividades con demora procedemos do mesmo xeito. Outro exemplo é o caso de C e D:

$$LF_C = LF_D = 60$$

E para o Late Start:

$$LS_C = min \begin{cases} LF_D - \text{DURACIÓN DE C} & = 60 - 30 = 30 \\ LS_D - \text{DEMORA} & = 40 - 15 = \mathbf{25} \end{cases}$$

Na Figura **??** temos calculados todos os tempos de todas as actividades.

4.2. Folguras

Como no método PERT, calcularemos as folguras que teñen as actividades para realizarse. Primeiro, calcularemos a **folgura total**, que nos indica o tempo que podemos atrasar unha actividade sen que esta altere a duración total proxecto. Neste caso é de vital importancia axustarse o máximo posible á duración que temos, xa que ao estar condicionados

por un proceso natural, temos que tentar non atrasar o proxecto.

Para o cálculo da folgura total (H^T), usamos a seguinte fórmula:

$$H^T_{\text{Actividade}} = LF_{\text{Actividade}} - ES_{\text{Actividade}} - \text{DURACIÓN} \tag{4.1}$$

No caso das actividades nas que non hai solapamento con outras, a H^T será igual a cero. No resto, calculámolas como indica a fórmula 4.1:

Actividade	A	B	C	D	E	F	G	H	I	J	K	L	M
Folg. Total	35	20	5	0	2	0	0	1	0	0	0	0	0

Por exemplo, se "gastamos" toda a folgura da actividade A, a actividade B tería que comezar ao mesmo tempo que A, se queremos que a duración do proxecto siga sendo de 190. É un dato interesante, xa que á dirección do proxecto pode interesarlle, por motivos de loxística, atrasar unha actividade porque non dispón de traballadores para facela no momento no que estaba programada nun principio.

A **folgura libre** (H^I), é o tempo que podemos atrasar unha actividade sen que afecte ás folguras das actividades seguintes. Obtémolo coa fórmula:

$$H^I_{\text{Actividade}} = ES_{\text{Actividade seguinte}} - \text{DEMORA} - ES_{\text{Actividade}} \tag{4.2}$$

Neste caso, todas as folguras libres valen cero.

4.3. Calendario do proxecto

Nesta sección, representaremos graficamente o calendario que se lle entregaría á dirección do proxecto, xa que nel mostramos dun xeito moito máis visual, os datos que fomos calculando anteriormente.

A duración do proxecto será de 190 días, pero son días laborables, entón primeiro debemos traducilo a días naturais, axudándonos dun calendario, para poder traballar con datas reais. Agora teríamos que escoller a data de comezo do proxecto, e polo que nos comentan na bodega, adoitan comezar a mediados/finais de decembro. Este ano comezaron o 18 de decembro, polo que escollemos esa data, e así podemos comprobar que as nosas contas correspóndense co proxecto real. A continuación representamos o calendario do proxecto que calculamos, comezando o 18 de decembro, e finalizando o 16 de setembro, data aproximada de comezo da nova vendima.

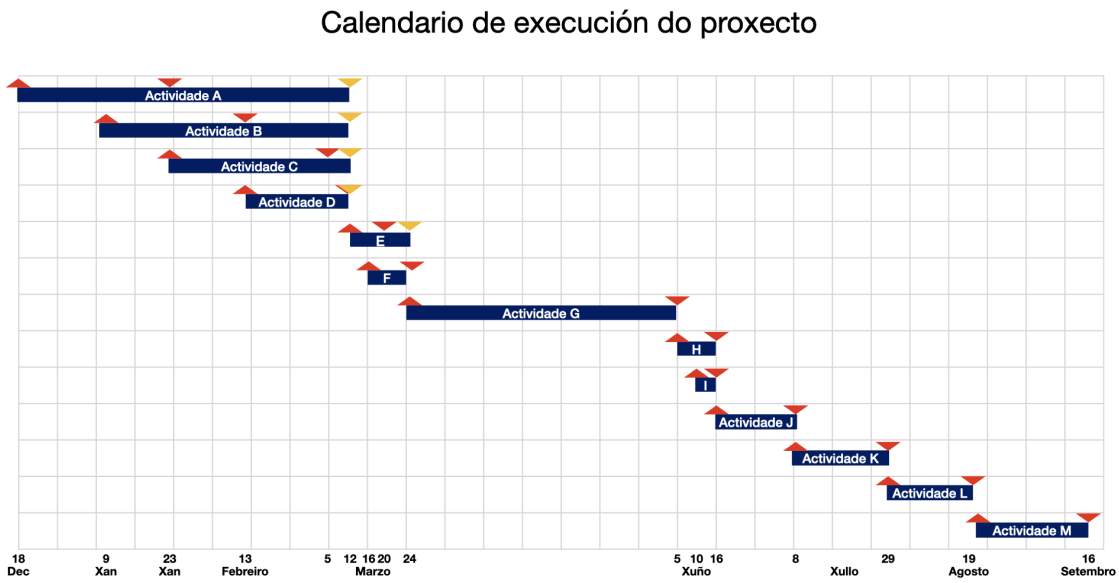


Figura 4.1: Calendario de execución do noso exemplo

Observación 4.3. Como vimos no diagrama en formato AON da páxina 44, os tempos Early Start e Last Start coinciden en todas as actividades, polo que están representados cun triángulo vermello nos ambos casos. No caso do Early Finish representámolo cun triángulo vermello invertido, e o Last Finish, cun triángulo amarelo invertido. No caso que coinciden estas dúas datas, só aparece un triángulo vermello invertido.

A utilidade de poder representar un calendario coma este, é que non é necesario explicar todo o proceso de cálculo que existe detrás para poder entendelo, de xeito que entregándolle isto á dirección dun proxecto, poidan comprender, simplemente cunha ollada, cales son as actividades coas que non se poden demorar, e cales teñen unha folgura de marxe para a súa realización.

Conclusións

Ao longo deste traballo de fin de grao, fixemos un primeiro contacto ás técnicas de programación e planificación de proxectos, expoñendo as cuestións que van xurdindo ao longo da formulación dos diferentes métodos, e atacándoas mediante as ferramentas que temos na Investigación Operativa.

No primeiro capítulo, repasamos nocións básicas da teoría de grafos para poder comezar coa construción dun grafo PERT e a dos seus elementos. Definimos que propiedades debe cumprir, e estudamos o problema da asignación dos tempos ás actividades. Supuxemos que as duracións seguían unha distribución tipo beta, xa que é a máis aceptada hoxe en día, pero tamén vimos que existen autores que asumen outra, como a triangular, e que na práctica podemos atoparnos con distribucións doutro tipo.

No segundo capítulo, atopámonos co caso da programación de proxectos a custo mínimo, de especial interese na práctica, xa que é o contexto no que se adoita traballar na vida real. Estudamos a relación entre a duración e o custos das actividades dun proxecto, e enfocamos o problema como un problema de programación lineal paramétrica. Debido ao alto custo computacional que esixe resolver este problema, explicamos un algoritmo heurístico que nos pode dar unha solución nun tempo moito máis razoable, e aplicámolo nun caso práctico.

No terceiro capítulo volvemos a atoparnos con outras situacións cun interese esencialmente práctico polo mesmo motivo que no caso anterior, pero que tamén ten a súa relevancia no ámbito académico, debido á complexidade de programar un proxecto con varias restricións. Neste capítulo falamos da nivelación e asignación de recursos, deixando exposto o problema de programación matemática que temos en cada caso, e dando un algoritmo heurístico que nos axuda a resolvelo de forma máis sinxela.

Finalmente rematamos cunha aplicación práctica a todo o exposto. É un caso real no

que nos enfrontamos ao estudo dun proxecto, no que tivemos que solicitar os datos que necesitábamos, e buscar a forma de expoñelo de forma axeitada. Nun principio a idea era analizalo mediante o método PERT, pero vendo que tiñamos un método como o ROY, que se axustaba moito máis ao que requiría a situación, decidimos introducir unha técnica no formato de actividades sobre nodos.

Este TFG é un primeiro contacto coas técnicas de programación de proxectos, abrindo varios frontes nos que podería continuarse o traballo, dende os distintos problemas de programación lineal paramétrica que se expuxeron, á predición dos tempos, ou o estudo das técnicas expostas nun contexto aleatorio.

Bibliografía

[1] Romero, C., *Técnicas de programación y control de proyectos*, 2da ed., Ediciones Pirámide, Madrid, 1983.

[2] Escudero, L. *Asignación óptima de recursos*, Ediciones Deusto, Bilbao, 1977.

[3] Kaufmann, A., e Desbazeille, G., *Método del camino crítico, Aplicación a los programas de ejecución de trabajos del método PERT y a la optimización de sus costes*, 2da ed., Sagitario, Barcelona, 1971.

[4] Vanhoucke, M., *Project Management with Dynamic Scheduling*, 2nd ed., Springer-Verlag, Berlin, 2012 .

[5] Yu, L., *Aplicaciones Prácticas del PERT y CPM*, 6a ed., Ediciones Deusto, Bilbao, 1983.

[6] Torleif H. and Magne J., *Time Predictions. Understanding and Avoiding Unrealism in Project Planning and Everyday Life*, Simula, 5, Springer, 2018.

[7] Miklós H., *Network Scheduling Techniques for Construction Project Management*, Springer, 1997.

Código de R

Código da linguaxe de programación R empregado na resolución dos exemplos. Empregando o paquete "ProjectManagement"

No seguinte programa obtemos os tempos early e last do proxecto, así como as folguras, e un diagrama das actividades do Exemplo 1.8:

```
matprec <-rbind(c(0,1,0,0,0,0,0),c(0,0,1,0,1,0,0),c(0,0,0,0,0,0,1),
c(0,0,0,0,1,0,0),c(0,0,0,0,0,1,0),c(0,0,0,0,0,0,1),c(0,0,0,0,0,0,0))
duracion <- c(10,3,15,5,2,1,2)

schedule.pert(prec1and2 = matprec, duration=duracion)
```

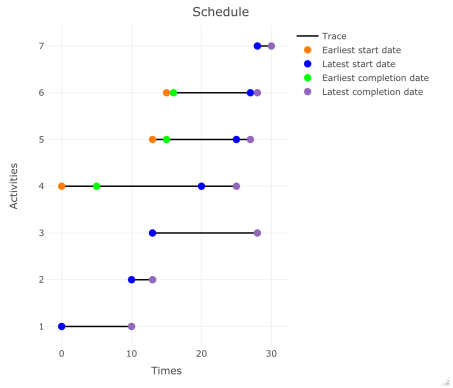


Figura 2: Cronograma do noso exemplo

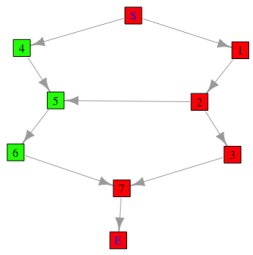


Figura 3: Grafo no formato AON do exemplo

No seguinte programa obtemos os tempos early e last:

```
matprec <- rbind(c(0,1,0,0,0,0,0),c(0,0,1,0,1,0,0),c(0,0,0,0,0,0,1),
c(0,0,0,0,1,0,0),c(0,0,0,0,0,1,0),c(0,0,0,0,0,0,1),c(0,0,0,0,0,0,0))
duracion <- c(10,3,15,5,2,1,2)
```

```
earlytimes<-early.time(prec1and2=matprec, duration = duracion)

lasttime <- last.time(prec1and2=matprec, duration = duracion,
early.times = earlytimes)
```

A resposta:

```
> earlytimes
0 10 13  0 13 15 28

> lasttime
10 13 28 25 27 28 30
```

Programa que resolve o problema do MCE, do exemplo 2.1:

```
tempos<-c(9,10,7,8,12,15,7) #Duracion das actividades
tempos.min<-c(4,5,4,5,8,9,5) #Tempos tope
custos.unit<-c(2,4,3,4,5,2,1) #Custos por reducion unitaria
matriz<-rbind(c(0,0,1,0,0,0,0),c(0,0,0,1,0,0,0),c(0,0,0,0,0,0,1),
c(0,0,0,0,1,1,0),c(0,0,0,0,0,0,1),c(0,0,0,0,0,0,0),c(0,0,0,0,0,0,0))

for (duracion in 23:37) {
A<-mce(duration=tempos, minimum.durations=tempos.min, prec1and2=matriz,
activities.costs=custos.unit, duration.project= duracion);
b<-colSums(A);
custo.total <- b[2]
cat("Custo total:",as.numeric(custo.total),"\n")
}
```

Obtendo para cada tempo, o custo mínimo asociado:

```
The project duration is 23
Custo total: 58
The project duration is 24
Custo total: 51
The project duration is 25
Custo total: 44
The project duration is 26
Custo total: 39
```

```
The project duration is 27
Custo total: 34
The project duration is 28
Custo total: 30
The project duration is 29
Custo total: 26
The project duration is 30
Custo total: 22
The project duration is 31
Custo total: 18
The project duration is 32
Custo total: 14
The project duration is 33
Custo total: 10
The project duration is 34
Custo total: 6
The project duration is 35
Custo total: 2
The project duration is 36
Custo total: 1
The project duration is 37
Custo total: 0
```

Como podemos observar, os resultados coinciden cos obtidos facendo uso do algoritmo de Ackoff-Sasieni

No seguinte programa achamos unha solución para o problema de nivelación de recursos da Sección 3.3.1:

```
duracion<-c(3,3,4,2,2,2,7)
recursos<-c(6,6,2,2,2,3,2)
mat.prec <- rbind(c(0,0,1,0,0,0,0),c(0,0,0,0,1,0,0),c(0,0,0,1,0,0,0),
c(0,0,0,0,0,0,0),c(0,0,0,0,0,0,0),c(0,0,0,1,0,0,0),c(0,0,0,0,0,0,0))

levelling.resources(duration=duracion,prec1and2=mat.prec,
resources=recursos, int=1)
```

Obtendo:

```
Early times =
[1] 0 3 3 7 7 5 2
Resources by period=
[1]  6  6  8 10 10 13  7  6  6
```

Esta é unha solución alternativa á obtida co algoritmo de Burgess-Killebrew na sección 3.3.1.

O seguinte programa dá unha solución ao problema da asignación de recursos da sección 3.3.2:

```
duracion<-c(3,3,4,2,2,2,7)
recursos<-c(6,6,2,2,2,3,2)
mat.prec <-rbind(c(0,0,1,0,0,0,0),c(0,0,0,0,1,0,0),c(0,0,0,1,0,0,0),
c(0,0,0,0,0,0,0),c(0,0,0,0,0,0,0),c(0,0,0,1,0,0,0),c(0,0,0,0,0,0,0))
mat.prec
recursos.max<-11

resource.allocation(duration=duracion,prec1and2=mat.prec,
resources=recursos,max.resources=recursos.max,int=1)
```

Obtendo a solución:

```
Project duration =
[1] 9
Early times =
[1] 0 3 3 7 6 0 0
Resources by period =
[1] 11 11  8 10 10 10  6  4  2
```

Esta é unha solución alternativa á obtida co algoritmo de Wiest-Levy na sección 3.3.2.