USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

FACULTADE DE MATEMÁTICAS

**Traballo Fin de Grao**

# Introducción a la programación dinámica

Lidia López Fernández

2020/2021

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

GRAO DE MATEMÁTICAS

**Traballo Fin de Grao**

# Introducción a la programación dinámica

Lidia López Fernández

Julio, 2021

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

# Trabajo propuesto

| |
|---|
| **Área de Coñecemento:** Estadística e investigación operativa |
| **Título:** Introducción a la programación dinámica |
| **Breve descrición do contido** |
| En este trabajo se estudiará la técnica de programación dinámica para la resolución de problemas de optimización. Se caracterizarán los problemas de programación dinámica y se ilustrará con diversas aplicaciones prácticas. |

# Índice general

## Resumen

La programación dinámica es un método que consiste en simplificar un problema de programación matemática complejo en subproblemas más simples, de manera recursiva, de forma que, resolviendo estos últimos, podamos hallar una solución óptima para el problema original.
En este trabajo veremos una introducción a la programación dinámica basándonos en diferentes casos prácticos, sobre los que veremos como aplicar esta técnica, lo que nos permitirá identificar las características comunes y las diferencias de los problemas que pueden ser resueltos mediante programación dinámica.

## Abstract

Dynamic programming is a method that consists of simplifying a complex mathematical programming problem into simpler subproblems, recursively, so that, by solving them, we can find an optimal solution for the original problem.
In this document we will see an introduction to dynamic programming based on different practical cases, on which we will see how to apply this technique, which will enable us to identify common characteristics and differences in problems that can be solved through dynamic programming.

# Introducción

Al terminar la Segunda Guerra Mundial, se empezó a considerar que había un gran número de actividades que podían ser clasificadas como problemas de decisión divisibles en subproblemas en diversos campos como ingeniería, economía, industrial y militar. Pronto se vió que estos problemas matemáticos que iban surgiendo sobrepasaban los límites convencionales del análisis y que se necesitaban nuevos métodos para una resolución exitosa de estos.
Las técnicas clásicas de cálculo eran a veces útiles en estas nuevas áreas, pero estaban claramente limitadas en alcance y versatilidad, y en ocasiones eran ineficientes en lo que se refiere a proporcionar respuestas numéricas.

La aceptación de estos hechos llevó a la creación de una serie de nuevos métodos y teorías matemáticas. Entre estos se halla la técnica de la programación dinámica, basada en el principio de optimalidad, con un gran potencial para el campo de las computadoras digitales, que en aquel momento se encontraba en plena expansión. Esta fue desarrollada por el matemático Richard E. Bellman en 1953, y posteriormente en su libro [BeDr].

El principio de optimalidad de Bellman nos dice que, dadas las decisiones óptimas que hallamos tomado en cada una de las etapas en las que podemos dividir un problema, estas no dependerán de las decisiones óptimas que hayamos tomado en las etapas anteriores y que la solución global del problema se obtiene a partir de las soluciones de los subproblemas.

Basándonos en esto, podemos ir resolviendo de forma recursiva cada uno de los subproblemas (etapas) independientemente, los cuales serán más sencillos que el problema principal, para al final combinar todas las decisiones que hallamos tomado y hallar así la solución óptima del problema inicial.
Se puede ver entonces una de las principales ventajas de esta técnica. Al resolver un problema localmente, cualquier imprevisto que surja en la vida real y nos haga añadir o eliminar opciones del problema implicará menos cálculos que en el caso de resolverlo globalmente,

pues entonces tendríamos que empezar de cero, y de esta forma solo tendremos que modificar el subproblema afectado.

Por otro lado, esto puede llegar también a ser una desventaja, ya que este número de cálculos irá aumentando según vayamos añadiendo variables o etapas al problema. Es fácil ver que llegará un punto donde el número de cálculos se haga rápidamente inviable, lo que es conocido como la "maldición de la dimensión". Aun así, gracias a la potencia de las computadoras, en la actualidad es un problema de menor importancia.

En cuanto a estos problemas, es la práctica la que nos va a permitir determinar si se pueden resolver mediante programación dinámica, así como una forma concreta de aplicarla, ya que según los problemas que tengamos habrá que actuar de distintas formas. Por eso, el aprendizaje de esta técnica exige la realización de diversos casos prácticos, como se puede ver en los libros dedicados a la programación dinámica, en los cuales nos basaremos para este trabajo.
Por ejemplo, podemos clasificar un problema según las etapas en las que podamos descomponerlo. Si lo hacemos en un número finito de etapas, lo denominaremos un problema de horizonte finito. En cambio, si resulta un número infinito de etapas, será un problema de horizonte infinito. Nos centraremos exclusivamente en problemas del primer tipo.

También podemos clasificar la programación dinámica en dos tipos: la determinística y la probabilística. La diferencia entre ambas será que en la primera no habrá incertidumbre en los resultados, mientras que en la segunda habrá una cierta incertidumbre proporcionada por una distribución de probabilidad. En este trabajo nos centraremos en la programación dinámica determinística, pero también presentaremos un ejemplo de programación dinámica probabilística.

En este trabajo presentaremos una introducción a la técnica de la programación dinámica, ilustrándola con diversos ejemplos prácticos, que nos harán comprender más fácilmente su funcionamiento.
Para esto, nos basaremos en los libros [BeDr] y [Be12], siguiendo la estructura de [HiLi].

# Capítulo 1

# El problema básico

Nosotros vamos a aplicar la técnica de programación dinámica en los problemas de programación matemática.

En un problema de programación matemática tenemos los siguientes componentes:

- **Variables de decisión** $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$.
- **Función objetivo** $f \colon \mathbb{R}^n \longrightarrow \mathbb{R}$.
- **Restricciones**, condiciones que deben cumplir las variables de decisión.
  Pueden ser de dos tipos:
  - **Restricciones de desigualdad**, $g_i(x) \leq 0, i \in 1, \ldots, m$ (con $g_i \colon \mathbb{R}^n \longrightarrow \mathbb{R}$).
  - **Restricciones de igualdad**, $h_j(x) = 0, j \in 1, \ldots, l$ (con $h_j \colon \mathbb{R}^n \longrightarrow \mathbb{R}$).

La formulación de un problema de programación matemática será de la forma:

$$\begin{aligned} &\text{minimizar } f(x) \\ &\quad \text{sujeto a } g_i(x) \leq 0 \quad i = 1, \ldots, m \\ &\qquad\qquad h_j(x) = 0 \quad j = 1, \ldots, l. \end{aligned}$$

Llamaremos *región factible* al conjunto de puntos $(x_1, \ldots, x_n) \in \mathbb{R}^n$ en los que se verifican las restricciones impuestas en el problema.

Cuando resolvemos estos problemas, obtenemos la solución óptima "global", es decir, el valor de la función objetivo por los valores óptimos $(x_1, \ldots, x_n)$ obtenidos "simultáneamente".
Pero, en algunas ocasiones, podemos resolver el problema "de manera secuencial", es decir,

considerando $n$ subproblemas relacionados entre si, obtenidos a partir del problema general, en los que solo consideraremos una variable de decisión $x_i \in \mathbb{R}^n,\ i : 1, \ldots, n$ y, una vez resueltos, podemos "enlazar" las soluciones de estos subproblemas para obtener la solución óptima del problema completo. Este enfoque se conoce como *Resolución de un problema en varias etapas.*
En este procedimiento se basa la programación dinámica.

## 1.1. Elementos principales

Como ya hemos indicado, la programación dinámica es un método para resolver problemas de programación matemática que puedan dividirse en subproblemas relacionados entre sí de forma que, resolviéndolos, podemos hallar una solución óptima para el problema original.

Los elementos principales para definir un problema de programación dinámica son:

- **Etapas** $i = \{1, 2, \ldots, n\}$. En cada una de las etapas tenemos uno de los subproblemas en los que dividimos el problema original. Podemos encontrar problemas con un número de etapas finitos o infinito. En este trabajo nos centraremos en los problemas de horizonte finito, aquellos con un número finito de etapas.

- **Variables de decisión** $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$. Estas representan las decisiones que tomaremos. Cada $x_i$ es la variable de decisión que tomaremos para llegar a la etapa $i + 1$.

- **Estados** $s = (s_1, \ldots, s_n) \in \mathbb{R}^n$. Los estados son las distintas condiciones posibles en las que se puede encontrar el sistema en cada etapa del problema, es decir, $s_i$ nos indica los posibles valores de la variable $x_i$.

- **Función objetivo** $f = (f_1, \ldots, f_n) \colon \mathbb{R}^n \longrightarrow \mathbb{R}$. Representa el coste o beneficio asociado a las variables de decisión. Esta será una función aditiva sobre el tiempo, por lo que tendremos una relación recursiva que identifica la decisión óptima para la etapa $i$, dada la política óptima para la etapa $i + 1$.

Ilustrando gráficamente esta estructura en la siguiente figura:

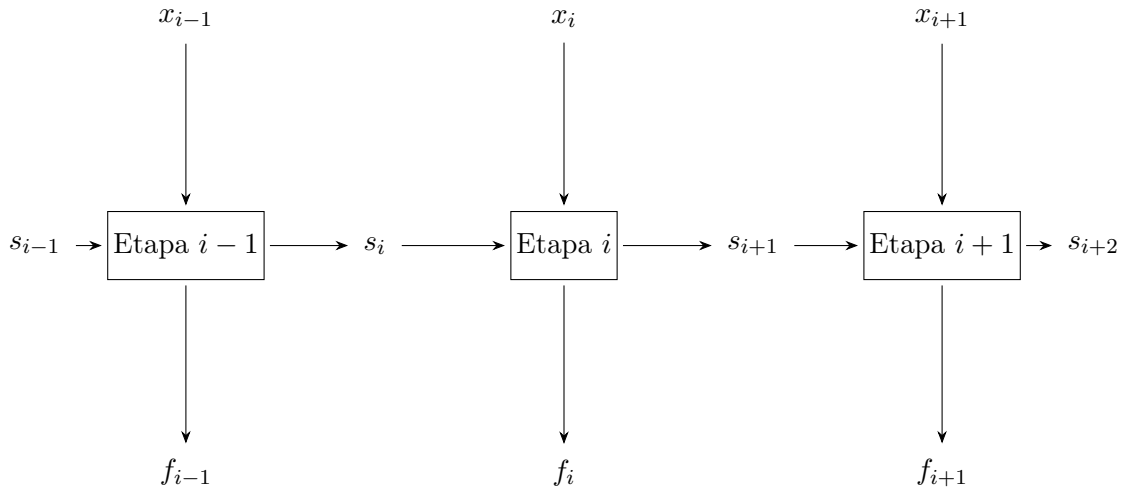



Figura 1.1: Estructura de la programación dinámica en la etapa $i$.

Una vez presentados estos conceptos, expondré un problema del camino más corto y veremos como resolverlo mediante programación dinámica.

**Ejemplo 1.1.** Este ejemplo es una variación del *Problema de la diligencia*, desarrollado por el profesor Harvey M. Wagner.
Presentamos a una joven que debe desplazarse desde A Coruña hasta Barcelona por motivos de trabajo, y la empresa decide que dispone de 15 días para hacer el viaje por carretera. Aprovechando el exceso de tiempo, la joven decide hacer paradas intermedias para hacer turismo en distintas ciudades de España. Aun así, quiere llegar en el menor tiempo posible al destino final, Barcelona, para así familiarizarse con la ciudad.
Dependiendo de en que ciudades intermedias decida pararse, estará en ella más o menos días, ya que algunas tienen más interés turístico que otras.
Representado como A el punto de partida (A Coruña), y como J el destino (Barcelona), ilustramos las rutas posibles a seguir:

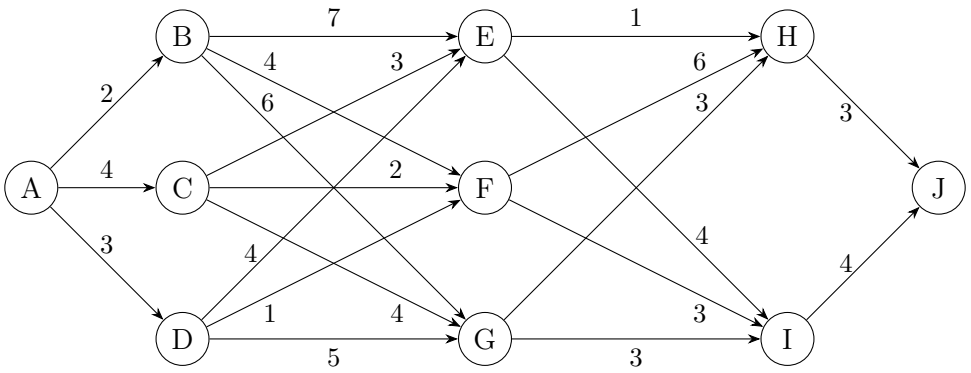



Figura 1.2: Posibles rutas a seguir por la joven.

Es evidente que se requieren cuatro etapas para viajar desde A hasta J, por lo que $n = 4$. En el grafo están representados los días que tardaremos desde que llegamos al estado $s_i$ hasta que llegamos al estado $s_{i+1}$, es decir, el coste de la decisión de ir de $s_i$ a $s_{i+1}$, denotados por $c(s_i, s_j)$.
Por tanto, la pregunta a responder será: ¿Cuál es la ruta que minimiza el coste del viaje?

Ahora se presentarán las dos estrategias a seguir para así poder resolver el problema usando programación dinámica.

## 1.2. Métodos backward y forward

La programación dinámica intenta mejorar la eficiencia del cálculo de problemas descomponiéndolos en subproblemas de menor tamaño, más fáciles de resolver. Esta está basada en el *Principio de Optimalidad* de Bellman:

«*Una política óptima tiene la propiedad de que cualquiera que sea el estado inicial y la decisión inicial, las decisiones restantes deben constituir una política óptima con respecto al estado resultante de la primera decisión.*» [Bellman,1962] [BeDr]

Entonces, volvemos a explicar la relación recursiva inducida por la función objetivo con más detalle. Tenemos entonces:

$n$ = número de etapas.
$i$ = etapa actual.

$s_i$ = estado actual de la etapa $i$.
$x_i$ = variable de decisión de la etapa $i$.
$f_i(s_i, x_i)$ = contribución a la función objetivo si el sistema se encuentra en el estado $s_i$ en la etapa $i$, la decisión inmediata es $x_i$, y en adelante se toman decisiones óptimas.
$x_i^*$ = solución óptima de $x_i$ dado $s_i$.
$f_i^*(s_i) = \text{máx}\{f_i(s_i, x_i)\} = f_i(s_i, x_i^*)$, o bien $f_i^*(s_i) = \text{mín}\{f_i(s_i, x_i)\} = f_i(s_i, x_i^*)$.

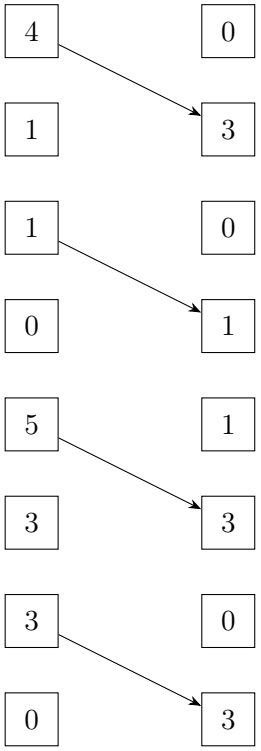
La relación recursiva será de la forma $f_i^*(s_i) = \text{máx}\{f_i(s_i, x_i)\}$ o bien $f_i^*(s_i) = \text{mín}\{f_i(s_i, x_i)\}$, donde $f_i(s_i, x_i)$ dependerá de $f_{i+1}^*(s_{i+1})$.
Procedemos a buscar la solución comenzando por el final y moviéndonos hacia atrás etapa por etapa para encontrar la decisión óptima para cada etapa hasta que llegamos a encontrar la decisión óptima desde la etapa inicial, la cual lleva directamente a una solución óptima del problema completo, gracias al *Principio de optimalidad*.
Queda así definido el **método "backward"** (método "hacia atrás") para la resolución de problemas de programación dinámica.

Cabe la posibilidad de que la relación recursiva se pueda definir en función de las etapas anteriores. Por tanto se empezará a resolver el problema comenzando al inicio y moviéndonos hacia delante etapa por etapa hasta llegar a la etapa final y obtener una solución para el problema completo. Esto es conocido como el **método "forward"** (método "hacia delante"), que suele ser menos usado.
Ahora pondremos un ejemplo para ilustrar estos dos métodos. Aunque no se trata de un problema de programación dinámica, nos servirá para explicar estos conceptos:

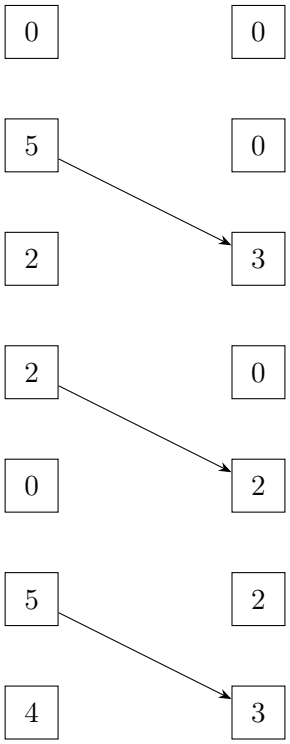
**Ejemplo 1.2.** En la película *La jungla de cristal 3*, el villano le va proponiendo diferentes retos al protagonista. Uno de ellos consiste en que, para poder evitar que una bomba explote, debe de realizar lo siguiente:
Disponiendo únicamente dos recipientes, uno con capacidad para 5 litros y el otro para 3, el protagonista tendrá que conseguir que en uno de ellos haya 4 litros de agua. Evidentemente tendrá que haber 4 litros en el recipiente con mayor capacidad.

**MÉTODO BACKWARD:** Partiremos del hecho de que en el recipiente grande debe de haber 4 litros de agua. En el recipiente pequeño puede haber tanto 3 litros como ninguno, para este ejemplo supondremos que está vacío. Podemos ver el proceso de resolución en la figura 1.3a.

**MÉTODO FORWARD:** Partiremos de que tenemos los dos recipientes vacíos y intentaremos llegar a tener 4 litros en el grande. Ilustramos el proceso de resolución en la figura 1.3b.




(a) Método backward.




(b) Método forward.

Figura 1.3: Resolución del problema mediante los dos métodos.

## 1.3. Ejemplo ilustrativo de la técnica de programación dinámica

Una vez hemos visto como funcionan las dos opciones, resolveremos el ejemplo 1.1 mediante los métodos backward y forward, tratándose ya de un problema de programación matemática (problema del camino más corto).

**Ejemplo 1.1** (Continuación)**.** Para empezar a resolverlo, primero resumimos los costes de cada ruta en las siguientes tablas, es decir, los $c(s_i, s_j)$ coste de ir de $s_i$ a $s_j$:

| | B | C | D |
|---|---|---|---|
| A | 2 | 4 | 3 |

| | E | F | G |
|---|---|---|---|
| B | 7 | 4 | 6 |
| C | 3 | 2 | 4 |
| D | 4 | 1 | 5 |

| | H | I |
|---|---|---|
| E | 1 | 4 |
| F | 6 | 3 |
| G | 3 | 3 |

| | J |
|---|---|
| H | 3 |
| I | 4 |

Figura 1.4: Coste de cada ruta.

Queremos entonces encontrar el camino más corto desde cada nodo $s_i$ al nodo J, es decir, una secuencia de movimientos que minimicen el coste total para llegar a J desde cada nodo $s_i$.
Para que el problema tenga solución, es necesario que los caminos que empiecen y acaben en el mismo nodo, es decir los ciclos, no tengan una longitud negativa. En caso de que esto ocurriera, sería posible disminuir la longitud de algunos caminos añadiendo infinitos ciclos de longitud negativa. Esto no será un inconveniente en este problema, ya que podemos comprobar que no hallamos ninguno en el grafo.

Observando el grafo, vemos que un camino óptimo necesitará exactamente 4 movimientos, por lo que introduciremos la formulación necesaria para resolver el problema mediante programación dinámica:
Trabajaremos con un problema de camino más corto con un solo nodo inicial y un solo nodo terminal.
Vamos a formular el problema del camino más corto en términos de un "problema de decisión secuencial", es decir, un problema en el que podemos aplicar la programación dinámica para su resolución, suponiendo que empleamos el método "hacia atrás". Consideraremos, entonces, los siguientes elementos:
Denotaremos como $S$ el espacio de estados, que serán los nodos del grafo, en nuestro caso $S = \{A, B, C, D, E, F, G, H, I, J\}$.
Entonces, para cada $s \in S$, definimos $D(s)$ como las posibles decisiones que podremos tomar dado que estemos en el estado $s$. Por ejemplo, si nos hallamos en el estado $C$, tendremos que $D(C) = \{E, F, G\}$.

Sean ahora los conjuntos

$$S^T \coloneqq \{s \in S \colon D(s) = \emptyset\} \qquad S^I \coloneqq \{s \in S \colon D(s) \neq \emptyset\}.$$

Vemos que $S^T$ es el nodo terminal, mientras que $S^I$ serán los nodos "no terminales". Por tanto $S^T = \{J\}$ y $S^I = \{A, B, C, D, E, F, G, H, I\}$.
Con respecto a la función objetivo, podemos escribirla como:

$$f(x_1, \ldots, x_n) = c(x_0, x_1) + c(x_1, x_2) + \cdots + c(x_{n-1}, x_n),$$

donde

- $x_0$ es el estado inicial, $x_0 = s_1 = A$.
- $x_i$ es la decisión tomada en la etapa $i$: el nodo al que decidimos desplazarnos.
- $c(x_{i-1}, x_i)$ es la longitud del arco $(x_{i-1}, x_i)$, en este caso su coste.
- $s_i = x_{i-1}$ es el estado en el que nos encontramos en la etapa $i$.
- $f(x_1, \ldots, x_n)$ es la longitud (coste) del camino $(A, x_1, \ldots, x_n)$.

Por tanto, usando la estrategia "hacia atrás" para resolver el problema, tenemos ahora que:

$$f_i^*(s_i) = \min_{x_i \in D(s_i)} \{c(s_i, x_i) + f_{i+1}^*(s_{i+1})\}, \qquad s_i \in S_i^I,\ i \in \{1, \ldots, n\}$$

con

$$f_n^*(s_n) = f_n^*(J) = 0$$

siendo así el valor de la función objetivo en la solución óptima $f_1^*(s_1)$, con los valores de las variables de decisión $(x_1^*, x_2^*, \ldots, x_n^*)$.
Definimos entonces $f_i(s_i, x_i)$ como el coste total de una política óptima para las etapas restantes estando la joven en el estado $s_i$, preparada para iniciar la etapa $i$ y eligiendo $x_i$ como siguiente decisión. Entonces sea $x_i^*$ la decisión (no necesariamente única) que minimiza $f_i(s_i, x_i)$, denotamos:

$$f_i^*(s_i) = \min_{x_i} f_i(s_i, x_i) = f_i(s_i, x_i^*).$$

## Resolución por el método backward:

Primero resolveremos el problema mediante el método **backward**, por tanto tenemos que

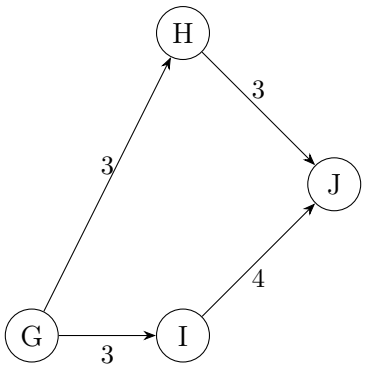
$$\begin{aligned} f_i(s_i, x_i) = &\text{ coste inmediato de la etapa } i \\ &+ \text{ coste futuro mínimo (etapas a partir de la } i+1) \\ = &\, c(s_i, x_i) + f_{i+1}^*(x_i) \end{aligned}$$

donde $c(s_i, x_i)$ esta determinado por las tablas para $c(s_i, s_j)$ donde $s_j = x_i$.

Queremos hallar $f_1^*(s_1)$, donde $s_1 = A$. Determinaremos entonces $f_4^*(s_4), f_3^*(s_3), f_2^*(s_2)$ y por último $f_1^*(s_1)$. Empecemos:
Cuando $i = 4$, a la joven sólo le queda una etapa por recorrer, por tanto $x_4 = J$ quedando esto totalmente determinado por su estado actual $s_4$ (H o I).
Tenemos entonces $f_4^*(s_4) = f_4(s_4, x_4^*) = c(s_4, x_4^*)$ siendo $x_4^* = J$ y las posibles soluciones:

| s | $f_4^*(s_4)$ | $x_4^*$ |
|---|---|---|
| H | 3 | J |
| I | 4 | J |

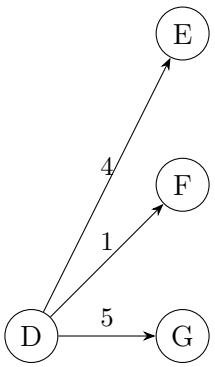
Ahora, si nos desplazamos a la etapa $i = 3$, tenemos ya 6 posibles soluciones.
Supongamos por ejemplo que nos encontramos en el estado G,es decir, $s_3 = G$; entonces la joven debe desplazarse al estado H o I con unos costes inmediatos de $c(s_3, x_3) = 3$ si $x_3 = H$ o $x_3 = I$.




Como podemos ver en el grafo anterior, si la joven elige $x_3 = H$, tendrá un coste adicional óptimo de $f_4^*(H) = 3$, por lo que el coste total de la decisión sería $f_3(s_3, x_3) = c(s_3, x_3) + f_4^*(x_3) = c(G, H) + f_4^*(H) = 3 + 3 = 6$. En cambio, si la joven elige $x_3 = I$, el coste adicional óptimo será $f_4^*(I) = 4$, y el coste total será $f_3(s_3, x_3) = c(s_3, x_3) + f_4^*(x_3) = c(G, I) + f_4^*(I) = 3 + 4 = 7$.
Por tanto, la decisión óptima sería $x_3^* = H$, ya que proporciona el coste mínimo, $f_3^*(x_2) = 6$, donde $x_2 = G$.
Los razonamientos en caso de encontrarnos en cualquiera de los otros dos estados posibles $s_3 = E$ o $s_3 = F$ son análogos, y por tanto los resumimos en esta tabla:

| | $f_3(s_3,x_3)=c(s_3,x_3)+f_4^*(x_3)$ | | | |
|---|---|---|---|---|
| | $x_3$ | | | |
| $s_3$ | H | I | $f_3^*(s_3)$ | $x_3^*$ |
| E | 4 | 8 | 4 | H |
| F | 9 | 7 | 7 | I |
| G | 6 | 7 | 6 | H |

Podemos entonces desplazarnos a la etapa $i=2$. Supongamos que nos encontramos en el estado D, es decir, $s_2=D$ y hagamos todos los cálculos necesarios:




La joven podrá desplazarse al estado E, F o G con costes inmediatos $c(s_2,x_2)=4$ si $x_2=E$, $c(s_2,x_2)=1$ si $x_2=F$ o $c(s_2,x_2)=5$ si $x_2=G$. Como ya lo hemos calculado en la etapa $i=3$, sabemos que el coste adicional mínimo para llegar a $J$ será $f_3^*(x_2)=4$ si $x_2=E$, $f_3^*(x_2)=7$ si $x_2=F$ o $f_3^*(x_2)=6$ si $x_2=G$. Entonces tenemos que, si $s_2=D$:

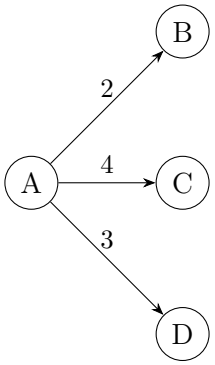
$$x_2=E: \quad f_2(D,E)=f_2(s_2,x_2)=c(s_2,x_2)+f_3^*(x_2)=c(D,E)+f_3^*(E)=4+4=8.$$
$$x_2=F: \quad f_2(D,F)=f_2(s_2,x_2)=c(s_2,x_2)+f_3^*(x_2)=c(D,F)+f_3^*(F)=1+7=8.$$
$$x_2=G: \quad f_2(D,G)=f_2(s_2,x_2)=c(s_2,x_2)+f_3^*(x_2)=c(D,G)+f_3^*(G)=5+6=11.$$

Por tanto, el coste mínimo para llegar de $D$ a $J$ es $f_2^*(D)=8$ y debemos elegir $x_2^*=E$ o $x_2^*=F$.

Haciendo estos mismos cálculos para $s_2=B$ y $s_2=C$ llegamos a la siguiente tabla:

| | $f_2(s_2,x_2)=c(s_2,x_2)+f_3^*(x_2)$ | | | | |
|---|---|---|---|---|---|
| | $x_2$ | | | | |
| $s_2$ | E | F | G | $f_2^*(s_2)$ | $x_2^*$ |
| B | 11 | 11 | 12 | 11 | E o F |
| C | 7 | 9 | 10 | 7 | E |
| D | 8 | 8 | 11 | 8 | E o F |

Podemos ver que, si partimos de los estados $B$ o $D$, es indiferente escoger $x_2^* = E$ o $x_2^* = F$, ya que las dos opciones minimizan el coste hasta llegar al destino final $J$.
Entonces podemos pasar a la última etapa, $i = 1$. Ahora tenemos la facilidad de que solo hay un estado inicial posible, $s_1 = A$, tal y como se muestra en el diagrama:




Hacemos los cálculos entonces sobre cada elección de $x_1$:

$$x_1 = B: \qquad f_1(A,B) = f_1(s_1,x_1) = c(s_1,x_1) + f_2^*(x_1) = c(A,B) + f_2^*(B) = 2 + 11 = 13.$$
$$x_1 = C: \qquad f_1(A,C) = f_1(s_1,x_1) = c(s_1,x_1) + f_2^*(x_1) = c(A,C) + f_2^*(C) = 4 + 7 = 11.$$
$$x_1 = D: \qquad f_1(A,D) = f_1(s_1,x_1) = c(s_1,x_1) + f_2^*(x_1) = c(A,D) + f_2^*(D) = 3 + 8 = 11.$$

Por lo cual se tiene que $f_1^*(s_1) = 11$ con $s_1 = A$ y $x_1^* = C$ o $x_1^* = D$, lo que nos lleva a la tabla:

| | $f_1(s_1,x_1) = c(s_1,x_1) + f_2^*(x_1)$ | | | | |
|---|---|---|---|---|---|
| | $x_1$ | | | | |
| $s_1$ | B | C | D | $f_1^*(s_1)$ | $x_1^*$ |
| A | 13 | 11 | 11 | 11 | C o D |

Hemos llegado así a la solución del problema global, que nos dice que, si escogemos como primera decisión $x_1^* = C$, tenemos la ruta óptima

$$A \to C \to E \to H \to J$$

En cambio, si escogemos como primera decisión $x_1 = D$, tenemos dos posibles rutas óptimas:

$$A \to D \to E \to H \to J$$
$$A \to D \to F \to I \to J$$

Como hemos ido comprobando a lo largo del ejemplo, las tres rutas óptimas tienen el coste mínimo $f_1^*(A) = 11$.
Finalmente, ilustramos las rutas óptimas en el siguiente diagrama:

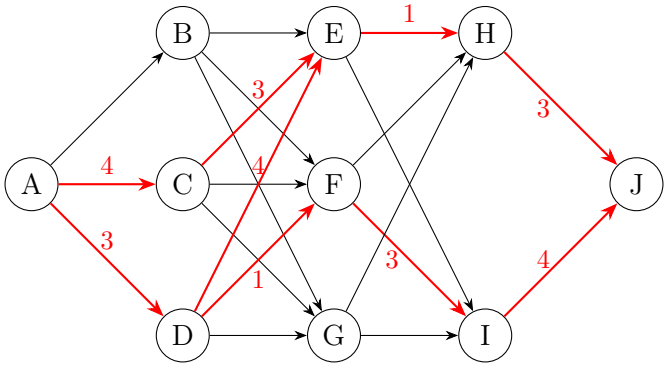



Figura 1.5: Soluciones óptimas del problema.

## Resolución por el método forward:

Ahora resolvemos el problema mediante el método **forward**, por lo que tenemos

$$f_i^*(s_i) = \min_{x_i} f_i(s_i, x_i) = f_i(s_i, x_i^*),$$

donde ahora

$$\begin{aligned} f_i(s_i, x_i) &= \text{coste inmediato de la etapa } i \\ &\quad + \text{ coste anterior mínimo (etapas hasta la } i-1) \\ &= c(x_i, s_i) + f_{i-1}^*(x_i) \end{aligned}$$

donde $c(x_i, s_i)$ esta determinado por las tablas de costes dadas anteriormente.
Ahora queremos hallar $f_4^*(s_4)$, donde $s_4 = J$.
Cuando $i = 1$, la joven se hallará en B, C o D, por lo que evidentemente $x_1 = A$. Tenemos que $f_1^*(s_1) = f_1(s_1, x_1) = c(x_1^*, s_1)$, siendo $x_1^* = A$ :

| $s_1$ | $f_1^*(s_1)$ | $x_1^*$ |
|---|---|---|
| B | 2 | A |
| C | 4 | A |
| D | 3 | A |

Nos desplazamos entonces a la etapa $i = 2$. Suponemos que nos encontramos en el estado $E$, es decir, $s_2 = E$. Por tanto debemos decidir si llegamos a E desde B,C o D. Si hacemos los cálculos para $s_2 = E$ tenemos

$$\begin{aligned} x_2 = B: &\quad f_2(E,B) = f_2(s_2,x_2) = c(x_2,s_2) + f_1^*(x_2) = c(B,E) + f_1^*(B) = 7+2 = 9. \\ x_2 = C: &\quad f_2(E,C) = f_2(s_2,x_2) = c(x_2,s_2) + f_1^*(x_2) = c(C,E) + f_1^*(C) = 3+4 = 7. \\ x_2 = D: &\quad f_2(E,D) = f_2(s_2,x_2) = c(x_2,s_2) + f_1^*(x_2) = c(D,E) + f_1^*(D) = 4+3 = 7. \end{aligned}$$

Por tanto el coste mínimo para llegar de $A$ a $E$ es $f_2^*(E) = 7$ y debemos elegir $x_2^* = C$ o $D$.
Haciendo los mismos cálculos para $s_2 = F$ y $G$ llegamos a la tabla

| | $f_2(s_2, x_2) = c(x_2, s_2) + f_1^*(x_2)$ | | | | |
|---|---|---|---|---|---|
| | $x_2$ | | | | |
| $s_2$ | B | C | D | $f_2^*(s_2)$ | $x_2^*$ |
| E | 9 | 7 | 7 | 7 | C o D |
| F | 6 | 6 | 4 | 4 | D |
| G | 8 | 8 | 8 | 8 | B, C o D |

Podemos entonces desplazarnos a la etapa $i = 3$.
Supongamos que la joven se encuentra en el estado $H$. Por tanto ella puede llegar hasta ahí desde E, F o G con costes inmediatos $c(x_3, s_3) = 1$ si $x_3 = E$, $c(x_3, s_3) = 6$ si $x_3 = F$ o $c(x_3, s_3) = 3$ si $x_3 = G$. Tenemos entonces que si $s_3 = H$:

$$x_3 = E: \quad f_3(H, E) = f_3(s_3, x_3) = c(x_3, s_3) + f_2^*(x_3) = c(E, H) + f_1^*(E) = 1 + 7 = 8.$$
$$x_3 = F: \quad f_2(H, F) = f_3(s_3, x_3) = c(x_3, s_3) + f_2^*(x_3) = c(F, H) + f_1^*(F) = 6 + 4 = 10.$$
$$x_3 = G: \quad f_2(H, G) = f_3(s_3, x_3) = c(x_3, s_3) + f_2^*(x_3) = c(G, H) + f_1^*(G) = 3 + 8 = 11.$$

Por tanto el coste mínimo para llegar de A a H será $f_3^*(H) = 8$ eligiendo $x_3 = E$.
Hacemos los cálculos también para $s_3 = I$ y tenemos:

| | $f_3(s_3, x_3) = c(x_3, s_3) + f_2^*(x_3)$ | | | | |
|---|---|---|---|---|---|
| | $x_3$ | | | | |
| $s_3$ | E | F | G | $f_3^*(s_3)$ | $x_3^*$ |
| H | 8 | 10 | 11 | 8 | E |
| I | 11 | 7 | 11 | 7 | F |

Finalmente llegamos a la etapa $i = 4$, donde el único estado posibles es $s_4 = J$, el destino final. Tenemos que:

| | $f_4(s_4, x_4) = c(x_4, s_4) + f_3^*(x_4)$ | | | |
|---|---|---|---|---|
| | $x_4$ | | | |
| $s_4$ | H | I | $f_4^*(s_4)$ | $x_4^*$ |
| J | 11 | 11 | 11 | H o I |

Hemos llegado entonces a la solución del problema global, que nos dice que podremos seguir las rutas:

$$A \to C \to E \to H \to J$$
$$A \to D \to E \to H \to J$$
$$A \to D \to F \to I \to J$$

con un coste óptimo de $f_4^*(J) = 11$.
Vemos claramente que las soluciones óptimas serán las mismas tanto si se resuelve mediante el método backward como por el forward.

Esto es un claro ejemplo de un problema de **programación dinámica determinística**, en la cual no hay incertidumbre sobre los resultados obtenidos en cada etapa.

## 1.4. Programación dinámica probabilística

En la programación dinámica **probabilística** habrá cierta incertidumbre sobre los resultados obtenidos en cada etapa, ya que existe una distribución de probabilidad para determinar cual será el siguiente estado. Esta distribución de probabilidad está determinada por el estado y la variable de decisión de la etapa actual.
Podemos describir su estructura básica a partir de la siguiente figura:

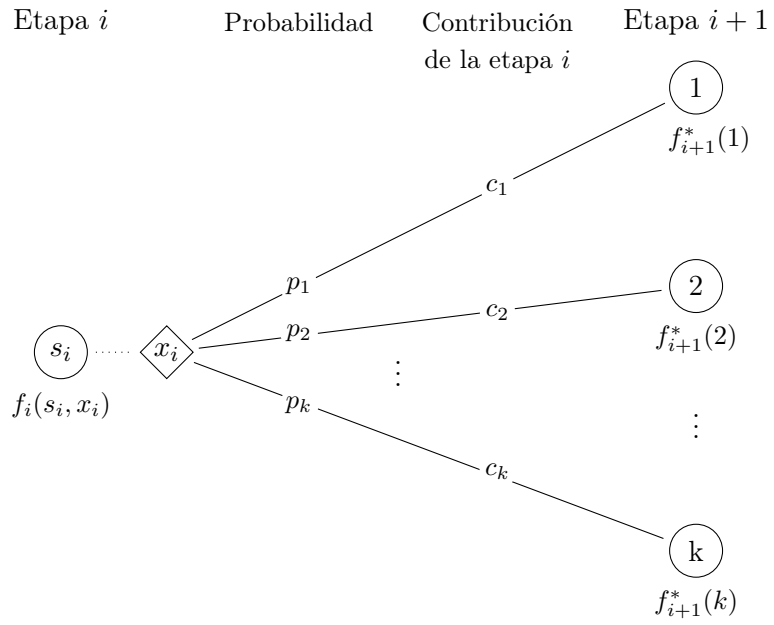



Figura 1.6: Estructura de la programación dinámica probabilística.

Si expandimos la figura 1.6 incluyendo todos los estados y decisiones posibles en cada una de las etapas del problema, se obtiene el árbol de decisión. Este, si no es muy grande, nos proporciona una forma muy útil de resumir todas las posibilidades.
Evidentemente, la relación entre $f_i(s_i, x_i)$ y $f^*_{i+1}(s_{i+1})$ será más complicada que en el caso determinístico, a causa de la estructura probabilística. Por ello, la forma exacta de la relación dependerá de la forma global de la función objetivo.

En este trabajo nos centraremos en la programación dinámica determinística, pero antes ilustraremos estos conceptos sobre la programación dinámica probabilística con un ejemplo:

**Ejemplo 1.3.** Una estudiante especializada en estadística ha desarrollado un sistema que cree que le puede hacer ganar el juego más popular de la feria de verano de su pueblo. Cada jugada consiste en apostar cualquier cantidad de monedas de 1€ disponibles, de forma que, si ganas, doblas la cantidad de monedas que apostaste, y si pierdes, pierdes las monedas involucradas. En cambio, sus amigos no creen que su método sirva de nada, por lo que hacen una apuesta. Esta consiste en que, si comienza con 3€, tendrá que obtener al menos 5€ después de tres jugadas para poder ganarla.
La estudiante estima que su sistema garantiza una probabilidad de $\frac{2}{3}$ de ganar una jugada dada. Si suponemos que esta estimación de probabilidad es la correcta, la pregunta será: ¿Cuántas monedas debe apostar en cada jugada para maximizar la probabilidad de ganar la apuesta a sus amigos?

**RESOLUCIÓN:** Usaremos la siguiente formulación:

$$\begin{aligned} \text{Etapa } i &= \text{i-ésima jugada } (i = 1, 2, 3), \\ x_i &= \text{número de monedas apostadas en la etapa } i, \\ s_i &= \text{número de monedas disponibles al empezar la etapa } i. \end{aligned}$$

El objetivo será maximizar la probabilidad de que la estudiante gane la apuesta, así que la función objetivo que debemos maximizar en cada etapa es la de la probabilidad de terminar las tres jugadas con cinco monedas o más. Por tanto

$$\begin{aligned} f_i(s_i, x_i) &= \text{probabilidad de terminar las tres jugadas con cinco monedas o más,} \\ &\quad \text{teniendo en cuenta que se comienza la etapa } i \text{ con } s_i \text{ monedas y su} \\ &\quad \text{decisión inmediata es } x_i, \text{ en adelante tomando decisiones óptimas,} \\ f^*_i(s_i) &= \max_{x_i=0,1,\ldots,s_i} f_i(s_i, x_i). \end{aligned}$$

En una jugada $i$ dada, si pierde, el estado de la siguiente etapa será $s_i - x_i$ y la probabilidad de acabar con al menos cinco monedas será $f^*_{i+1}(s_i - x_i)$. En cambio, si gana, el estado de la siguiente etapa será $s_i + x_i$ y la probabilidad respectiva $f^*_{i+1}(s_i + x_i)$. La probabilidad de ganar una jugada dada suponemos que es $\frac{2}{3}$, por lo que tenemos

$$f_i(s_i, x_i) = \frac{1}{3} f^*_{i+1}(s_i - x_i) + \frac{2}{3} f^*_{i+1}(s_i + x_i),$$

donde $f^*_4(s_4) = 0$ si $s_4 < 5$ y $f^*_4(s_4) = 1$ si $s_4 \geq 5$. Vemos entonces que la etapa $i$ no afecta directamente a la función objetivo.
Podemos resumir estas relaciones en la figura 1.7 :

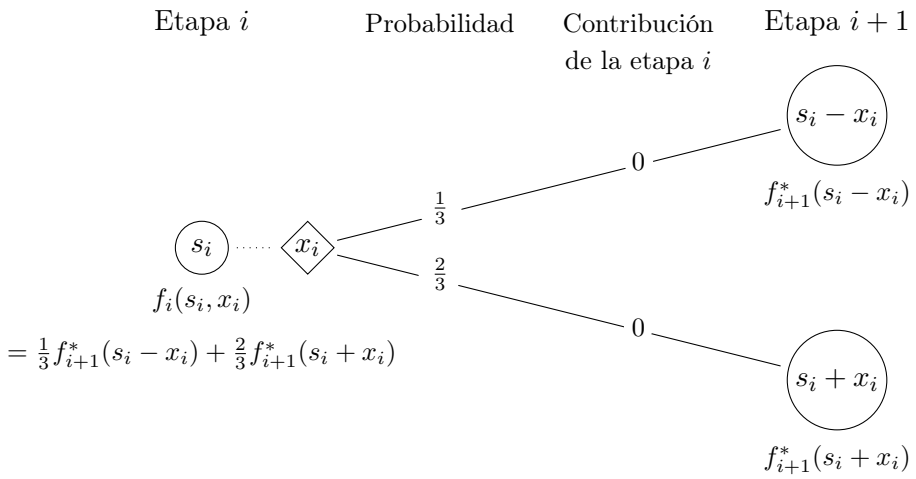



Figura 1.7: Estructura del problema.

Entonces, la relación recursiva dada será:

$$f^*_i(s_i) = \max_{x_i = 0,1,\ldots,s_i} \left\{ \frac{1}{3} f^*_{i+1}(s_i - x_i) + \frac{2}{3} f^*_{i+1}(s_i + x_i) \right\}, \qquad i = 1, 2, 3.$$

Resumimos los cálculos para hallar la solución en las siguientes tablas:

**Etapa $i = 3$:**

| $s_3$ | $f_3^*(s_3)$ | $x_3^*$ |
|---|---|---|
| 0 | 0 | |
| 1 | 0 | |
| 2 | 0 | |
| 3 | $\frac{2}{3}$ | 2 (o más) |
| 4 | $\frac{2}{3}$ | 1 (o más) |
| $\geq 5$ | 1 | 0 |

**Etapa $i = 2$:**

| | $x_2$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | $f_2(s_2, x_2) = \frac{1}{3}f_3^*(s_2 - x_2) + \frac{2}{3}f_3^*(s_2 + x_2)$ | | | | | | |
| $s_2$ | 0 | 1 | 2 | 3 | 4 | $f_2^*(s_2)$ | $x_2^*$ |
| 0 | 0 | | | | | 0 | |
| 1 | 0 | 0 | | | | 0 | |
| 2 | 0 | $\frac{4}{9}$ | $\frac{4}{9}$ | | | $\frac{4}{9}$ | 1 o 2 |
| 3 | $\frac{2}{3}$ | $\frac{4}{9}$ | $\frac{2}{3}$ | $\frac{2}{3}$ | | $\frac{2}{3}$ | 0, 2 o 3 |
| 4 | $\frac{2}{3}$ | $\frac{8}{9}$ | $\frac{2}{3}$ | $\frac{2}{3}$ | $\frac{2}{3}$ | $\frac{8}{9}$ | 1 |
| $\geq 5$ | 1 | | | | | 1 | 0 |

**Etapa $i = 1$:**

| | $x_1$ | | | | | |
|---|---|---|---|---|---|---|
| | $f_1(s_1, x_1) = \frac{1}{3}f_2^*(s_1 - x_1) + \frac{2}{3}f_2^*(s_1 + x_1)$ | | | | | |
| $s_1$ | 0 | 1 | 2 | 3 | $f_1^*(s_1)$ | $x_1^*$ |
| 3 | $\frac{2}{3}$ | $\frac{20}{27}$ | $\frac{2}{3}$ | $\frac{2}{3}$ | $\frac{20}{27}$ | 1 |

Finalmente obtenemos que la secuencia de decisiones óptima es

$$x_1^* = 1 \begin{cases} \text{si gana,} & x_2^* = 1 \begin{cases} \text{si gana,} & x_3^* = 0 \\ \text{si pierde,} & x_3^* = 2 \text{ o } 3 \end{cases} \\ \text{si pierde,} & x_2^* = 1 \text{ o } 2 \begin{cases} \text{si gana,} & x_3^* = \begin{cases} 2 \text{ o } 3 & \text{si } x_2^* = 1 \\ 1, 2, 3 \text{ o } 4 & \text{si } x_2^* = 2 \end{cases} \\ \text{si pierde,} & \text{la apuesta está perdida} \end{cases} \end{cases}$$

Así, la estudiante tendrá una probabilidad de $\frac{20}{27}$ de ganarle la apuesta a sus amigos.

# Capítulo 2

# Programación dinámica determinística

En los problemas de programación dinámica determinística no existe incertidumbre a la hora de calcular las soluciones óptimas, etapa a etapa.
Podemos describirla a partir de la siguiente figura:

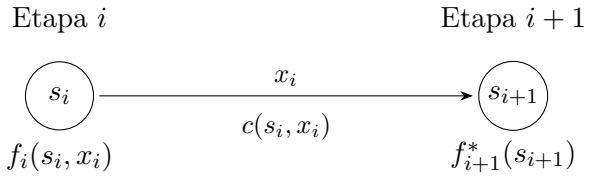



Figura 2.1: Estructura de la programación dinámica determinística.

Si nos hallamos en la etapa $i$, estamos en algún estado $s_i$. Al tomar la decisión $x_i$ pasamos a un estado $s_{i+1}$ en la etapa $i+1$. La contribución óptima a la función objetivo a partir de ese punto la calculamos como $f^*_{i+1}(s_{i+1})$, y sabemos que la variable de decisión $x_i$ también contribuye a la función objetivo. Al combinar estos dos datos de la forma correcta se obtiene $f_i(s_i, x_i)$, la contribución de la etapa $i$ en adelante.
Por tanto, una vez hallados $x^*_i$ y $f^*_i(s_i)$ para cada valor posible de $s_i$, el procedimiento se mueve hacia atrás una etapa.
Se puede ver claramente que este es el procedimiento que hemos utilizado para resolver el ejemplo 1.1.

Hay varias formas en las que podemos clasificar los problemas de programación dinámica determinística:

- **Función objetivo:** El objetivo puede ser tanto minimizar como maximizar la suma de las contribuciones en cada etapa (como en el Ejemplo 1.1), o también minimizar o maximizar el producto de los términos,etc.

- **Estados:** Estos pueden ser discretos o continuos. También se puede dar que necesitemos un vector de estado.

- **Naturaleza de las variables de decisión:** De la misma forma que los estados, las variables de decisión $(x_1, x_2, \ldots, x_n)$ pueden ser discretas o continuas.

A continuación presentaré varios ejemplos para ilustrar estos casos, donde veremos que estas diferencias no serán de importancia, pues la estructura de los problemas es semejante, y el procedimiento de la programación dinámica es similar en todos los casos.

La mayoría de los ejemplos que vamos a ver corresponden a un tipo de problemas que es muy común en la programación dinámica, y consiste en que disponemos solamente de una clase de recurso que debemos asignar a una determinada cantidad de actividades. El objetivo es determinar cómo llevar a cabo esa tarea de la forma más eficaz.
Debido a que estos problemas siempre consisten en asignar un tipo de recurso a cierto número de actividades, tendremos la siguiente formulación de programación dinámica en todos los casos, siendo el orden de las actividades arbitrario:

Etapa $i$ = actividad $i$.

$x_i$ = cantidad de recursos que asignamos a la actividad $i$.

$s_i$ = cantidad de recursos todavía disponibles para asignar a las actividades restantes $(i, \ldots, n)$.

Definimos así el estado $s_i$ ya que la cantidad de recursos que quedan por asignar es justamente la información que necesitamos sobre el estado actual (al comenzar la etapa $i$) para tomar las decisiones de como asignarlos para el resto de actividades.
Por tanto, cuando se inicia la etapa $i$ en el estado $s_i$, la elección de $x_i$ da como resultado que el siguiente estado de la etapa $i+1$ sea $s_{i+1} = s_i - x_i$, tal y como se muestra en la figura 2.2 :

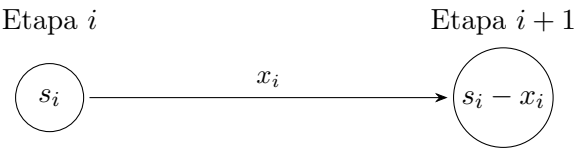



Figura 2.2: Estructura del problema.

Pongamos una serie de ejemplos para ilustrar lo dicho hasta ahora.

## 2.1. Ejemplo con función objetivo lineal

**Ejemplo 2.1.** La organización *Médicos sin fronteras* se dedica a ayudar con equipos médicos a países con recursos escasos o insuficientes.
Recientemente se han detectado 3 países que están en una situación sanitaria deficiente. La organización decide distribuir los 5 equipos de profesionales médicos disponibles entre esos 3 países. Estos no pueden dividirse en equipos más pequeños, es decir, debemos asignar a cada país un número entero de equipos .
La eficiencia de estos se evalúa en función de los años de vida adicionales por persona. En la siguiente tabla se muestran las estimaciones de estos años de vida adicionales por persona (en múltiplos de mil) para cada país y para cada asignación de equipos médicos:

| | Miles de años por persona de vida adicionales | | |
|---|---|---|---|
| | País | | |
| Equipos médicos | 1 | 2 | 3 |
| 0 | 0 | 0 | 0 |
| 1 | 45 | 20 | 50 |
| 2 | 70 | 45 | 70 |
| 3 | 90 | 75 | 80 |
| 4 | 105 | 110 | 100 |
| 5 | 120 | 150 | 130 |

Cuadro 2.1: Años de vida adicionales por persona según la asignación de equipos médicos.

La pregunta a contestar será: ¿Cuál es la asignación que maximiza los años de vida adicionales?

**RESOLUCIÓN:** Este problema requiere tomar tres decisiones interrelacionadas: cuántos equipos asignar a cada uno de los tres países. Consideraremos los países como las tres etapas en las que dividiremos nuestro problema. Por tanto, las variables de decisión $x_i$ $(i = 1, 2, 3)$ son el número de equipos que se le asignan a la etapa o país $i$.
Definiremos entonces los estados como el número de equipos médicos que quedan por ser asignados a los países restantes. Así, en la etapa 1, cuando aún no se ha asignado ningún equipo, $s_1 = 5$. Luego en las etapa 2 y 3, $s_i$ será 5 menos el número de equipos asignados en etapas anteriores, es decir,

$$s_1 = 5, \qquad s_2 = 5 - x_1, \qquad s_3 = s_2 - x_2.$$

Resolveremos este ejemplo mediante el método "backward", por lo que, al hallarnos al inicio de las etapas 2 y 3, debemos considerar todos los estados posibles, es decir, $s_i = 0, 1, 2, 3, 4, 5$.
Establezcamos el problema completo de programación matemática. Sea $p_i(x_i)$ la medida de eficiencia que se obtiene al asignar $x_i$ brigadas al país $i$, entonces debemos elegir $x_1, x_2, x_3$ para:

$$\begin{aligned}
&\text{Maximizar } \sum_{i=1}^{3} p_i(x_i), \\
&\text{sujeto a} \\
&\sum_{i=1}^{3} x_i = 5, \\
&x_i \geq 0, \quad x_i \in \mathbb{Z} \quad \forall i = 1, 2, 3.
\end{aligned}$$

Usando la notación ya definida anteriormente,

$$f_i(s_i, x_i) = p_i(x_i) + \text{máx} \sum_{j=i+1}^{3} p_j(x_j),$$

donde el máximo se toma sobre las $x_{i+1}, \ldots, x_3$ tales que $\sum_{j=i}^{3} x_j = s_i$ y las $x_j$ son enteros no negativos, para $i = 1, 2, 3$. Además,

$$f_i^*(s_i) = \underset{x_i=0,1,\cdots,s_i}{\text{máx}} f_i(s_i, x_i).$$

Por tanto,

$$f_i(s_i, x_i) = p_i(x_i) + f_{i+1}^*(s_i - x_i).$$

Podemos entonces describir la estructura de programación dinámica de este problema mediante la siguiente figura :

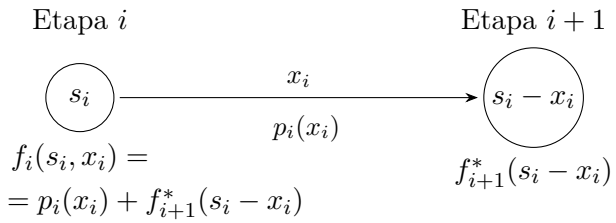



Figura 2.3: Estructura de programación dinámica del problema.

Consecuentemente, la relación recursiva que enlazará las funciones $f_1^*, f_2^*$ y $f_3^*$ en este problema será:

$$f_i^*(s_i) = \max_{x_i=0,1,\cdots,s_i} \{p_i(x_i) + f_{i+1}^*(s_i - x_i)\}, \quad \text{para } i = 1, 2.$$

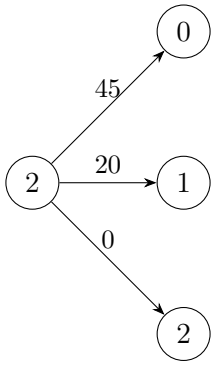
En el caso de la última etapa ($i = 3$),

$$f_3^*(s_3) = \max_{x_3=0,1,\cdots,s_3} p_3(x_3).$$

Una vez aclarada toda la notación, procedemos a hacer los cálculos para la resolución del problema:
En la etapa $i = 3$ podemos observar que los valores de $p_3(x_3)$ aumentan cuanto mayor sea $x_3$. Entonces, si disponemos de $s_3$ equipos médicos para asignar al país número 3, el máximo de $p_3(x_3)$ se logrará asignando los $s_3$ equipos, y así $x_3^* = s_3$ y $f_3^*(s_3) = p_3(s_3)$, como podemos ver en la siguiente tabla:

| $s_3$ | $f_3^*(s_3)$ | $x_3^*$ |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 50 | 1 |
| 2 | 70 | 2 |
| 3 | 80 | 3 |
| 4 | 100 | 4 |
| 5 | 130 | 5 |

Ahora nos movemos hacia atrás a la etapa $i = 2$. Primero necesitamos calcular $f_2(s_2, x_2)$ para todos los valores posibles de $x_2$, es decir, $x_2 = 0, 1, \cdots, s_2$. Supongamos que $s_2 = 2$:



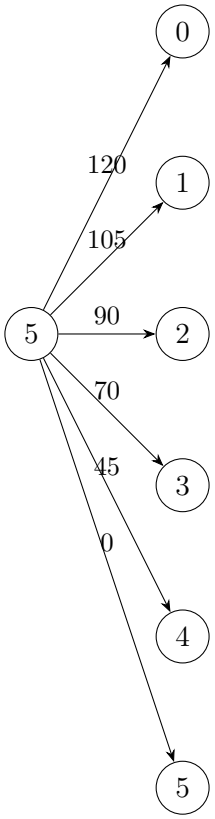

Los costes de las aristas vienen determinados por $p_2(x_2)$. Así, si $x_2 = 0$, entonces tenemos que el estado de la etapa 3 será $s_3 = s_2 - x_2 = 2 - 0 = 2$, mientras que si $x_2 = 1$, entonces $s_3 = 1$ y $x_2 = 2$ conduce a que $s_3 = 0$. Hagamos entonces los cálculos para $s_2 = 2$:

$$x_2 = 0: \quad f_2(2,0) = f_2(s_2, x_2) = p_2(x_2) + f^*_{2+1}(s_2 - x_2) = p_2(0) + f^*_3(2) = 0 + 70 = 70.$$
$$x_2 = 1: \quad f_2(2,1) = f_2(s_2, x_2) = p_2(x_2) + f^*_{2+1}(s_2 - x_2) = p_2(1) + f^*_3(1) = 20 + 50 = 70.$$
$$x_2 = 2: \quad f_2(2,2) = f_2(s_2, x_2) = p_2(x_2) + f^*_{2+1}(s_2 - x_2) = p_2(2) + f^*_3(0) = 45 + 0 = 45.$$

Como nuestra intención es maximizar la función objetivo, entonces si $s_2 = 2$ se tiene que $x^*_2 = 0$ o 1 con $f^*_2(s_2) = 70$. Si hacemos los cálculos también para $s_2 = 0, 1, 3, 4, 5$, resultan en la siguiente tabla:

| | $f_2(s_2, x_2) = p_2(x_2) + f^*_3(s_2 - x_2)$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $x_2$ | | | | | | | |
| $s_2$ | 0 | 1 | 2 | 3 | 4 | 5 | $f^*_2(s_2)$ | $x^*_2$ |
| 0 | 0 | | | | | | 0 | 0 |
| 1 | 50 | 20 | | | | | 50 | 0 |
| 2 | 70 | 70 | 45 | | | | 70 | 0 o 1 |
| 3 | 80 | 90 | 95 | 75 | | | 95 | 2 |
| 4 | 100 | 100 | 115 | 125 | 110 | | 125 | 3 |
| 5 | 130 | 120 | 125 | 145 | 160 | 150 | 160 | 4 |

Con todo los cálculos realizados para la etapa 2, nos movemos hacia atrás para resolver el problema original desde el principio, la etapa $i = 1$. Por ser la primera etapa, el único estado que debemos considerar es el inicial, es decir, $s_1 = 5$:




Los costes de las aristas vienen determinados por $p_1(x_1)$. Si asignamos $x_1$ equipos al país número 1 se llega a que tendremos $5 - x_1$ equipos disponibles en la etapa 2.
Procedemos entonces a realizar todos los cálculos necesarios:

$x_1 = 0: \quad f_1(5,0) = f_1(s_1,x_1) = p_1(x_1) + f^*_{1+1}(s_1 - x_1) = p_1(0) + f^*_2(5) = 0 + 160 = 160.$

$x_1 = 1: \quad f_1(5,1) = f_1(s_1,x_1) = p_1(x_1) + f^*_{1+1}(s_1 - x_1) = p_1(1) + f^*_2(4) = 45 + 125 = 170.$

$x_1 = 2: \quad f_1(5,2) = f_1(s_1,x_1) = p_1(x_1) + f^*_{1+1}(s_1 - x_1) = p_1(2) + f^*_2(3) = 70 + 95 = 165.$

$x_1 = 3: \quad f_1(5,3) = f_1(s_1,x_1) = p_1(x_1) + f^*_{1+1}(s_1 - x_1) = p_1(3) + f^*_2(2) = 90 + 70 = 160.$

$x_1 = 4: \quad f_1(5,4) = f_1(s_1,x_1) = p_1(x_1) + f^*_{1+1}(s_1 - x_1) = p_1(4) + f^*_2(1) = 105 + 50 = 155.$

$x_1 = 5: \quad f_1(5,5) = f_1(s_1,x_1) = p_1(x_1) + f^*_{1+1}(s_1 - x_1) = p_1(5) + f^*_2(0) = 120 + 0 = 120.$

Obtenemos que $x^*_1 = 1$, con $f^*_1(s_1) = 170$. Podemos ver los resultados de la etapa 1 resumidos en la siguiente tabla:

| | $f_1(s_1,x_1) = p_1(x_1) + f^*_2(s_1 - x_1)$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $x_1$ | | | | | | | |
| $s_1$ | 0 | 1 | 2 | 3 | 4 | 5 | $f^*_1(s_1)$ | $x^*_1$ |
| 5 | 160 | 170 | 165 | 160 | 155 | 120 | 170 | 1 |

Llegamos así a la solución del problema global, que nos dice que debemos de escoger como variables de decisión $x_1^* = 1, x_2^* = 3$ y $x_3^* = 1$, lo que conducirá a una estimación de 170000 años de vida adicionales.

## 2.2. Ejemplo con función objetivo no lineal

Ahora mostraremos otro ejemplo en el que la función objetivo no será una función expresada mediante "sumas" , sino mediante "productos". Es un ejemplo en el que la función objetivo no es una función lineal.
En la primera impresión puede parecer que no es un problema de programación dinámica determinística, ya que trata con probabilidades. Pese a esto podremos ver que sí se ajusta a la definición que hemos proporcionado, ya que no existe incertidumbre a la hora de calcular el estado de la siguiente etapa a partir del estado y la decisión de la etapa actual.

**Ejemplo 2.2.** Una productora musical se ha enterado recientemente de un nuevo concurso de música a nivel nacional, cuyo premio por quedar en primer lugar es un gran incentivo para participar en él. La productora quiere estar lo más segura posible de quedar en primer puesto, por lo que presenta a 3 bandas diferentes para participar en él.
En la situación actual, se estima que la probabilidad de que las bandas 1, 2 y 3 fracasen (no queden en primer lugar) es de 0.40, 0.60 y 0.80 respectivamente. Por tanto, la probabilidad total de fracaso es $(0.40)(0.60)(0.80) = 0.192$. El objetivo de la productora es minimizar la probabilidad de que las bandas fracasen, por lo que se asignan 2 músicos de un nivel superior.
En la siguiente tabla mostramos la probabilidad estimada de que las bandas fracasen si se les asigna 0, 1, o 2 músicos para colaborar con ellos:

| | Probabilidades de fracaso | | |
|---|---|---|---|
| | Banda | | |
| Nuevos músicos | 1 | 2 | 3 |
| 0 | 0.40 | 0.60 | 0.80 |
| 1 | 0.20 | 0.40 | 0.50 |
| 2 | 0.15 | 0.20 | 0.30 |

Cuadro 2.2: Probabilidad de fracaso según los músicos asignados.

Evidentemente, solo consideraremos números enteros de músicos ya que no pueden participar en dos bandas a la vez.
La pregunta a responder es: ¿Cómo deben de asignarse los músicos para minimizar la probabilidad de que las tres bandas fracasen?

**RESOLUCIÓN:** La estructura básica de este problema será análoga a la del ejemplo 2.1. El recurso serán los músicos disponibles y las actividades serán las bandas. La diferencia entre los dos problemas es la función objetivo.
Tenemos que la etapa $i$ $(i = 1, 2, 3)$ será el músico $i$, y el estado $s_i$ será el número de músicos todavía disponibles que deben asignarse a las bandas restantes. Las variables de decisión $x_i$ $(i = 1, 2, 3)$ serán el número de músicos que se asignan a la banda $i$.
Sea $p_j(x_j)$ la probabilidad de que el equipo $j$ fracase si se le asignan $x_j$ músicos adicionales, el objetivo de la productora será escoger $x_1, x_2, x_3$ para:

$$\begin{aligned}
&\text{Minimizar } \prod_{i=1}^{3} p_i(x_i),\\
&\text{sujeto a}\\
&\sum_{i=1}^{3} x_i = 2,\\
&x_i \geq 0, \quad x_i \in \mathbb{Z} \quad \forall i = 1, 2, 3.
\end{aligned}$$

Por tanto $f_i(s_i, x_i)$ en este problema es

$$f_i(s_i, x_i) = p_i(x_i) \cdot \text{mín} \prod_{j=i+1}^{3} p_j(x_j),$$

donde el mínimo se toma sobre $x_{i+1}, \cdots, x_3$ tal que

$$\sum_{j=i}^{3} x_j = s_i$$

y los $x_j$ son enteros no negativos para $i = 1, 2, 3$. Así,

$$f_i^*(s_i) = \min_{x_i = 0, 1, \cdots, s_i} f_i(s_i, x_i),$$

con

$$f_i(s_i, x_i) = p_i(x_i) \cdot f_{i+1}^*(s_i - x_i).$$

Podemos entonces describir la estructura de programación dinámica de este problema mediante la siguiente figura :

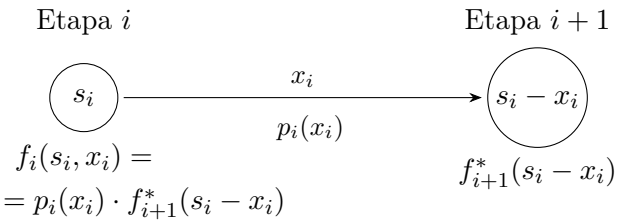



Figura 2.4: Estructura de programación dinámica del problema.

Por tanto, la relación recursiva entre las funciones $f_1^*, f_2^*$ y $f_3^*$ es

$$f_i^*(s_i) = \min_{x_i=0,1,\cdots,s_i} \{p_i(x_i) \cdot f_{i+1}^*(s_i - x_i)\}, \quad \text{para } i = 1, 2.$$

En el caso de la etapa $i = 3$,

$$f_3^*(s_3) = \min_{x_3=0,1,\cdots,s_3} p_3(x_3).$$

Ahora se procedería a realizar todos los cálculos necesario, pero son también análogos a los del Ejemplo 2.1, por lo que los resumimos en las siguientes tablas:

**Etapa $i = 3$:**

| $s_3$ | $f_3^*(s_3)$ | $x_3^*$ |
|---|---|---|
| 0 | 0.80 | 0 |
| 1 | 0.50 | 1 |
| 2 | 0.30 | 2 |

**Etapa $i = 2$:**

| | $f_2(s_2, x_2) = p_2(x_2) + f_3^*(s_2 - x_2)$ | | | | |
|---|---|---|---|---|---|
| | $x_2$ | | | | |
| $s_2$ | 0 | 1 | 2 | $f_2^*(s_2)$ | $x_2^*$ |
| 0 | 0.48 | | | 0.48 | 0 |
| 1 | 0.30 | 0.32 | | 0.30 | 0 |
| 2 | 0.18 | 0.20 | 0.16 | 0.16 | 2 |

**Etapa $i = 1$:**

| | $f_1(s_1, x_1) = p_1(x_1) + f_2^*(s_1 - x_1)$ | | | | |
|---|---|---|---|---|---|
| | $x_1$ | | | | |
| $s_1$ | 0 | 1 | 2 | $f_1^*(s_1)$ | $x_1^*$ |
| 2 | 0.064 | 0.060 | 0.072 | 0.060 | 1 |

De estas tablas se deduce que $x_1^* = 1$, lo que deja $s_2 = 2 - 1 = 1$, luego $x_2^* = 0$, y por último $x_3^* = 1$. Por tanto, a las bandas 1 y 3 se les debe asignar un músico adicional a cada una, siendo la probabilidad de que las bandas fracasen $f_1^*(s_1) = 0.060$ .

## 2.3. Ejemplo con espacio de estados continuo

Hasta ahora, hemos visto ejemplos donde las variables de estado $s_i$ son discretas. Otra característica que presentan estos ejemplos es que son "reversibles en el tiempo", es decir, pueden resolverse tanto por el método backward como por el método forward.
El ejemplo que presentaremos a continuación difiere en ambas características, el estado $s_i$ en la etapa $i$ es una variable continua, por lo que podrá tomar cualquier valor dentro de determinados intervalos. Por tener $s_i$ un número de valores infinito ya no será posible considerar cada uno de sus valores de forma individual. Entonces los valores $f_i^*(s_i)$ y $x_i^*$ deben expresarse como funciones de $s_i$.
Por otro lado, tampoco será reversible, ya que las etapa corresponderán a periodos temporales, por lo que el procedimiento adecuado de solución debe ser "hacia atrás", es decir, debemos utilizar el método backward.

**Ejemplo 2.3.** La carga de trabajo de una famosa empresa de construcción fluctúa ampliamente dependiendo de la temporada en la que se encuentre. Pero debe tener en cuenta que es difícil encontrar empleados capacitados para contratar y costoso formar nuevos trabajadores, por lo que el encargado del equipo de contratación rechaza despedir empleados durante las temporadas con menos demanda. Eso si, tampoco quiere continuar pagando la nómina de temporada alta cuando no son necesarios. Por otro lado, como todos los trabajos se realizan por encargo, es imposible acumular inventario. Por tanto, el encargado debe determinar cuál tiene que ser la programación del número de empleados.
En esta tabla se muestran los resultados proporcionados por un estudio de mercado sobre los trabajadores que se requerirán durante las cuatro temporadas del año siguiente:

| **Temporada** | **Primavera** | **Verano** | **Otoño** | **Invierno** | **Primavera** |
|---|---|---|---|---|---|
| Personal necesario | 255 | 220 | 240 | 200 | 255 |

Cuadro 2.3: Resultados del estudio de mercado realizado.

No se puede permitir que el número de empleados sea menor que el indicado en la tabla

2.3, y cualquier contratación mayor provoca unas pérdidas aproximadas de 2000€ por temporada. Se ha estimado que el coste de cambiar el número de empleados de una temporada a la siguiente son 200€ multiplicado por el cuadrado de la diferencia de nivel, debido a las indemnizaciones que tendrán que pagar y los nuevos empleados que habrá que formar.
Es importante observar que ahora sí será posible contar con fracciones ya que es posible contratar empleados a tiempo parcial, y los costes se aplican de la misma forma a estas fracciones.
La pregunta a responder es: ¿Cuál es el número de empleados de los que se debe disponer en cada temporada para minimizar los costes?

**RESOLUCIÓN:** Podemos observar fácilmente que el número de empleados no será mayor que 255 en ningún momento, ya que esta es la demanda en la temporada pico. Por tanto podemos establecer que el número de empleados en primavera debe de ser 255, y el problema se centrará en hallar el de las otras 3 temporadas.
Las etapas en este problema serán las temporadas. En realidad, vemos que existe un número indefinido de etapas, pues el problema se prolonga hacia el futuro indefinidamente, pero cada año establece un mismo ciclo, por lo que es factible tomar un ciclo de cuatro etapas que termine en primavera. Entonces:

Etapa 1 = verano,

Etapa 2 = otoño,

Etapa 3 = invierno,

Etapa 4 = primavera.

$x_i$ = número de trabajadores de la etapa $i$ $(i = 1, 2, 3, 4)$.

Evidentemente $x_4 = 255$. En las demás etapas, para hallar el nivel óptimo de empleo debe tenerse en cuenta el efecto sobre los costes de la siguiente temporada.
Denotamos por $r_i$ la mano de obra mínima de la temporada $i$, es decir, $r_1 = 220, r_2 = 240, r_3 = 200$ y $r_4 = 255$. Por tanto, tenemos que los valores de $x_i$ factibles serán:

$$r_i \leq x_i \leq 255 \quad \forall\, i = 1, 2, 3, 4.$$

Se tiene también que:

$$\text{Coste en la etapa } i = 200(x_i - x_{i-1})^2 + 2000(x_i - r_i).$$

Podemos observar que el coste de la etapa $i$ dependerá de $x_i$ y de $x_{i-1}$, las decisiones tomadas en esa etapa y en la anterior. Es decir, el número de empleados de la etapa

anterior es toda la información sobre el estado actual que se necesita para determinar las siguientes decisiones óptimas. Entonces el estado de la etapa $i$ será $s_i = x_{i-1}$.
Teniendo en cuenta estos datos, podemos representar la estructura básica de este problema mediante el siguiente diagrama:

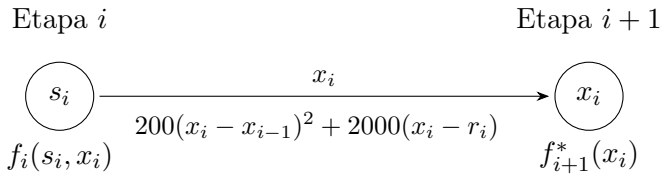



Figura 2.5: Estructura de programación dinámica del problema.

Cuando $i = 1$, $s_1 = x_0 = x_4 = 255$. Es decir,

| $i$ | $r_i$ | $x_i$ | $s_i = x_{i-1}$ | Coste |
|---|---|---|---|---|
| 1 | 220 | $220 \leq x_1 \leq 255$ | 255 | $200(x_1 - 255)^2 + 2000(x_1 - 220)$ |
| 2 | 240 | $240 \leq x_2 \leq 255$ | $220 \leq s_2 \leq 255$ | $200(x_2 - x_1)^2 + 2000(x_2 - 240)$ |
| 3 | 200 | $200 \leq x_3 \leq 255$ | $240 \leq s_3 \leq 255$ | $200(x_3 - x_2)^2 + 2000(x_3 - 200)$ |
| 4 | 255 | 255 | $200 \leq s_4 \leq 255$ | $200(255 - x_3)^2$ |

El objetivo de este problema será escoger $x_1, x_2, x_3$ (con $x_0 = x_4 = 255$) para

$$\text{Minimizar} \sum_{i=1}^{4} [200(x_i - x_{i-1})^2 + 2000(x_i - r_i)],$$

sujeta a

$$r_i \leq x_i \leq 255, \quad \forall\, i = 1, 2, 3, 4.$$

Así, de la etapa $i$ en adelante, como $s_i = x_{i-1}$ tenemos

$$f_i(s_i, x_i) = 200(x_i - x_{i-1})^2 + 2000(x_i - r_i) + \min_{r_j \leq x_j \leq 255} \sum_{j=i+1}^{4} [200(x_j - x_{j-1})^2 + 2000(x_j - r_j)].$$

Así,

$$f^*_i(s_i) = \min_{r_i \leq x_i \leq 255} f_i(s_i, x_i).$$

Entonces,

$$f_i(s_i, x_i) = 200(x_i - x_{i-1})^2 + 2000(x_i - r_i) + f^*_{i+1}(x_i).$$

Por tanto, la relación recursiva que se establece es

$$f^*_i(s_i) = \min_{r_i \leq x_i \leq 255} \{200(x_i - x_{i-1})^2 + 2000(x_i - r_i) + f^*_{i+1}(x_i)\}.$$

Una vez tenemos los elementos necesarios definidos, iniciamos el procedimiento de solución en la etapa $i = 4$.
Se sabe que $x_4^* = 255$, así que se tiene

| $s_4$ | $f_4^*(s_4)$ | $x_4^*$ |
|---|---|---|
| $200 \leq s_4 \leq 255$ | $200(255 - s_4)^2$ | 255 |

Una vez tenemos esta tabla, podemos movernos a la etapa $i = 3$, donde la relación recursiva queda reducida a

$$\begin{aligned} f_3^*(s_3) &= \min_{200 \leq x_3 \leq 255} \{200(x_3 - s_3)^2 + 2000(x_3 - 200) + f_4^*(x_3)\} \\ &= \min_{200 \leq x_3 \leq 255} \{200(x_3 - s_3)^2 + 2000(x_3 - 200) + 200(225 - x_3)^2\}, \end{aligned}$$

donde los posibles valores de $s_3$ son $240 \leq s_3 \leq 255$.
Ahora queremos hallar el valor de $x_3$ que minimiza $f_3(s_3, x_3)$, en términos de $s_3$, donde $s_3$ tiene un valor fijo, aunque desconocido. Para ello igualaremos a cero la primera derivada parcial de $f_3(s_3, x_3)$ con respecto a $x_3$:

$$\begin{aligned} \frac{\partial}{\partial x_3} f_3(s_3, x_3) =& 400(x_3 - s_3) + 2000 - 400(255 - x_3) \\ =& 400(2x_3 - s_3 - 250) \\ =& 0. \end{aligned}$$

Lo que nos dice que

$$x_3 = \frac{s_3 + 250}{2}.$$

Como la segunda derivada es positiva y , para todos los valores de $s_3$ tal que $240 \leq s_3 \leq 255$, tenemos que $200 \leq x_3 \leq 255$, es el mínimo que estábamos buscando. Utilizando que:

$$\begin{aligned} f_3^*(s_3) = f_3(s_3, x_3^*) = 200 \left( \frac{s_3 + 500}{2} - s_3 \right)^2 + 200 \left( 255 - \frac{s_3 + 500}{2} \right)^2 \\ + 2000 \left( \frac{s_3 + 250}{2} - 200 \right) \end{aligned}$$

y reduciendo algebraicamente la expresión, llegamos a que:

| $s_3$ | $f_3^*(s_3)$ | $x_3^*$ |
|---|---|---|
| $240 \leq s_3 \leq 255$ | $50(250 - s_3)^2 + 50(260 - s_3)^2 + 1000(s_3 - 150)$ | $\frac{s_3+250}{2}$ |

Podemos entonces movernos a la etapa $i = 2$.

Se tiene que:

$$\begin{aligned} f_2(s_2, x_2) &= 200(x_2 - s_2)^2 + 2000(x_2 - r_2) + f_3^*(x_2) \\ &= 200(x_2 - s_2)^2 + 2000(x_2 - 240) \\ &\quad + 50(250 - x_2)^2 + 50(260 - x_2)^2 + 1000(x_2 - 150). \end{aligned}$$

La región factible de $s_2$ será $220 \leq s_2 \leq 255$ y la de $x_2$ será $240 \leq x_2 \leq 255$. Entonces el problema será minimizar $x_2$ de forma que

$$f_2^*(s_2) = \min_{240 \leq x_2 \leq 255} f_2(s_2, x_2).$$

Para ello igualamos a cero la primera derivada parcial de $f_2(s_2, x_2)$ con respecto a $x_2$:

$$\begin{aligned} \frac{\partial}{\partial x_2} f_2(s_2, x_2) &= 400(x_2 - s_2) + 2000 - 100(250 - x_2) - 100(260 - x_2) + 1000 \\ &= 200(3x_2 - 2s_2 - 240) \\ &= 0, \end{aligned}$$

y por tanto

$$x_2 = \frac{2s_2 + 240}{3}.$$

También vemos que

$$\frac{\partial^2}{\partial x_2^2} f_2(s_2, x_2) = 600 > 0,$$

así que este valor de $x_2$ nos daría el mínimo de la función si fuera factible, es decir $240 \leq x_2 \leq 255$, cuando $220 \leq s_2 \leq 255$.
En cambio, esta solución sólo será factible si $240 \leq s_2 \leq 255$. Por tanto aún tenemos que encontrar el valor factible de $x_2$ que minimiza $f_2(s_2, x_2)$ cuando $220 \leq s_2 < 240$.
Cuando $s_2 < 240$,

$$\frac{\partial}{\partial x_2} f_2(s_2, x_2) > 0, \qquad \forall\, 240 \leq x_2 \leq 255,$$

Por tanto el valor que estábamos buscando es $x_2 = 240$. Sustituyendo estos valores de $x_2$ en $f_2(s_2, x_2)$ obtenemos:

| $s_2$ | $f_2^*(s_2)$ | $x_2^*$ |
|---|---|---|
| $220 \leq s_2 \leq 240$ | $200(240 - s_2)^2 + 115000$ | 240 |
| $240 \leq s_2 \leq 255$ | $\frac{200}{9}[(240 - s_2)^2 + (255 - s_2)^2$ $+(270 - s_2)^2] + 2000(s_2 - 195)$ | $\frac{2s_2+240}{3}$ |

Pasamos entonces a la etapa $i = 1$.
Tenemos que:

$$\begin{aligned} f_1(s_1, x_1) &= 200(x_1 - s_1)^2 + 2000(x_1 - r_1) + f_2^*(x_1) \\ &= 200(x_1 - s_1)^2 + 2000(x_1 - 220) + f_2^*(x_1), \end{aligned}$$

donde $220 \leq x_1 \leq 250$ es la región factible para $x_1$. Como la expresión de $f_2^*(x_1)$ es diferente según si nos encontramos en el intervalo $220 \leq x_1 \leq 240$ o en $240 \leq x_1 \leq 255$, se tiene:

$$f_1(s_1, x_1) = \begin{cases} 200(x_1 - s_1)^2 + 2000(x_1 - 200) + 200(240 - x_1)^2 & \\ +115000, & \text{si } 220 \leq x_1 \leq 240 \\ 200(x_1 - s_1)^2 + 2000(x_1 - 200) + \frac{200}{9}[(240 - s_2)^2 & \\ +(255 - s_2)^2 + (270 - s_2)^2] + 2000(s_2 - 195), & \text{si } 240 \leq x_1 \leq 255 \end{cases}$$

Considerando el primer caso ($220 \leq x_1 \leq 240$), tenemos

$$\begin{aligned} \frac{\partial}{\partial x_1} f_1(s_1, x_1) &= 400(x_1 - s_1) + 2000 - 400(240 - x_1) \\ &= 400(2x_1 - s_1 - 235). \end{aligned}$$

Sabemos que los empleados en primavera deben de ser $r_1 = 255$, por tanto

$$\frac{\partial}{\partial x_1} f_1(s_1, x_1) = 800(x_1 - 245) < 0 \qquad \forall\, x_1 \leq 240.$$

Es decir, $x_1 = 240$ es el valor que minimiza $f_1(s_1, x_1)$ en el intervalo $220 \leq x_1 \leq 240$.

En el segundo caso ($240 \leq x_1 \leq 255$),

$$\begin{aligned} \frac{\partial}{\partial x_1} f_1(s_1, x_1) &= 400(x_1 - s_1) + 2000 \\ &\quad - \frac{400}{9}[(240 - x_1) + (255 - x_1) + (270 - x_1)] + 2000 \\ &= \frac{400}{3}(4x_1 - 3s_1 - 225). \end{aligned}$$

Como

$$\frac{\partial^2}{\partial x_1^2} f_1(s_1, x_1) > 0 \qquad \forall\, x_1,$$

igualamos la primera derivada parcial de $f_1(s_1, x_1)$ con respecto a $x_1$ a 0, y nos da el valor

$$x_1 = \frac{3s_1 + 225}{4}.$$

Sabemos que $s_1 = 255$, por lo que tenemos que $x_1 = 247,5$ minimiza el valor de $f_1(s_1, x_1)$ en el intervalo $240 \leq x_1 \leq 255$. Por otro lado, vemos que $f_1(s_1, 240) > f_1(s_1, 247{,}5)$, por lo que podemos concluir que $x_1 = 247{,}5$ minimiza $f_1(s_1, x_1)$ en toda la región factible, $220 \leq x_1 \leq 255$.
Ahora, sustituyendo:

$$\begin{aligned} f_1^*(s_1) = f_1^*(255) &= 200(247{,}5 - 255)^2 + 2000(247{,}5 - 220) \\ &\quad + \frac{200}{9}[2(250 - 247{,}5)^2 + (265 - 247{,}5)^2 + 30(742{,}5 - 575)] \\ &= 185000. \end{aligned}$$

Llegando a

| $s_1$ | $f_1^*(s_1)$ | $x_1^*$ |
|---|---|---|
| 255 | 185000 | 247,5 |

Finalmente llegamos a que, si tomamos $s_i = x_{i-1}^*$, obtenemos la solución óptima $x_1^* = 247{,}5$, $x_2^* = 245$, $x_3^* = 247{,}5$, $x_4^* = 255$, con un coste total estimado de 185000€.

## 2.4. Ejemplo con más de una variable por etapa

Al contrario que en los ejemplos anteriores, el problema que presentaremos a continuación tiene tres estados en cada etapa en lugar de uno. Es decir, un vector de estado con tres componentes. Desde el punto de vista teórico, no será una diferencia relevante, ya que simplemente basta con considerar las combinaciones posibles de los valores de los estados. En cambio, desde el punto de vista práctico, la situación es más complicada, ya que la cantidad de cálculos necesarios tiende a hacerse exageradamente grande con rapidez cuando se añaden estados adicionales.
Este es el principal problema de la programación dinámica: el **problema de la dimensión**. Por eso la programación dinámica es más útil en problemas sin un número excesivo de variables.
Aunque, con las capacidades actuales de las computadoras, es ahora un problema menor. El siguiente problema es lo suficientemente pequeño para poder resolverlo sin dificultad.

**Ejemplo 2.4.** Una famosa compañía que produce joyería de alta gama ha decidido reorganizar la línea de producción de la empresa, debido a una reducción de las ganancias.
Su obrador cuenta con tres plantas. En la primera se forja la plata para darle la forma adecuada ; en la segunda realizan el mismo proceso pero con el oro; y en la tercera pulen las piedras preciosas y así las añaden a sus productos.

Han decidido dejar libre una parte de la capacidad de producción para comenzar la elaboración de dos nuevos productos cuyas ventas potenciales son muy prometedoras: un collar de plata y esmeraldas y un anillo de oro y diamantes.
El producto número 1, el collar de plata y esmeraldas, necesita espacio de trabajo solamente en las plantas 1 y 3 para poder llevar a cabo su producción. El producto número 2, el anillo de oro y diamantes, sólo necesita espacio de trabajo en las plantas 2 y 3. El equipo de ventas han estimado que se podrán vender todos los productos que se elaboren. En la siguiente tabla mostramos los datos reunidos por la compañía relevantes sobre las horas de producción disponibles, las necesarias, y las ganancias por lote de cada producto nuevo:

| | Tiempo de producción por lote | | |
|---|---|---|---|
| | Producto | | |
| Planta | 1 | 2 | Tiempo disponible a la semana |
| 1 | 1 | 0 | 4 |
| 2 | 0 | 2 | 12 |
| 3 | 3 | 2 | 18 |
| Ganancia por lote | 3000€ | 5000€ | |

Cuadro 2.4: Datos relevantes para el problema

¿De que forma debemos dividir las horas de producción disponibles entre cada producto para maximizar los beneficios?

**RESOLUCIÓN:** Denotando por $x_1$ las horas de producción que se dedicarán al producto 1 y por $x_2$ las que se dedicarán al 2, vemos que tenemos que resolver un problema de programación lineal de la forma:

$$
\begin{aligned}
&\text{Maximizar } Z = 3x_1 + 5x_2, \\
&\text{sujeta a} \\
&x_1 \leq 4 \\
&2x_2 \leq 12 \\
&3x_1 + 2x_2 \leq 18 \\
&x_1 \geq 0,\ x_2 \geq 0.
\end{aligned}
$$

En este problema se deben de tomar dos decisiones interrelacionadas, $x_1$ y $x_2$. Por tanto, para resolverlo mediante programación dinámica podemos interpretarlas como las dos eta-

pas, es decir $i = 1, 2$. Así, $x_i$ será la decisión de la etapa $i$.
El estado $s_i$ será la cantidad de los respectivos recursos todavía disponibles para ser asignados a las actividades (elaboración de los respectivos productos) restantes. Podemos ver que disponemos de 3 recursos (tiempo disponible en cada planta del obrador), entonces tenemos

$$s_i = (R_1, R_2, R_3),$$

donde $R_j$ es la cantidad del recurso $j$ ($j = 1, 2, 3$) todavía disponible. Es decir,

$$s_1 = (4, 12, 18), s_2 = (4 - x_1, 12, 18 - 3x_1).$$

No obstante, como empezamos la resolución del problema en la etapa 2, no conocemos el valor de $x_1$, por lo que emplearemos $s_2 = (R_1, R_2, R_3)$. Por tanto tenemos

$$\begin{aligned} f_2(R_1, R_2, R_3, x_2) &= \text{contribución de la actividad 2 a Z si el sistema se encuentra en el} \\ &\quad \text{estado } (R_1, R_2, R_3) \text{ al iniciar la etapa 2 y la decisión es } x_2 \\ &= 5\, x_2, \\ f_1(4, 12, 18, x_1) &= \text{contribución de las actividades 1 y 2 a Z si el sistema se encuentra} \\ &\quad \text{en el estado (4,12,18) al iniciar la etapa 1, la decisión inmediata es} \\ &\quad x_1 \text{ y después se toma una decisión óptima en la etapa 2} \\ &= 3\, x_1 + \max_{2x_2 \le 12,\ 2x_2 \le 18 - 3x_1,\ x_2 \ge 0} 5\, x_2. \end{aligned}$$

Por tanto

$$f_i^*(R_1, R_2, R_3) = \max_{x_i} f_i(R_1, R_2, R_3, x_i),$$

donde el máximo lo tomamos entre los valores factibles de $x_i$, por lo que, usando las restricciones del problema obtenemos

$$f_2^*(R_1, R_2, R_3) = \max_{2x_2 \le R_2,\ 2x_2 \le R_3,\ x_2 \ge 0} 5x_2, \tag{2.1}$$

$$f_1(4, 12, 18, x_1) = 3x_1 + f_2^*(4 - x_1, 12, 18 - 3x_1), \tag{2.2}$$

$$f_1^*(4, 12, 18) = \max_{x_1 \le 4,\ 3x_1 \le 18,\ x_1 \ge 0} \{3x_1 + f_2^*(4 - x_1, 12, 18 - 3x_1)\}. \tag{2.3}$$

La ecuación (2.1) se usará para resolver el problema de dos etapas. La ecuación (2.2) proporciona la estructura de programación dinámica que muestra el problema, que también podemos ver en la siguiente figura:

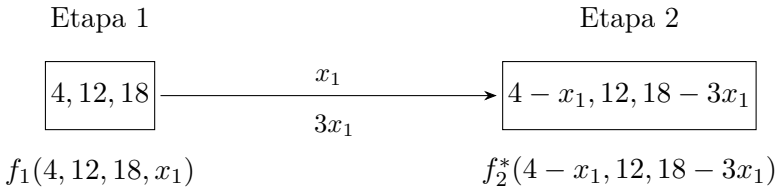



Figura 2.6: Estructura de programación dinámica del problema.

Finalmente, la ecuación (2.3) proporciona la relación recursiva entre $f_1^*$ y $f_2^*$ que se usará para resolver el problema de la etapa 1.

Para la etapa $i = 2$, la ecuación (2.1) nos dice que $x_2^*$ debe ser el mayor valor de $x_2$ que cumple, de manera simultánea, $2x_2 \leq R_2,\ 2x_2 \leq R_3,\ x_2 \geq 0$.
Suponemos que $R_2 \geq 0$ y $R_3 \geq 0$ para que puedan existir soluciones factibles. Entonces $x_2^*$ será el menor valor entre $\frac{R_2}{2}$ y $\frac{R_3}{2}$. Por tanto llegamos a la tabla

| $(R_1, R_2, R_3)$ | $f_2^*(R_1, R_2, R_3)$ | $x_2^*$ |
|---|---|---|
| $R_2 \geq 0,\ R_3 \geq 0$ | $5 \cdot \text{mín}\left\{\frac{R_2}{2}, \frac{R_3}{2}\right\}$ | $\text{mín}\{\frac{R_2}{2}, \frac{R_3}{2}\}$ |

Ahora nos movemos entonces a la etapa $i = 1$, donde, para resolver el problema, sustituimos la solución que acabamos de obtener para $f_2^*(R_1, R_2, R_3)$ en la ecuación (2.3). Como en la etapa 2 $(R_1, R_2, R_3) = (4 - x_1, 12, 18 - 3x_1)$, tenemos que

$$f_2^*(4 - x_1, 12, 18 - 3x_1) = 5 \cdot \text{mín}\left\{\frac{R_2}{2}, \frac{R_3}{2}\right\} = 5 \cdot \text{mín}\left\{\frac{12}{2}, \frac{18 - 3x_1}{2}\right\}$$

es la solución que tenemos que sustituir en la ecuación (2.3). Sustituyendo y combinando las restricciones sobre $x_1$ se llega a que

$$f_1^*(4, 12, 18) = \max_{0 \leq x_1 \leq 4} \left\{3x_1 + 5 \cdot \text{mín}\left\{\frac{12}{2}, \frac{18 - 3x_1}{2}\right\}\right\}.$$

Podemos ver que en la región factible $0 \leq x_1 \leq 4$,

$$\text{mín}\left\{\frac{12}{2}, \frac{18 - 3x_1}{2}\right\} = \begin{cases} 6, & \text{si } 0 \leq x_1 \leq 2 \\ 9 - \frac{3}{2}x_1, & \text{si } 2 \leq x_1 \leq 4 \end{cases}$$

de forma que

$$3x_1 + 5 \cdot \text{mín}\left\{\frac{12}{2}, \frac{18 - 3x_1}{2}\right\} = \begin{cases} 3x_1 + 30, & \text{si } 0 \leq x_1 \leq 2 \\ 45 - \frac{9}{2}x_1, & \text{si } 2 \leq x_1 \leq 4 \end{cases}$$

Como ambas las dos funciones

$$\max_{0\leq x_1\leq 2}\{3x_1+30\} \qquad \text{y} \qquad \max_{2\leq x_1\leq 4}\{45-\frac{9}{2}x_1\}$$

alcanzan su máximo en $x_1 = 2$, concluimos que

| $(R_1, R_2, R_3)$ | $f_1^*(R_1, R_2, R_3)$ | $x_1^*$ |
|---|---|---|
| (4,12,18) | 36 | 2 |

Tenemos que $x_1^* = 2$, por tanto

$$R_1 = 4 - 2 = 2 \qquad R_2 = 12 \qquad R_3 = 18 - 3 \cdot 2 = 12$$

para la etapa 2. En consecuencia, $x_1^* = 2,\ x_2^* = 6$ será la solución óptima de este problema, con el valor de Z igual a 36, es decir, la empresa debe elaborar los productos 1 y 2 a un ritmo de 2 y 6 lotes por semana respectivamente, con un beneficio de 36000€ semanales.

# Capítulo 3

# Aplicaciones de la programación dinámica

En este capítulo presentaremos dos problemas resueltos con la técnica de programación dinámica. En el primer caso utilizaremos también el lenguaje R para hallar su solución.

## 3.1. El problema de la mochila resuelto con R

En esta sección resolveremos un problema clásico mediante programación dinámica e implementaremos la solución en R.

### 3.1.1. El problema de la mochila

Disponemos de una mochila con una cierta capacidad limitada. Por otro lado, disponemos de una serie de objetos, cada uno de ellos ocupa un cierto sitio en la mochila (al cual llamaremos peso) y nos proporciona un beneficio determinado.
Entonces, debemos decidir que objetos meteremos en la mochila para maximizar el beneficio, de forma que los objetos no podrán superar la capacidad máxima de la mochila.
Tomaremos un caso particular del problema de la mochila donde los objetos no podrán repetirse, es decir, solo dispondremos de una unidad de cada objeto. Veámoslo:

**Ejemplo 3.1.** Un comerciante va a hacer un viaje a una famosa empresa de otra ciudad, para acudir a una reunión donde venderá algunos de sus productos.
En la siguiente tabla podemos ver una lista de sus productos con sus respectivos pesos y beneficios que obtiene al venderlos:

| Producto | Peso (en kg) | Beneficio (en €) |
|---|---|---|
| Sartén | 2 | 30 |
| Plancha | 3 | 35 |
| Olla convencional | 6 | 40 |
| Olla a presión | 7 | 55 |

Cuadro 3.1: Lista de productos del comerciante.

Solo tendrá que llevar una unidad de cada producto, ya que la empresa no necesita más y, evidentemente, no podremos fraccionar los productos.
Una vez comprado el billete de avión, le informan de que sólo le dejarán facturar una maleta que pese como máximo 10kg.
Suponiendo que venderá todo lo que lleve en la maleta, ¿qué productos tendrá que llevar para obtener el máximo beneficio?

**RESOLUCIÓN:** Para resolver este problema, consideraremos los productos del comerciante como las 4 etapas en las que lo dividiremos. Por tanto, las variables de decisión $x_i$ ($i = 1, 2, 3, 4$) serán si optamos por meter el objeto $i$ en la maleta o no ($x_i = 0$ si dejamos el objeto fuera y $x_i = 1$ si decidimos llevarlo).
Dado un producto $i$, denominaremos $c_i$ al beneficio que genera y $p_i$ a su peso.
Entonces definiremos los estados como el peso disponible que queda en la maleta. Por tanto, en la etapa 1, cuando aún no se ha metido ningún producto en la maleta, $s_1 = 10$. Por tanto en las etapas 2,3 y 4 será 10 menos el peso ocupado anteriormente, es decir

$$s_1 = 10, \qquad s_2 = 10 - p_1x_1 = 10 - 2x_1,$$

$$s_3 = s_2 - p_2x_2 = s_2 - 3x_2, \qquad s_4 = s_3 - p_3x_3 = s_3 - 6x_3.$$

Es decir, tendremos que escoger $x_1, x_2, x_3$ y $x_4$ para:

Maximizar $\sum_{i}^{4} c_i x_i$
sujeto a
$\sum_{i}^{4} p_i x_i \leq 10$
$x_i \in \{0, 1\} \quad i = 1, 2, 3, 4.$

Definimos entonces

$$f_i(s_i, x_i) = c_i x_i + \text{máx} \sum_{j=i+1}^{4} c_j x_j,$$

tomando el máximo sobre los $x_{i+1}, \ldots, x_4$ tales que $\text{máx} \sum_{j=i}^{4} p_j x_j = s_i$. Por tanto,

$$f_i^*(s_i) = \underset{x_i=0,1}{\text{máx}} f_i(s_i, x_i).$$

Entonces

$$f_i(s_i, x_i) = c_i x_i + f_{i+1}^*(s_i - p_i x_i).$$

Describimos pues la estructura de programación dinámica del problema a través de la siguiente figura:

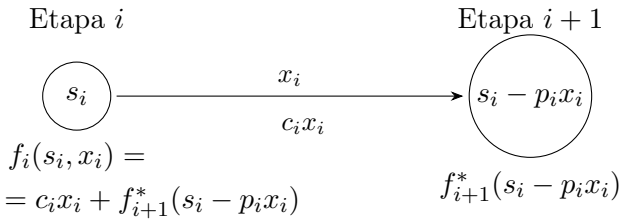



Figura 3.1: Estructura de programación dinámica del problema.

Tenemos entonces que la relación recursiva descrita en este problema será

$$f_i^*(s_i) = \underset{x_i=0,1}{\text{máx}} \{c_i x_i + f_{i+1}^*(s_i - p_i x_i)\}, \quad \text{para } i = 1, 2, 3.$$

En el caso de la etapa $i = 4$ tendremos

$$f_4^*(s_4) = \underset{x_4=0,1}{\text{máx}} c_4 x_4.$$

Resolveremos este problema mediante el método backward, por lo que comenzaremos por la etapa $i = 4$.
Podemos ver que $c_4 x_4$ será mayor si $x_4 = 1$. Por tanto, si queda el suficiente espacio en la maleta, $x_4^* = 1$ y $f_4^*(s_4) = c_4 x_4 = 55$, como podemos ver en la tabla

| $s_4$ | $f_4^*(s_4)$ | $x_4^*$ |
|---|---|---|
| $10 > s_4 \geq 7$ | 55 | 1 |
| $s_4 < 7$ | 0 | 0 |

Nos movemos entonces hacia atrás a la etapa $i = 3$. Debemos diferenciar según el espacio libre que quede en la mochila. Teniendo en cuenta que $p_3 = 6$ y $p_4 = 7$, diferenciaremos los casos $s_3 \leq 6$, $s_3 = 6$ y $6 < s_3 \leq 10$.
Por ejemplo, si $6 < s_3 \leq 10$,

$$x_3 = 0: \quad f_3(s_3, 0) = f_3(s_3, x_3) = c_3x_3 + f^*_{3+1}(s_3 - p_3x_3) = 40 \cdot 0 + f^*_4(s_3 - 6 \cdot 0) = 0 + 55 = 55.$$
$$x_3 = 1: \quad f_3(s_3, 1) = f_3(s_3, x_3) = c_3x_3 + f^*_{3+1}(s_3 - p_3x_3) = 40 \cdot 1 + f^*_4(s_3 - 6 \cdot 1) = 40 + 0 = 40.$$

Por tanto vemos que, si $6 < s_3 \leq 10$, tendremos $x^*_3 = 0$ con $f^*_3(s_3) = 55$. Haciendo estos cálculos para los tres casos posibles, tenemos

| | $f_3(s_3, x_3) = c_3x_3 + f^*_4(s_3 - p_3x_3)$ | | | |
|---|---|---|---|---|
| | $x_3$ | | | |
| $s_3$ | 0 | 1 | $f^*_3(s_3)$ | $x^*_3$ |
| $s_3 \leq 6$ | 0 | | 0 | 0 |
| $s_3 = 6$ | 0 | 40 | 40 | 1 |
| $6 < s_3 \leq 10$ | 55 | 40 | 55 | 0 |

Moviéndonos hacia atrás llegamos a la etapa $i = 2$. Volvemos a tener que dividir el problema en casos según el valor del estado. Como $p_2 = 3$, tendremos en cuenta los casos $s_2 < 3$, $3 \leq s_2 \leq 6$, $s_2 = 6$, $6 < s_2 < 9$, $s_2 = 9$ y $s_2 = 10$.
Vemos que, si $s_2 = 9$, se tiene

$$x_2 = 0: \quad f_2(9, 0) = f_2(s_2, x_2) = c_2x_2 + f^*_{2+1}(s_2 - p_2x_2) = 35 \cdot 0 + f^*_3(9 - 3 \cdot 0) = 0 + 55 = 55.$$
$$x_2 = 1: \quad f_2(9, 1) = f_2(s_2, x_2) = c_2x_2 + f^*_{2+1}(s_2 - p_2x_2) = 35 \cdot 1 + f^*_3(9 - 3 \cdot 1) = 35 + 40 = 75.$$

Es decir, si $s_2 = 9$, obtenemos $x^*_2 = 1$ con $f^*_2(s_2) = 75$. Realizando los cálculos para los seis casos posibles, se llega a que

| | $f_2(s_2, x_2) = c_2x_2 + f^*_3(s_2 - p_2x_2)$ | | | |
|---|---|---|---|---|
| | $x_2$ | | | |
| $s_2$ | 0 | 1 | $f^*_2(s_2)$ | $x^*_2$ |
| $s_2 < 3$ | 0 | | 0 | 0 |
| $3 \leq s_2 < 6$ | 0 | 35 | 35 | 1 |
| $s_3 = 6$ | 40 | 35 | 40 | 0 |
| $6 < s_3 < 9$ | 55 | 35 | 55 | 0 |
| $s_2 = 9$ | 55 | 75 | 75 | 1 |
| $s_2 = 10$ | 55 | 90 | 90 | 1 |

Una vez realizados todos los cálculos de la etapa $i = 2$, podemos retroceder para resolver el problema original desde el principio, la etapa $i = 1$.
Para la primera etapa hemos visto que $s_1 = 10$, por lo que entonces los cálculos necesarios se resumen a los siguientes:

$$x_1 = 0: \quad f_1(10,0) = f_1(s_1,x_1) = c_1x_1 + f^*_{1+1}(s_1 - p_1x_1) = 30 \cdot 0 + f^*_3(10 - 2 \cdot 0) = 0 + 90 = 90.$$
$$x_1 = 1: \quad f_1(10,1) = f_2(s_2,x_2) = c_1x_1 + f^*_{1+1}(s_1 - p_1x_1) = 30 \cdot 1 + f^*_3(9 - 2 \cdot 1) = 30 + 55 = 85.$$

Resumiendo,

| | $f_1(s_1,x_1) = c_1x_1 + f^*_2(s_1 - p_1x_1)$ | | | |
|---|---|---|---|---|
| | $x_1$ | | | |
| $s_1$ | 0 | 1 | $f^*_1(s_1)$ | $x^*_1$ |
| 10 | 90 | 85 | 90 | 0 |

Así llegamos a la solución global del problema, siendo esta

$$x^*_1 = 0,\ x^*_2 = 1,\ x^*_3 = 0,\ x^*_4 = 1 \quad \text{con } f^*_1(s_1) = 90.$$

Es decir, en la maleta el comerciante llevará la plancha y la olla a presión, pesando su equipaje justamente 10 kg y obteniendo un beneficio de 90€.

### 3.1.2. Solución del problema de la mochila en R

Una vez vista la resolución a mano, veamos y apliquemos el código que nos permite solucionar este problema en R.

```
s=c(1,2,3,4,5,6,7,8,9,10) #Espacio de estados
p=c(2,3,6,7)              #Peso de cada objeto
c=c(30,35,40,55)          #Beneficio de cada objeto
s1=10                     #Estado en la etapa 1


#CUARTA ETAPA

fop=cbind(s, rep(0,length(s)), rep(0,length(s)), rep(0,length(s)), rep(0,length(s)))
colnames(fop)=c("Estados", "x=0", "x=1","ValueMax","Decisión óptima")
```

```
fop[,2]= ifelse(p[4]*0>fop[,1],0,0)
fop[,3]= ifelse(p[4]*1>fop[,1],0,c[4])
fop[,4]= apply(fop[,2:3],1,max)
fop[,5]= apply(fop[,2:3],1,which.max) -1

#Por tanto la tabla para la etapa 4 será:
fop
```

```
      Estados x=0 x=1 ValueMax Decisión óptima
[1,]        1   0   0        0               0
[2,]        2   0   0        0               0
[3,]        3   0   0        0               0
[4,]        4   0   0        0               0
[5,]        5   0   0        0               0
[6,]        6   0   0        0               0
[7,]        7   0  55       55               1
[8,]        8   0  55       55               1
[9,]        9   0  55       55               1
[10,]      10   0  55       55               1
```

```
#TERCERA ETAPA

folc = cbind(s, rep(0,length(s)), rep(0,length(s)), rep(0,length(s)), rep(0,length(s)))
colnames(folc)=c("Estados", "x=0", "x=1","ValueMax","Decisión óptima")
folc[,2]=fop[,4]
folc[,3]= ifelse(p[3]>folc[,1],0,ifelse(p[4]+p[3]>folc[,1],c[3],c[3]+c[4]))
folc[,4]= apply(folc[,2:3],1,max)
folc[,5]= apply(folc[,2:3],1,which.max) -1

#La tabla para la etapa 3 será:
folc
```

```
      Estados x=0 x=1 ValueMax Decisión óptima
```

```
[1,]       1   0   0        0                0
[2,]       2   0   0        0                0
[3,]       3   0   0        0                0
[4,]       4   0   0        0                0
[5,]       5   0   0        0                0
[6,]       6   0  40       40                1
[7,]       7  55  40       55                0
[8,]       8  55  40       55                0
[9,]       9  55  40       55                0
[10,]     10  55  40       55                0
```

```
#SEGUNDA ETAPA

fpla= cbind(s, rep(0,length(s)), rep(0,length(s)), rep(0,length(s)), rep(0,length(s)))
colnames(fpla)=c("Estados", "x=0", "x=1","ValueMax","Decisión óptima")
fpla[,2]= folc[,4]
fpla[,3]= ifelse(p[2]>fpla[,1],0,ifelse(p[2] + p[3] >fpla[,1],c[2],
          ifelse(p[2] + p[4] >fpla[,1],c[2] + c[3],ifelse(p[2]+p[3] + p[4] >fpla[,1],
          c[2]+c[4],c[2]+c[3]+c[4]))))
fpla[,4]= apply(fpla[,2:3],1,max)
fpla[,5]= apply(fpla[,2:3],1,which.max) -1

#La tabla para la etapa 2 será:
fpla
```

```
      Estados x=0 x=1 ValueMax Decisión óptima
[1,]       1   0   0        0                0
[2,]       2   0   0        0                0
[3,]       3   0  35       35                1
[4,]       4   0  35       35                1
[5,]       5   0  35       35                1
[6,]       6  40  35       40                0
[7,]       7  55  35       55                0
[8,]       8  55  35       55                0
[9,]       9  55  75       75                1
```

```
[10,]     10  55  90       90                1
```

```
#PRIMERA ETAPA
fsar=c(s1,0,0)
fsar[2]= fpla[s1,4]
fsar[3]= c[1]+ fpla[s1-p[1],4]

fsar
[1] 10 90 85

#Por tanto, la solución del problema será:

print(paste0("Debo meter ",which.max((fsar[2:3]))-1, " sartenes ",
 fpla[s1-p[1]*(which.max((fsar[2:3]))-1),5], " planchas ",
 folc[s1-p[1]*(which.max((fsar[2:3]))-1)
 -p[2]*fpla[s1-p[1]*(which.max((fsar[2:3]))-1),5],5], " ollas convencionales y ",
 fop[s1-p[1]*(which.max((fsar[2:3]))-1)-p[2]*(which.max(fpla[2:3])-1)
 -p[3]*folc[s1-p[1]*(which.max((fsar[2:3]))-1)
 -p[2]*fpla[s1-p[1]*(which.max((fsar[2:3]))-1),5],5],5], " ollas a presión." ))

[1] "Debo meter 0 sartenes 1 planchas 0 ollas convencionales y 1 ollas a presión."

print(paste0("El beneficio será: ", max(fsar[2:3])))

[1] "El beneficio será: 90"
```

## 3.2. Alineamiento de secuencias de ADN

En esta sección nos basaremos en el artículo *El Genoma Humano*, de Antonio Gómez Tato y Enrique Macías Virgós, profesores de la USC [GoMa].
El ADN, o ácido desoxirribonucleico, es la molécula que contiene la información genética de los humanos y de la mayoría de organismos, formada por secuencias de pequeñas unidades llamadas nucleótidos. Esta molécula consiste en dos cadenas que se enlazan entre ellas para formar una estructura de doble hélice.

La información genética del ADN está almacenada como un código que consiste en cuatro "bases" químicas: adenina (A), guanina (G), citosina (C) y timina (T).
Un proyecto de secuenciación de ADN implica un gran esfuerzo, ya que hay que estudiar su estructura, determinar sus genes, ver cuáles son funcionales e identificar esa función, relacionarlos con los genes encontrados en otras especies, entre otras tareas. Todas estas forman parte de la bioinformática, una disciplina científica que utiliza técnicas de computación y matemáticas.

Nos encontramos con el problema de que las bases de datos genéticos contienen cantidades extremadamente grandes de datos en bruto. El genoma humano por sí solo contiene aproximadamente 3 mil millones de pares de bases de ADN. Para buscar a través de todos estos datos y encontrar relaciones significativas dentro de ellos, los biólogos moleculares dependen cada vez más de algoritmos de computación eficientes.
La forma más común para comparar dos secuencias es alinearlas, de manera que se tenga el mayor número de coincidencias posible. Un proceso similar se hace en lingüística, para investigar sobre la procedencia de las palabras. Por ejemplo, si tomamos las palabras *puerto* (español) y *porto* (portugués) vemos que podemos alinearlas si introducimos un hueco de forma que tenemos p-orto y puerto, que indica una transformación del fonema "o" en el diptongo "ue" en español.

Un alineamiento puede ser **local** o **global**. Si tenemos las palabras *bajorrelieve* y *altibajo*, vemos que solamente tendrá sentido alinear un trozo de ellas, *bajo*, por lo que realizaríamos un alineamiento local. En cambio, en el ejemplo anterior con las palabras *puerto* y *porto*, la mejor opción será hacer un alineamiento global, para así ver su evolución desde la palabra latina *portus*.

Dadas dos secuencias de ADN, la forma de encontrar el mejor alineamiento entre ellas parece sencilla, basta establecer un sistema de puntuación para calificar cada alineamiento, y seleccionar el que tenga la mejor calificación. Sin embargo, el número de alineamientos posibles crece muy rápidamente a medida que aumentan las longitudes de las secuencias. Dadas dos secuencias de longitud 100 existirán aproximadamente unos $2 \cdot 10^{78}$ alineamientos distintos.
Por tanto, se recurre a diversos algoritmos que nos proporcionen el mejor alineamiento en un tiempo aceptable. Los más populares son el de Needleman-Wunsch para alineamiento global y Smith-Waterman para alineamiento local.

### 3.2.1. El algoritmo de Needleman-Wunsch

El objetivo de este algoritmo será buscar alineamientos óptimos de subsecuencias más pequeñas y luego reconstruir el alineamiento óptimo global, es decir, hallar el alineamiento óptimo mediante programación dinámica.

El primer paso es establecer un sistema de puntuación para así poder calificar cada alineamiento posible. La puntuación de los huecos se suele dejar al criterio del investigador. Cuánto más alejadas estén las especies de las que provienen las secuencias menos penalización se debe poner a la inserción de uno o más huecos.
Usaremos:

- Coincidencia de letra: 2 puntos.
- No coincidencia: -1 punto.
- Un hueco: -2 puntos.

Luego, se crea una tabla con las dos secuencias de ADN que se quieren alinear. En este caso, alinearemos las secuencias AAGTC y AATG.

| | - | A | A | G | T | C |
|---|---|---|---|---|---|---|
| - | 0 | -2 | -4 | -6 | -8 | - 10 |
| A | -2 | | | | | |
| A | -4 | | | | | |
| T | -6 | | | | | |
| G | -8 | | | | | |

Se van rellenando las casillas ”paso a paso”. Cada movimiento que se realice significa que se va completando ”base a base” cada secuencia. Un movimiento en vertical representa la inserción de un hueco en la secuencia que está en horizontal y un movimiento en horizontal nos indica un hueco en la secuencia que está en vertical. Un movimiento diagonal representa avanzar una posición en cada secuencia.
Viendo las casillas ya rellenadas, vemos que alinear A y - tiene una puntuación de -2. Si avanzamos en horizontal, estaremos alineando AA y - -, lo que tendrá una puntuación de $-2 + (-2) = -4$.

Ahora se va completando la tabla con las puntuaciones que correspondan: se calculan los valores de las celdas, a partir de las reglas de puntuación que se han dado, sumando las puntuaciones conforme se va avanzando y partiendo de la esquina superior izquierda con un valor de 0, es decir, utilizando el método ”forward”.

| | - | A |
|---|---|---|
| - | 0 | -2 |
| A | -2 | |

Calculemos el valor de la casilla inferior derecha:
Si venimos desde la casilla superior, significa que hemos añadido un hueco, y por tanto se sumarán -2 puntos a los -2 que ya había en la casilla superior, por lo que tendremos -4 puntos.
Si venimos desde la casilla de la izquierda, significa que hemos añadido un hueco, es decir, sumamos -2 puntos a los -2 de la casilla de la izquierda. En total, -4 puntos.
Si venimos "en diagonal", significa que hemos avanzado una posición en ambas secuencias. En ambos casos tenemos una A, por lo que habrá una coincidencia y sumaremos 2 puntos a los 0 puntos que ya teníamos. Es decir, un total de 2 puntos.

Es evidente entonces que la decisión óptima será venir en diagonal, con un valor de 2 puntos. Por tanto las tablas serán:

| | - | A | A | G | T | C |
|---|---|---|---|---|---|---|
| - | 0 | -2 | -4 | -6 | -8 | - 10 |
| A | -2 | 2 | | | | |
| A | -4 | | | | | |
| T | -6 | | | | | |
| G | -8 | | | | | |

| | - | A | A | G | T | C |
|---|---|---|---|---|---|---|
| - | 0 | -2 | -4 | -6 | -8 | - 10 |
| A | -2 | Diag | | | | |
| A | -4 | | | | | |
| T | -6 | | | | | |
| G | -8 | | | | | |

Si realizamos este proceso casilla a casilla, llegamos a las tablas:

| | - | A | A | G | T | C |
|---|---|---|---|---|---|---|
| - | 0 | -2 | -4 | -6 | -8 | - 10 |
| A | -2 | 2 | 0 | -2 | -4 | - 6 |
| A | -4 | 0 | 4 | 2 | 0 | -2 |
| T | -6 | -2 | 2 | 3 | 4 | 2 |
| G | -8 | -4 | 0 | 4 | 2 | 3 |

| | - | A | A | G | T | C |
|---|---|---|---|---|---|---|
| - | 0 | -2 | -4 | -6 | -8 | - 10 |
| A | -2 | Diag | Diag | Izq | Izq | Izq |
| A | -4 | Diag | Diag | Izq | Izq | Izq |
| T | -6 | Arr | Arr | Diag | Diag | Izq |
| G | -8 | Arr | Arr | Diag | Diag | Diag |

Por tanto, si nos situamos en la esquina inferior derecha y retrocedemos en las direcciones óptimas, obtenemos el alineamiento:

A A G T C
A A - T G

con una puntuación de 3.

Esto es solo un pequeño ejemplo, ya que en la vida real habrá que trabajar con secuencias de ADN con una longitud muchísimo mayor, para las cuales nos tendremos que apoyar en una computadora.

# Bibliografía

[BeDr] Bellman, R. E. and Dreyfus, S. E., *Applied Dynamic Programming*, 1st ed., Princeton University Press, 1962.

[LeMa] Lew, A. and Mauch, H., *Dynamic Programming A Computational Tool*, 1st ed., Springer-Verlag, Berlin, 2007.

[Be12] Bertsekas, D. P., *Dynamic programming and optimal control*, 3rd ed., Dover, 2012.

[HiLi] Hillier, F. S. y Lieberman, G. J., *Introducción a la investigación de operaciones*, 9th ed., The McGraw-Hill Companies, Mexico, 2010.

[Sn92] Sniedovich, M., *Dynamic Programming*, 1st ed., Marcel Dekker, New York, 1992.

[HiRiS] Hinderer, K., Rieder, U. and Stieglitz, M., *Dynamic Optimization*, 1st ed., Sringer, 2016.

[GoMa] Gómez Tato, A., y Macías Virgós, E., *El Genoma Humano*, Unidad Didáctica «Matemáticas del Planeta Tierra»: Fundación Española para la Ciencia y la Tecnología (FECYT), 2014.