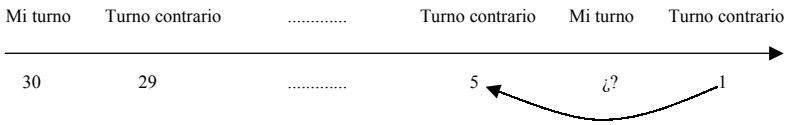
TEMA 1: PROGRAMACIÓN DINÁMICA

INTRODUCCIÓN: La programación dinámica es un método de solución de problemas que permiten descomponer un modelo matemático de gran magnitud, que puede ser muy difícil de resolver, en diversos problemas más pequeños que por lo general son de resolución mucho más fáciles. Además, el método de programación dinámica permite descomponer el problema de manera que una vez que se han resuelto los problemas menores se tiene la solución óptima para el problema mayor, *principio de optimalidad*. Cada uno de tales problemas más pequeños que se crean, se identifican con una etapa del procedimiento de solución de la programación dinámica. Con frecuencia, tales etapas se crean por el hecho de que se debe tomarse una secuencia de decisiones en el transcurso del tiempo. En la mayor parte de los casos, no es posible considerar a estos problemas más pequeños como si fueran completamente independientes de los demás y es aquí en donde el método de la programación dinámica resulta útil.

EJEMPLO: Supóngase que hay 30 cerillos en una mesa. Se comienza tomando 1,2 o 3. Entonces el oponente debe tomar 1, 2 o 3 cerillos. Continuando de este modo hasta que se toma el último cerillo. El perdedor es el jugador que toma el último cerillo. ¿Cómo se puede estar seguro (si mi turno es el primero) de ganar el juego?.

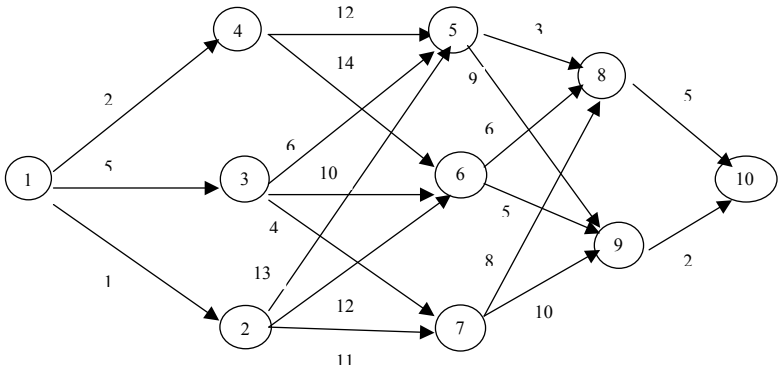
SOLUCIÓN: Si puedo asegurar que el turno de mi contrario sea cuando queda un cerillo, ganaré. Dando un paso hacia atrás, si puedo asegurar que sea el turno de mi contrario cuando queden 5 cerillos, ganaré. La razón de ello es que independientemente de lo que haga cuando queden 5 cerillos, puedo asegurar que cuando sea su siguiente turno quedará sólo un cerillo, ¿por qué?. Igualmente, puedo hacer que mi contrario juegue cuando queden 5, 9, 13, 17, 21, 25 o 29 cerillos. Tengo segura la victoria. Así, no puedo perder si escojo 30-29=1 cerillo en mi primer turno. Después tan sólo me aseguro que mi contrario siempre tenga 29, 25, 21, 17, 13, 9 o 5 cuando sea su turno. Nótese que hemos resuelto este rompecabezas avanzando hacia atrás o en reversa desde el final del problema hacia el principio.



EJERCICIO: Se tiene una taza de 9 litros de capacidad y una de 4 litros. ¿Cómo se puedo medir la cantidad de 6 litros de agua, en el supuesto que se puede utilizar todo el agua que sea necesario?.

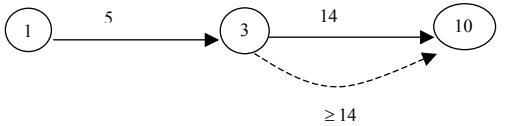
NOTA: En contra a lo que sucede en otros problemas, la programación dinámica no cuenta con una formulación estándar, sino que se trata de un enfoque de tipo general para soluciones de problemas y las ecuaciones específicas que se usan se deben desarrollar para que representan cada situación individual. Es por ello que se presentarán a continuación una batería de ejemplos con características específicas que se resolverán utilizando la técnica de programación dinámica.

PROBLEMA DE RED o RUTA MÁS CORTA: Considérese la siguiente red, donde se representa distintos nodos, así como los arcos de conexión entre los mismos, anotándose la distancia en kilómetros entre los nodos.



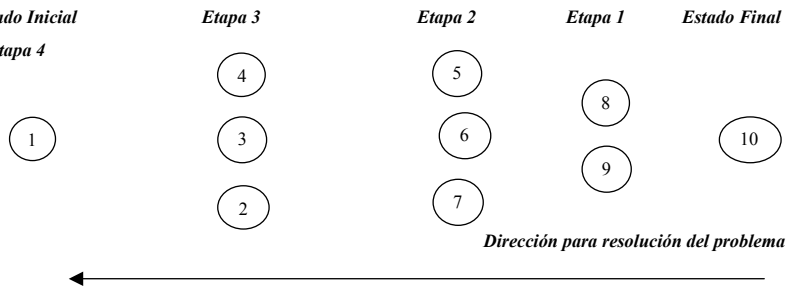
Se pide encontrar la ruta más corta entre el nodo 1 y 10.

SOLUCIÓN: El abordamiento de este problema con programación dinámica esencialmente implica considerar a cada nodo como si se encontrara en la ruta óptima para hacer los cálculos de acuerdo a esta consideración. Se comenzará en el nodo terminal 10 hasta el origen 1. Una vez determinado la ruta óptima de cada nodo, se obtiene la ruta óptima de toda la red. Para ello se basa en el *principio de optimidad*: si un nodo determinado se encuentra en la ruta óptima, entonces el trayecto más corto de ese nodo al final se encuentra también en la ruta óptima.



Objetivo: Obtener el nodo 10 desde el 1 minimizando la distancia recorrida.

Etapas: Se necesitan 4 etapas para alcanzar el nodo 10 desde el nodo 1.

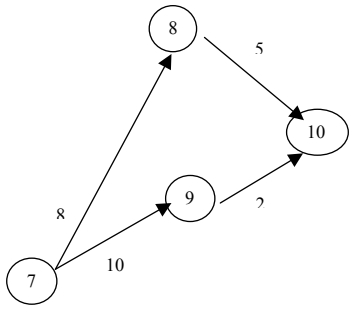


Procedimiento: Determinar la distancia mínima de cada nodo al nodo final 10. Para ello empezamos con los de la etapa 1, más cerca al nodo final y acabamos en la etapa 4, nodo inicial. A partir de esta información se resuelve el problema desde el nodo 1 al 10.

ETAPA 1			
Nodo de Entrada	***Decisión sobre el arco***	***Distancia entre los nodos***	***Distancia más corta al nodo final 10***
8	10	5	5
9	10	2	2

ETAPA 2			
Nodo de Entrada	***Decisión sobre el arco***	***Distancia entre los nodos***	***Distancia más corta al nodo final 10***
5	***8***	***3***	***3+5=8***
	9	9	9+2=11
6	8	6	6+5=11
	9	***5***	***5+2=7***
7	8	8	8+5=13
	9	***10***	***10+2=12***

Observación: La ruta óptima no es aquella que se obtiene menor distancia en la etapa, pues depende de las etapas anteriores.

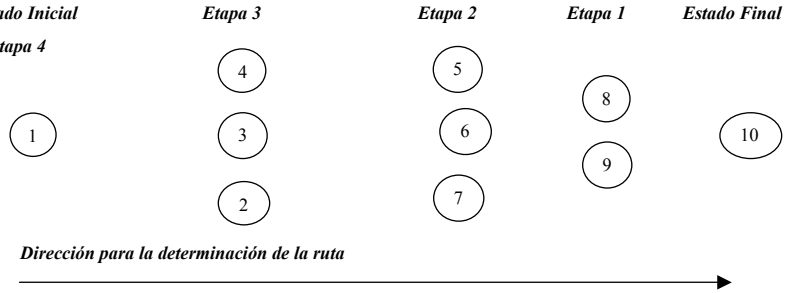


ETAPA 3			
Nodo de Entrada	***Decisión sobre el arco***	***Distancia entre los nodos***	***Distancia más corta al nodo final 10***
2	*5*	13	13+8=21
	6	***12***	***12+7=19***
	7	11	11+12=23
3	***5***		
	6		
	7		
4	***5***		
	6		

ETAPA 4			
Nodo de Entrada	***Decisión sobre el arco***	***Distancia entre los nodos***	***Distancia más corta al nodo final 10***
1	*2*		
	3		
	4		

Conclusión: La distancia mínima entre el nodo 1 y 10 es de 19 kilómetros. A continuación representamos la ruta óptima.

ANÁLISIS DE RESULTADOS



Ruta óptima: *1-3-5-8-10* con *19* kilómetros de distancia.

Puede ocurrir que varias rutas sean óptimas, es decir tienen el mismo valor de recorrido.

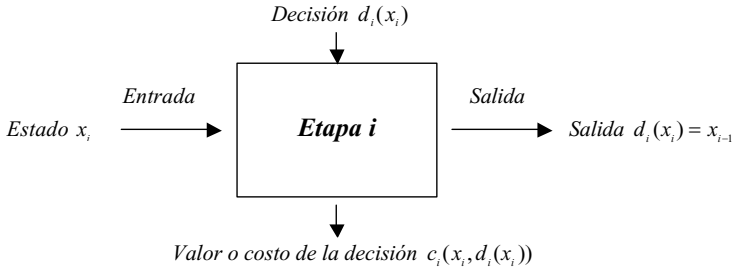
Nota importante: Obsérvese que la aplicación de la programación dinámica ha reducido el números de cálculos que hubiéramos utilizado para la determinación de la ruta óptima a partir de enumeración exhaustiva de todas las posibles.

CARACTERÍSTICAS GENERALES DE LA PROGRAMACIÓN DINÁMICA: En primer lugar reseñar, que una manera de reconocer una situación que puede ser formulada como un problema de programación dinámica, es poder identificar una estructura análoga a la del problema de red, anteriormente presentado. Se distinguen las siguientes características:

1. Obtener una solución óptima en algún sentido, *función objetivo,* del problema planteado.
2. División del problema en etapas que requieren una política de decisión óptima en cada una de ellas, digamos *N* etapas, enumeradas del final al inicio.
3. Cada etapa tiene cierto número de estados asociado con su inicio. Denotaremos por x_i a un estado de la etapa *i*-ésima, $i = 1,...,N$. En general, los estados son las distintas condiciones posibles en las que se puede encontrar el sistema en cada etapa del problema. El número de estados puede ser *finitos o infinitos*.

	Etapa 1	***Etapa 2***	***Etapa 3***	***Etapa 4***
Estados x_i				

4. El efecto de la política de decisión en cada etapa es transformar el estado actual en un estado asociado con el inicio de la etapa anterior, tal vez de acuerdo a una distribución de probabilidad, *programación dinámica probabilística.*



5. El procedimiento de solución, está diseñado para encontrar una política óptima para el problema completo, es decir una política de decisión óptima en cada etapa para cada uno de los estados posibles. Dado el estado actual x_i en la etapa *i*, la decisión óptima depende sólo del estado actual y de las etapas $i-1,...,1$ y no como se llegó al estado x_i. *Principio de optimalidad para programación dinámica.*

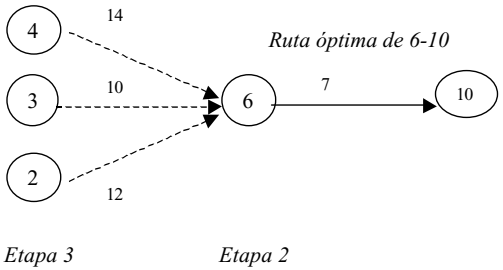
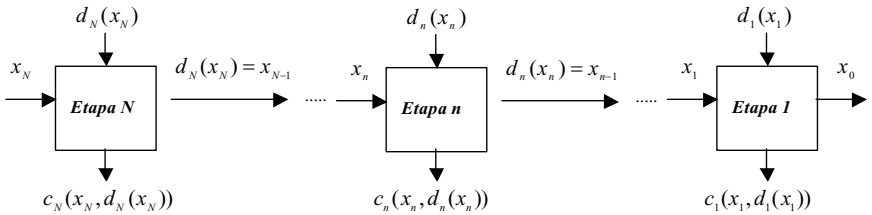


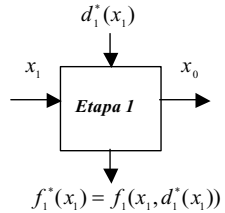
Diagrama de decisión:



Determinación de la política óptima en cada etapa

6. El procedimiento de solución se inicia al encontrar la política óptima para la primera etapa. La política óptima para la primera etapa prescribe la política óptima de decisión $d_1^*(x_1)$ para cada estado x_1 posible de esa etapa. La determinación de la política óptima $d_1^*(x_1)$ se obtiene maximizando o minimizando, según proceda la función objetivo del estado x_1 al estado x_0.

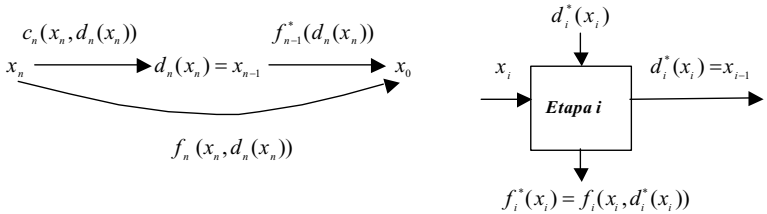
$$f_1^*(x_1) = f_1(x_1, d_1^*(x_1)) = \min_{d_1(x_1)} \{c_1(x_1, d_1(x_1))\},\ x_1 \mapsto d_1^*(x_1)$$



En el caso del problema de red, sólo había una única decisión la cual era óptima en la etapa primera.

7. Con el fin de determinar la política óptima en cada etapa *i* para el estado x_i se obtiene una relación recursiva $\propto$ con la etapa anterior, de tipo suma, producto..., del costo total como sigue:

$$f_i^*(x_i) = f_i(x_i, d_i^*(x_i)) = \min_{d_i(x_i)} \{c_i(x_i, d_i(x_i)) \propto f_{i-1}(x_{i-1}, d_{i-1}(x_{i-1}))\},\ x_i \mapsto d_i^*(x_i)$$



8. Una vez encontrado la política óptima de cada estado, se determina la política óptima del problema final, de manera recursiva a partir de la solución óptima de la última etapa, como sigue:

	Etapa N	*Etapa N-1*	*Etapa 2*	*Etapa 1*	
Inicio: x_N	$d_N^*(x_N) = x_{N-1}^*$	$d_{N-1}^*(x_{N-1}^*) = x_{N-2}^*$	$d_2^*(x_2^*) = x_1^*$	$d_1^*(x_1^*) = x_0^*$	Final: x_0^*

Determinación dela política óptima del problema original

Política óptima: $x_N^* \mapsto x_{N-1}^* \mapsto ... \mapsto x_2^* \mapsto x_1^* \mapsto x_0^*$, ***Valor total:*** $f_N(x_N, d_N^*(x_N))$

Resumen de la notación utilizada:

N: número de etapas en las que se divide el problema.
n: etiqueta para la etapa actual $(n = 1,..., N)$
x_n: estado actual para la etapa *n*.
$d_n(x_n)$: variable de decisión para la etapa *n* en el estado x_n.
$c_n(x_n, d_n(x_n))$: Costo por tomar la decisión $d_n(x_n)$ en el estado x_n de la etapa *n*.
$f_n(x_n, d_n(x_n))$: contribución a la función objetivo de la etapas *n, n-1,..., 1* si el sistema se encuentra en el estado x_n en la etapa *n* y la decisión inmediata es $d_n(x_n)$ y en adelante se toma la decisión óptima.
$d_n^*(x_n)$: decisión óptima dado el estado x_n de la etapa *n*.
$f_n^*(x_n) = f(x_n, d_n^*(x_n)) = \max_{d_n(x_n)} o \min \{f_n(x_n, d_n(x_n))\}$

Aplicación al problema de red:

ETAPA 1				
x_1	$c_1(x_1, d_1(x_1))$	$d_1^*(x_1)$	$f_1^*(x_1)$	x_0
8				
9				

ETAPA 2					
x_2	$c_2(x_2, d_2(x_2)) + f_1^*(d_2(x_2))$		$d_2^*(x_2)$	$f_2^*(x_2)$	x_1^*
	8	***9***			
5					
6					
7					

ETAPA 3						
x_3	$c_3(x_3, d_3(x_3)) + f_2^*(d_3(x_3))$			$d_3^*(x_3)$	$f_3^*(x_3)$	x_2^*
	5	***6***	***7***			
2						
3						
4						

ETAPA 4						
x_4	$c_4(x_4, d_4(x_4)) + f_3^*(d_4(x_4))$			$d_4^*(x_4)$	$f_4^*(x_4)$	x_3^*
	2	***3***	***4***			
1						

Política óptima: $x_N^* \mapsto x_{N-1}^* \mapsto ... \mapsto x_2^* \mapsto x_1^* \mapsto x_0^*$, ***Valor total:*** $f_N(x_N, d_N^*(x_N))$

NOTAS Y COMENTARIOS GENERALES SOBRE PROGRAMACIÓN DINÁMICA:

- Utilizando programación dinámica, puede dividirse un problema grande y complejo en una secuencia de problemas menores interrelacionales. Al resolver en forma secuencial los problemas de menor tamaño, se encuentra la solución óptima para el problema mayor. La programación dinámica es un método para resolver problemas, no es una técnica específica que se puede aplicar de la misma manera a diversos problemas.

- La programación dinámica se ha aplicado a una amplia gama de problemas: programación de producción, control de inventarios, presupuestos de capital, asignación de recursos, reemplazo de equipos,...

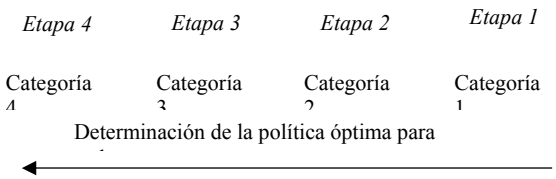
- Se podría realizar una clasificación en cuanto a la acumulación de las eficiencias y/o efectividades parciales del sistema. Estas pueden añadirse en forma de suma, producto o bien máximo y mínimo. Otra clasificación atendiendo a la optimización total, maximizar o minimizar. También el sistema puede tener restricciones o no. El número de etapas o estados puede ser finito o infinito. En cuanto a la certeza de la función de eficacia, puede ser determinista o probabilística.

PROBLEMA DE LA MOCHILA: Con frecuencia se encuentra el problema de la mochila en aplicaciones dinámica. La idea básica es que existan *N* tipos de artículos que pueden llevarse en una mochila. Cada artículo tiene asociado un peso y un valor. El problema consiste en determinar cuántas unidades de cada artículo se deben colocar en una mochila con el fin de maximizar el valor total, teniendo en cuenta la *restricció*n de peso máximo permisible.

EJEMPLO: Supóngase que como programador de tareas se le encarga el estudio de selección de ciertas tareas que se deben procesar durante los próximos 10 días. En la siguiente tabla se presenta la lista de tareas que esperan ser procesadas, no simultáneamente, en un procesador. Asimismo, se muestra el tiempo estimado que se requiere para determinar cada tarea y la calificación sobre el valor que le corresponde a cada categoría de tareas.

Tarea	***Número de tareas a procesar***	***Tiempo estimado por tareas en días***	***Valor de tarea***
Categoría 1	***4***	***1***	***2***
Categoría 2	***3***	***3***	***8***
Categoría 3	***2***	***4***	***11***
Categoría 4	***2***	***7***	***20***

SOLUCIÓN: Este problema se puede plantear como uno de programación dinámica que implica cuatro etapas. En la etapa 1 se debe decidir cuántas tareas se deben procesar del la categoría 1, en la etapa 2 se decide sobre el número de tareas de la categoría 2 que se van a procesar , y así sucesivamente. Darse cuenta que no influye la enumeración de las categorías de las tareas.



Notación:

x_n : Número de días de procesamiento que restan cuando se ha llegado a la etapa *n*.

$d_n(x_n)$: Tareas a procesar de la categoría *n* cuando quedan días x_n de procesamiento.

$c_n(x_n, d_n(x_n))$: Valor obtenido por tomar la decisión de procesar $d_n(x_n)$ tareas de la categoría *n* en el estado x_n.

$f_n(x_n, d_n(x_n))$: Valor total obtenido por procesar el número óptimo de tareas de las categorías *n, n-1,..., 1* si restan x_n días de procesamiento en la etapa *n* y la decisión inmediata es $d_n(x_n)$.

$d_n^*(x_n)$: decisión óptima dado el estado x_n de la etapa *n.*

$f_n^*(x_n) = f(x_n, d_n^*(x_n)) = \max_{d_n(x_n)} \{f_n(x_n, d_n(x_n))\}$

IMPORTANTE: Darse cuenta que en esta situación los estados dependen de las etapas posteriores como sigue:

Etapa	***Estados***
4	$x_4 = 10$ *(número total de días disponibles al inicio de la asignación)*
3	$x_3 = x_4 - 7d_4(x_4)$
2	$x_2 = x_3 - 4d_3(x_3)$
1	$x_1 = x_2 - 3d_2(x_2)$
0	$x_0 = x_1 - 1d_1(x_1)$ *(número de días que han sobrado)*

Asimismo, los rendimientos o valores de la tareas serán función lineal del número de tareas a programar como sigue:

Etapa	***Rendimiento o valor de las tareas***
4	$c_4(x_4, d_4(x_4)) = 20d_4(x_4)$
3	$c_3(x_3, d_3(x_3)) = 11d_3(x_3)$
2	$c_2(x_2, d_2(x_2)) = 8d_2(x_2)$

1	$c_1(x_1, d_1(x_1)) = 2d_1(x_1)$

ETAPA 1: $f_1(x_1, d_1(x_1)) = 2d_1(x_1)$ y $f_1^*(x_1) = f_1^*(x_1, d_1^*(x_1)) = \max_{d_1(x_1)} \{f_1(x_1, d_1(x_1))\}$

Estados x_1	***Decisión óptima***	***Valor total*** $f_1(x_1)$	$x_0 = x_1 - d_1^*(x_1)$
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			

ETAPA 2: $f_2(x_2, d_2(x_2)) = 8d_2(x_2) + f_1^*(x_2 - 3d_2(x_2))$

ETAPA 2							
x_2	$c_2(x_2, d_2(x_2)) + f_1^*(x_2 - 3d_2(x_2))$				$d_2^*(x_2)$	$f_2^*(x_2)$	$x_1^* = x_2 - 3d_2^*(x_2)$
	0	***1***	***2***	***3***			
0							
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							

ETAPA 3: $f_3(x_3, d_3(x_3)) = 11d_3(x_3) + f_2^*(x_3 - 4d_3(x_3))$

ETAPA 3						
x_3	$c_3(x_3,d_3(x_3)) + f_2^*(x_3 - 4d_3(x_3))$			$d_3^*(x_3)$	$f_3^*(x_3)$	$x_2^* = x_3 - 4d_3^*(x_3)$
	0	***1***	***2***			
0						
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

ETAPA 4: $f_4(x_4, d_4(x_4)) = 20d_4(x_4) + f_3^*(x_4 - 7d_4(x_4))$

ETAPA 4					
x_4	$c_4(x_4,d_4(x_4)) + f_3^*(x_4 - 7d_4(x_4))$		$d_4^*(x_4)$	$f_4^*(x_4)$	$x_3^* = x_4 - 7d_4^*(x_4)$
	0	***1***			
10					

Política óptima: $x_4^* \mapsto x_3^* \mapsto x_2^* \mapsto x_1^* \mapsto x_0^*$, ***Valor total:*** $f_4(x_4, d_4^*(x_4))$

OBSERVACIONES:

- La función objetivo es maximizar.
- Existe una serie de restricciones sobre los estados y las decisiones
- Una ventaja del método de programación dinámica es que con pocos cálculos se puede determinar las tareas si se precisa de un periodo inferior a 10, por ejemplo con 8 días. Será suficiente calcular de nuevo al etapa 4 con $x_4 = 8$.
- Puede representarse el problema como una red donde los nodos *(i, j)* son la capacidad *i* disponible para la etapa *j, j-1, ...,1,* donde los arcos determinan el valor obtenido.

ETAPA 4					
x_4	$c_4(x_4,d_4(x_4)) + f_3^*(x_4 - 7d_4(x_4))$		$d_4^*(x_4)$	$f_4^*(x_4)$	$x_3^* = x_4 - 7d_4^*(x_4)$
	0	***1***			
8					

Primera política óptima: $x_4^* \mapsto x_3^* \mapsto x_2^* \mapsto x_1^* \mapsto x_0^*$, ***Valor total:*** $f_4(x_4, d_4^*(x_4))$

Segunda política óptima: $x_4^* \mapsto x_3^* \mapsto x_2^* \mapsto x_1^* \mapsto x_0^*$, ***Valor total:*** $f_4(x_4, d_4^*(x_4))$

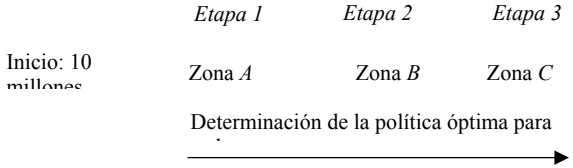
PROBLEMA DE INVERSIÓN O ASIGNACIÓN DE RECURSOS: La programación dinámica es comúnmente utilizado para la resolución de problemas de inversión o asignación de recursos. En este grupo de problemas existe sólo una clase de recursos (material, dinero,...) que deben asignarse a un cierto número de actividades o inversiones. El objetivo es determinar cómo distribuir el recurso entre las actividades de la manera más efectiva.

EJEMPLO 1: Supóngase que la Junta de Extremadura tiene un capital presupuestado de 10 millones de euros que puede utilizar para sus planes de formación en nuevas tecnologías en zonas rurales *(A),* zonas urbanas-rurales *(B)* y zonas urbanas *(C).* Para la zona *A* hay tres alternativas de ejecución, 4 para la *B* y 2 para la *C*. El costo estimado por zonas para cada alternativa, así como el valor de utilidad obtenido si se realiza dicha opción, viene especificado en la siguiente tabla:

Alternativa	***A***		***B***		***C***	
	Costo	***Utilidad***	***Costo***	***Utilidad***	***Costo***	***Utilidad***
1	***0***	***0***	***0***	***0***	***0***	***0***
2	***2***	***5***	***5***	***8***	***1***	***3***
3	***4***	***6***	***6***	***9***	***-***	***-***
4	***-***	***-***	***8***	***12***	***-***	***-***

SOLUCIÓN: Con el fin de resolver el problema con programación dinámica, consideraremos 3 etapas representando a las 3 zonas de atención. La entrada al sistema son 10 millones de euro. La decisión en cada etapa consiste en *determinar la alternativa a asignar a cada zona teniendo en cuenta el costo y su utilidad.*

Vamos a resolver el problema de *entrada a salida*, es decir siguiendo el siguiente diagrama:



Notación:

x_n : Inversión disponible después de asignar alternativa a la zona *n*.

$d_n(x_n)$: Alternativa elegida para la zona *n* cuando quedaban x_n millones por asignar para la etapa *n+1*.

$c_n(x_n, d_n(x_n))$: Utilidad obtenida por tomar la alternativa $d_n(x_n)$ cuando quedaban x_n millones por asignar para la etapa *n+1*.

$f_n(x_n, d_n(x_n))$: Utilidad total obtenido por las alternativas óptimas las zonas *1, 2,..., n*

$d_n^*(x_n)$: alternativa óptima dado el estado x_n de la etapa *n.*

$f_n^*(x_n) = f(x_n, d_n^*(x_n)) = \max_{d_n(x_n)} \{f_n(x_n, d_n(x_n))\}$

IMPORTANTE: Darse cuenta que en esta situación los estados dependen de las etapas anteriores como sigue:

$$x_n = x_{n-1} - \cos to(x_n, d_n(x_n)) \text{ } o \text{ } equivalentemente \text{ } x_{n-1} = x_n + \cos to(x_n, d_n(x_n))$$

ETAPA 1: $f_1(x_1, d_1(x_1)) = c_1(x_1, d_1(x_1))$ y $f_1^*(x_1) = f_1^*(x_1, d_1^*(x_1)) = \max_{d_1(x_1)} \{f_1(x_1, d_1(x_1))\}$

ETAPA 1 (Zona A)						
x_1	$c_1(x_1, d_1(x_1))$			$d_1^*(x_1)$	$f_1^*(x_1)$	$x_0^* = x_1 + \cos to(d_1^*(x_1))$
	1	***2***	***3***			
0						
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

ETAPA 2: $f_2(x_2, d_2(x_2)) = c_2(x_2, d_2(x_2)) + f_1^*(x_2 + \cos to(d_2(x_2)))$

ETAPA 2 (Zona B)							
x_2					$d_2^*(x_2)$	$f_2^*(x_2)$	$x_1^* = x_2 + \cos to(d_2^*(x_2))$
	1	***2***	***3***	***4***			
0							
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							

ETAPA 3: $f_3(x_3, d_3(x_3)) = c_3(x_3, d_3(x_3)) + f_2^*(x_3 + \cos to(d_3(x_3)))$

ETAPA 3 (Zona C)					
x_3	$c_3(x_3, d_3(x_3)) + f_2^*(x_3 + \cos to(d_3(x_3)))$		$d_3^*(x_3)$	$f_3^*(x_3)$	$x_2^* = x_3 + \cos to(d_3^*(x_3))$
	1	***2***			
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

Política óptima: $x_3^* \mapsto x_2^* \mapsto x_1^*$, ***Valor total:*** $f_3(x_3, d_3^*(x_3))$

Etapa 3		***Etapa 2***	***Etapa 1***
x_3=0	***1 (0)***	***4(8)***	***2 (10)***
x_3=0	***2 (1)***	***2 (6)***	***3 (10)***
		3 (7)	***2 (9) (Gasto 9-0=9)***
x_3=1	***2 (2)***	***3 (8)***	***2 (10) (Gasto 10-1=9)***

OBSERVACIONES:

- La función objetivo es maximizar la utilidad obtenida.
- Existe una serie de restricciones sobre los estados y las decisiones
- Darse cuenta que se podría resolver el problema con programación dinámica, de salida a entrada, análogo al problema de la mochila. Basta considerar los estados como el dinero que falta por invertir. Como es lógico se obtendrá las mismas soluciones. ¡Inténtalo¡ Resuelva también el problema de la mochila de entrada a salida.

NOTA: Es muy difícil dar una regla que indique cual es el método más apropiado para resolver la función recursiva de un problema dinámico. Se han analizado problemas los cuales pueden ser resueltos indiferentemente de distintas maneras. En cambio hay otros en donde la dirección de la solución es crucial. Se recomienda que si resulta difícil en una dirección, se trate de la otra.

El siguiente ejemplo, se caracteriza en el hecho de que su relación recursiva es minimizar un producto (no lineal) de términos para las etapas respectivas. Aunque se trate con probabilidades, se puede resolver con programación dinámica determinista pues el estado en la siguiente etapa queda completamente determinado por el estado y la política de decisión en la etapa actual.

EJEMPLO 2: Un proyecto de investigación de la empresa de computadores *IBM* lleva a cabo el estudio sobre cierto problema de arquitectura informática que debe resolver. Por el momento, tres equipos de investigación independientes tratan de resolver el problema desde 3 punto de vista diferentes. Se estima que en las circunstancias actuales, la probabilidad de que los respectivos equipos 1,2 y 3 fracasen es de 0.4, 0.6 y 0.8 respectivamente. Así, la probabilidad actual de que los 3 equipos fracasan es de 0.192. Como el objetivo es minimizar la probabilidad de fracaso, se asignarán al proyecto dos científicos más de alto nivel. La probabilidad estimada de que los equipos respectivos fracasen si se les asigna 0, 1 o 2 científicos con ellos son dados en la siguiente tabla:

Científicos	***Grupo 1***	***Grupo 2***	***Grupo 3***
0	***0.4***	***0.6***	***0.8***
1	***0.2***	***0.4***	***0.5***
2	***0.15***	***0.2***	***0.3***

Determinar como deben asignarse los dos científicos adicionales para minimizar la probabilidad de fracaso en los tres equipos.

SOLUCIÓN: Este problema se puede plantear como uno de programación dinámica que implica tres etapas, tantos como grupos de investigación. En la etapa 1 se debe decidir cuántos científicos incorporar al grupo investigador 1, en la etapa 2 se decide sobre el

número de científicos que se van a incorporar en el grupo 2 y en la última etapa se decide los científicos del grupo 3. Darse cuenta que no influye la enumeración de los grupos de investigación.

Etapa 3 *Etapa 2* *Etapa 1*

Grupo 3 Grupo 2 Grupo 1

Determinación de la política óptima para

Se podría resolver de entrada a salida.

Notación:

x_n : Número de científicos que restan por asignar en la etapa *n* cuando no se ha asigna al grupo *n*.

$d_n(x_n)$: Número de científicos a asignar al grupo *n* cuando quedan x_n científicos.

$c_n(x_n, d_n(x_n))$: Probabilidad de fracaso por tomar la decisión de asignar $d_n(x_n)$ científicos del *n* en el estado x_n.

$f_n(x_n, d_n(x_n))$: Probabilidad de fracaso total obtenido por asignar el número óptimo de científicos del los grupos *n, n-1,..., 1* si restan x_n científicos por asignar.

$d_n^*(x_n)$: decisión óptima dado el estado x_n de la etapa *n*.

$$f_n^*(x_n) = f(x_n, d_n^*(x_n)) = \min_{d_n(x_n)} \{f_n(x_n, d_n(x_n))\}$$

IMPORTANTE: Darse cuenta que en esta situación los estados dependen de las etapas posteriores como sigue:

Etapa	***Estados***
3	$x_3 = 2$
2	$x_2 = x_3 - d_3(x_3)$
1	$x_1 = x_2 - d_2^*(x_2)$
0	$x_0 = x_1 - d_1^*(x_1)$

ETAPA 1: $f_1(x_1, d_1(x_1)) = p_1(x_1, d_1(x_1))$ y $f_1^*(x_1) = f_1^*(x_1, d_1^*(x_1)) = \min_{d_1(x_1)} \{f_1(x_1, d_1(x_1))\}$

ETAPA 1 (Grupo 1)						
x_1	$f_1(x_1, d_1(x_1))$			$d_1^*(x_1)$	$f_1^*(x_1)$	x_0^*
	0	***1***	***2***			
0	***0.4***	***-***	***-***	***0***	***0.4***	***0***
1	***0.4***	***0.2***	***-***	***1***	***0.2***	***0***
2	***0.4***	***0.2***	***0.15***	***2***	***0.15***	***0***

ETAPA 2: $f_2(x_2, d_2(x_2)) = p_2(d_2(x_2)) \times f_1^*(x_2 - d_2(x_2))$

ETAPA 2 (Grupo 2)						
x_2	$p_2(x_2, d_2(x_2)) \times f_1^*(x_2 - d_2(x_2))$			$d_2^*(x_2)$	$f_2^*(x_2)$	$x_1^* = x_2 - d_2^*(x_2)$
	0	***1***	***2***			
0						
1						
2						

ETAPA 3: $f_3(x_3, d_3(x_3)) = p_3(d_3(x_3)) \times f_2^*(x_3 - d_3(x_3))$

ETAPA 3 (Grupo 3)						
x_3	$p_3(x_3, d_3(x_3)) \times f_2^*(x_3 - d_3(x_3))$			$d_3^*(x_3)$	$f_3^*(x_3)$	$x_2^* = x_3 - d_3^*(x_3)$
	0	***1***	***2***			
2						

Política óptima: $x_3^* \mapsto x_2^* \mapsto x_1^* \mapsto x_0^*$, ***Valor total:*** $f_3(x_3, d_3^*(x_3))$

INFLUENCIA DEL DINERO EN EL TIEMPO: En ocasiones cuando se considera obtener el valor óptimo de cierto valor, se tiene que tener en cuenta dicho valor a lo largo del tiempo, sobre todo si el número de etapas es excesivamente grande. En el siguiente ejemplo, se pone de manifiesto:

EJEMPLO: El dueño de un lago debe decidir cuánto róbalo pescar y vender cada año. Si vende *x* róbalos durante un año *t*, su utilidad es *r(x).* El costo de pescar *t* róbalos durante un año es una función *c(x, b)* del número de peces atrapados durante el año y de *b*, el número de róbalos en el lago al principio del año. Naturalmente, los róbalos se reproducen. Para hacer un modelo, suponemos que el número de róbalos en el lago al principio de un año es 20% más que el que había al final del año anterior. Supongamos que hay 10000 róbalos en el lago al principio del primer año. Deduzca una fórmula recursiva de programación dinámica que se pueda usar para elevar al máximo las utilidades netas del propietario en un horizonte de *T* años.

SOLUCIÓN: El pescador tiene que maximizar sus beneficios, luego la fórmula recursiva tiene que ser de maximizar. Al principio del año *T*, el dueño del lago no debe preocuparse del efecto que tendrá la captura de róbalos sobre la población futura del lago. Por lo tanto, al principio del año *T* es fácil resolver el problema. Por esta razón haremos que el tiempo sea la etapa. En cada etapa, el dueño del lago debe decidir cuánto róbalo pescar. Definimos x_t como el número de róbalos pescados durante el año *t.* Para determinar un valor óptimo de x_t, el dueño del lago sólo necesita conocer el número de róbalos que hay en el lago al principio del año *t* (llamémosle b_t). Por lo tanto, el estado al principio del año *t* es b_t. Por ello, se define $f_t(b_t)$ como las utilidades netas máximas que se puede alcanzar de la pesca del róbalo durante los años *t, t+1, ..., T,* dado que hay

b_t róbalos en el lago al principio del año *t.* A continuación escribimos la fórmula recursiva, teniendo en cuenta los 3 aspectos de la formulación, para resolver el problema.

Tendremos en cuenta las siguientes restricciones:

- Durante cualquier año, no se puede pescar más róbalo del que hay disponibles en el lago: $0 \le x_t \le b_t$
- Si se pescan x_t róbalos cuando había b_t, entonces la ganancia neta será: $r(x_t) - c(x_t, b_t)$. Nótese que depende de la decisión y del estado de la etapa.
- Al final de la etapa *t* hay $b_t - x_t$ róbalos en el lago, por tanto el estado en la etapa *t+1* será $1.2(b_t - x_t)$.

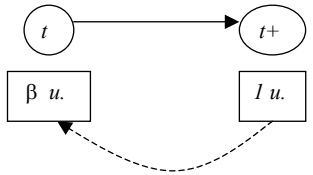
Con todo ello se tiene las siguientes funciones objetivos:

$$f_T(b_T) = \max_{x_T} \{r_T(x_T) - c(x_T, b_T)\}$$

$$f_t(b_t) = \max_{x_t} \{r(x_t) - c(x_t, b_t) + f_{t+1}(1.2(b_t - x_t))\}$$

donde $b_t \in \{0,...,10000(1.2)^{t-1}\}$

IMPORTANTE: Dado que maximizamos el beneficio total obtenido hasta el tiempo *T*, podría ser de interés incorporar o tener en cuenta el valor del dinero a través del tiempo. Para ello, darse cuenta que las ganancias recibidas durante los años futuros tienen ahora un valor inferior. Definimos por $\beta < 1$ una cantidad positiva que representa el valor en la etapa *t* de una unidad monetaria en la etapa *t+1,* tiempo posterior, obteniéndose por tanto las siguientes fórmulas recursivas



$$f_t(b_t) = \max_{x_t} \{r(x_t) - c(x_t, b_t) + \beta f_{t+1}(1.2(b_t - x_t))\}$$

Darse cuenta que β^T es el valor actual de una unidad monetaria dentro de *T* años.

PROGRAMACIÓN DINÁMICA PROBABILÍSTICA: En los ejemplos de programación dinámica determinista es suficiente una especificación del estado y de la decisión actual para determinar con certeza el estado nuevo y los costos durante el estado actual. En muchos problemas prácticos es posible que no se conozca con certeza el costo del periodo actual o el del siguiente periodo, aún si se conoce el estado y la decisión actual. Aún conocido el estado actual del periodo, así como la decisión, el estado del siguiente periodo y el costo del periodo actual serán variables aleatorias cuyos valores no se conocen hasta que se conozca el valor del estado del periodo *t*.

La programación dinámica probabilística es utilizada en la resolución de problemas en los que el costo del periodo actual o el estado del siguiente periodo son aleatorios. La meta del tomador de decisiones es minimizar el *costo incurrido esperado*

o maximizar la *recompensa esperada* que se gana en un determinado horizonte de tiempo.

COSTOS INCIERTOS EN LA ETAPA ACTUAL, PERO DETERMINADO EL ESTADO DEL SIGUIENTE PERIODO: En este tipo de problemas se conoce con certeza el estado del siguiente periodo, pero no es conocido la recompensa que se gana durante el periodo actual, dado el estado y la decisión actuales.

EJEMPLO (Asignación de recursos): La cadena de supermercado Safeco, compra a una lechería local a un precio de 1 euro/galón, 6 galones de leche. Cada galón se vende en las tres tiendas de la cadena a 2 euro/galón. La lechería compra la leche sobrante a 0.5 euro/galón al término del día. Desafortunadamente para Safeco, es incierta la demanda en cada una de sus tres tiendas. Los datos acumulados indican que la demanda diaria en cada tienda es como la que se muestra en la siguiente tabla:

	Demanda diaria (galones)	***Probabilidad***
Tienda 1	*1*	*0.6*
	3	*0.4*
Tienda 2	*1*	*0.5*
	2	*0.4*
	3	*0.1*
Tienda 3	*1*	*0.4*
	2	*0.3*
	3	*0.3*

Safeco desea asignar los 6 galones de leche a las tres tiendas para maximizar la ganancia neta diaria (ingresos menos costos) que da la leche. Mediante la programación dinámica determine como debe distribuir los 6 galones entre las tiendas.

SOLUCIÓN: En primer lugar observar que como los costos diarios de compra de Safeco siempre son 6 euros, pues se reparten los 6 galones de leche, podemos concretar nuestra atención al problema de asignar la leche para elevar al máximo la ganancia diaria esperada debida a los 6 galones. Para obtener la ganancia neta diaria, bastaría restar a esa cantidad 6 euros

Con excepción del hecho de que la demanda es incierta, el problema es muy semejante a los problemas de asignación de recursos. Con el fin de resolver el problema con programación dinámica se define:

x_t : Galones disponibles para la tienda *t,..,1*.

$d_t(x_t)$: Galones asignados a la tienda *t*.

$r_t(x_t)$: ganancia *esperada* de x_t galones asignados a la tienda *t.*

$f_t(x_t)$: ganancia máxima *esperada* de x_t galones asignados a las tiendas *t, t - 1, ..,1*.

Por tanto se tiene las siguientes fórmulas recursivas:

$$f_1(x) = r_1(x)\,;\; f_t(x_t) = \max_{d_t(x_t)\le x_t} \{r_t(d_t(x_t)) + f_{t-1}(x_t - d_t(x_t))\}$$

Etapa 3 *Etapa 2* *Etapa 1*

Tienda 3 Tienda 2 Tienda 1

Determinación de la política óptima para

IMPORTANTE: Darse cuenta que en esta situación los estados dependen de las etapas posteriores como sigue:

Etapa	***Estados***
3	$x_3 = 6$
2	$x_2 = x_3 - d_3(x_3)$
1	$x_1 = x_2 - d_2(x_2)$
0	$x_0 = x_1 - d_1(x_1)$ *(Galones por asignar)*

Vamos a determinar $r_t(x_t)$, la ganancia *esperada* de x_t galones asignados a la tienda *t.*

$$r_t(x_t) = \begin{cases} 0, \; si\; x_t = 0 \\ p_1 \times 2 \times 1 + p_2 \times 2 \times 1 + p_3 \times 2 \times 1, \; si\; x_t = 1 \\ p_1 \times (2 \times 1 + 0.5 \times 1) + p_2 \times 2 \times 2 + p_3 \times 2 \times 2, \; si\; x_t = 2 \\ p_1 \times (2 \times 1 + 0.5 \times 2) + p_2 \times (2 \times 2 + 0.5 \times 1) + p_3 \times 2 \times 3, \; si\; x_t = 3 \end{cases}$$

Estados	***Tienda 1***	***Tienda 2***	***Tienda 3***
0	***0***	***0***	***0***
1	***2***	***2***	***2***
2	***3.10***	***3.25***	***3.4***
3	***4.20***	***4.35***	***4.35***

ETAPA 1: $f_1(x_1, d_1(x_1)) = r_1(d_1(x_1))$ y $f_1^*(x_1) = f_1^*(x_1, d_1^*(x_1)) = \max_{d_1(x_1)} \{f_1(x_1, d_1(x_1))\}$

ETAPA 1 (Tienda 1)							
x_1	$f_1^*(x_1)$				$d_1^*(x_1)$	$f_1^*(x_1)$	$x_0^* = x_1 - d_1^*(x_1)$
	0	***1***	***2***	***3***			
0							
1							
2							
3							
4							
5							
6							

ETAPA 2: $f_2(x_2, d_2(x_2)) = r_2(d_2(x_2)) + f_1^*(x_2 - d_2(x_2))$

ETAPA 2 (Tienda 2)							
x_2	$c_2(x_2, d_2(x_2)) + f_1^*(x_2 - d_2(x_2))$				$d_2^*(x_2)$	$f_2^*(x_2)$	$x_1^* = x_2 - d_2^*(x_2)$
	0	***1***	***2***	***3***			
0							
1							
2							
3							
4							
5							
6							

ETAPA 3: $f_3(x_3, d_3(x_3)) = r_3(d_3(x_3)) + f_2^*(x_3 - d_3(x_3))$

ETAPA 3 (Tienda 3)							
x_3	$r_3(x_3, d_3(x_3)) + f_2^*(x_3 - d_3(x_3))$				$d_3^*(x_3)$	$f_3^*(x_3)$	$x_2^* = x_3 - d_3^*(x_3)$
	0	***1***	***2***	***3***			
6							

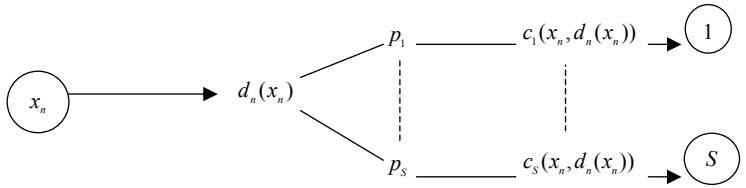
Política óptima: $x_3^* \mapsto x_2^* \mapsto x_1^* \mapsto x_0^*$, ***Valor total:*** $f_3(x_3, d_3^*(x_3))$

Tienda 3	***Tienda 2***	***Tienda 1***
2	***2***	***2***
	3	***1***

NOTA: Aunque con esta política se obtiene la ganancia máxima esperada máxima de 9.75 euros, la ganancia total que se recibe en realidad en un día determinado puede ser mayor o menor que 9.75 euros. Por ejemplo si la demanda en cada tienda fuera de 1 galón, la ganancia total sería de 7.5 euros. Así, si todo se vende, se obtiene una ganancia de 12 euros.

CARACTERÍSTICAS GENERALES: Como ya hemos señalado anteriormente, la programación dinámica probabilística difiere de la determinista en que el valor o estado en la siguiente etapa no está completamente determinado por el estado y política de decisión de la etapa actual. En su lugar existe una distribución de probabilidad para determinar cuál será el siguiente estado.

Etapa ***Decisión*** ***Probabilida*** ***Valor de la decisión*** ***Etapa n-***



Dirección para la determinación de la

Debido a la estructura probabilística, la relación entre $f_n(x_n, d_n(x_n))$ y $f_{n-1}^*(x_{n-1}, d_{n-1}(x_{n-1}))$ necesariamente será más complicado que para el caso determinista, obteniéndose los siguiente:

$$f_n(x_n, d_n(x_n)) = \sum_{i=1}^{S} p_i(c_i + f_{n-1}^*(i))$$

SUCESOS FAVORABLES: En ocasiones es de interés obtener una estrategia óptima tal que maximice la probabilidad de obtener cierto suceso, llamado favorable. Para poner de manifiesto estos tipos de problemas, considérese los siguientes dos ejemplos.

EJEMPLO 1: Un jugador propone un método para ganar un popular juego en las Vegas. Sus colegas no piensan que este sistema sea tan bueno, por lo que le apuestan que si comienzan con tres fichas, no tendrá al menos cinco ficha después de tres jugadas. Cada jugada incluye apostar cualquier cantidad de las fichas disponibles y ganar o perder este mismo número de fichas. Se piensa que el sistema dará una probabilidad de *2/3* de ganar una jugada.

Suponiendo que se está en lo cierto, se quiere usar programación dinámica para determinar su política óptima sobre cuántas fichas apostar (si apuesta) en cada una de las 3 jugadas. La decisión en cada jugada deberá tomar en cuenta los resultados de las jugadas anteriores. El objetivo es maximizar la probabilidad de ganar la apuesta hecha a sus colegas.

SOLUCIÓN: Con el fin de resolver el problema aplicando programación dinámica probabilística, distinguiremos 3 etapas que corresponderán a las distintas jugadas.

x_n : Número de fichas disponibles en la jugada *n*

$d_n(x_n)$: Número de fichas que se apuestan en la jugada *n*

$f_n(x_n, d_n(x_n))$: Probabilidad de terminar las n,...,3 jugadas con 5 o más fichas

Si pierde la apuesta en el juego *n,* en la siguiente etapa tendrá disponible $x_n - d_n(x_n)$ fichas y la probabilidad de terminar con al menos cinco fichas será $f_{n+1}^*(x_n - d_n(x_n))$. Por el contrario, si gana la siguiente jugada, el estado del sistema será $x_n + d_n(x_n)$ y $f_{n+1}^*(x_n + d_n(x_n))$ la probabilidad máxima de ganar la apuesta.

$$f_n(x_n, d_n(x_n)) = \frac{1}{3} f^*_{n+1}(x_n - d_n(x_n)) + \frac{2}{3} f^*_{n+1}(x_n + d_n(x_n))$$

ETAPA 1: $f_3(x_3, d_3(x_3))$

ETAPA 1 (Jugada 3)		
x_3	$f^*_3(x_3)$	$d^*_3(x_3)$
0		
1		
2		
3		
4		
≥ 5		

ETAPA 2: $f_2(x_2, d_2(x_2)) = \frac{1}{3} f^*_3(x_2 - d_2(x_2)) + \frac{2}{3} f^*_3(x_2 + d_2(x_2))$

ETAPA 2 (Juego 2)							
x_2	$\frac{1}{3} f^*_3(x_2 - d_2(x_2)) + \frac{2}{3} f^*_3(x_2 + d_2(x_2))$					$d^*_2(x_2)$	$f^*_2(x_2)$
	0	***1***	***2***	***3***	***4***		
0							
1							
2							
3							
4							
≥ 5							

ETAPA 3: $f_1(x_1, d_1(x_1)) = \frac{1}{3} f^*_2(x_1 - d_1(x_1)) + \frac{2}{3} f^*_2(x_1 + d_1(x_1))$

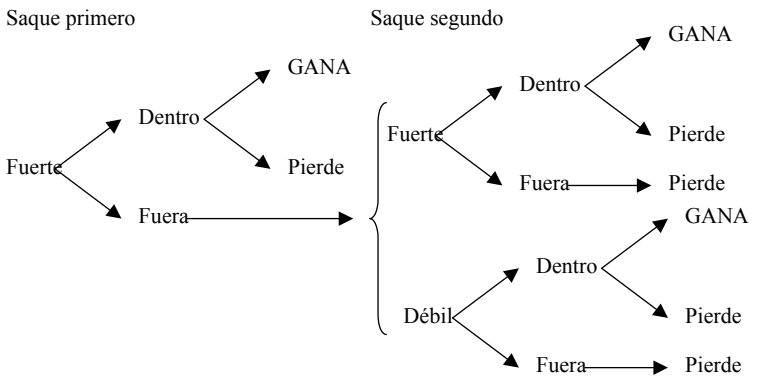
ETAPA 3 (Jugada 1)						
x_1	$\frac{1}{3} f^*_2(x_1 - d_1(x_1)) + \frac{2}{3} f^*_2(x_1 + d_1(x_1))$				$d^*_1(x_1)$	$f^*_1(x_1)$
	0	***1***	***2***	***3***		
3						

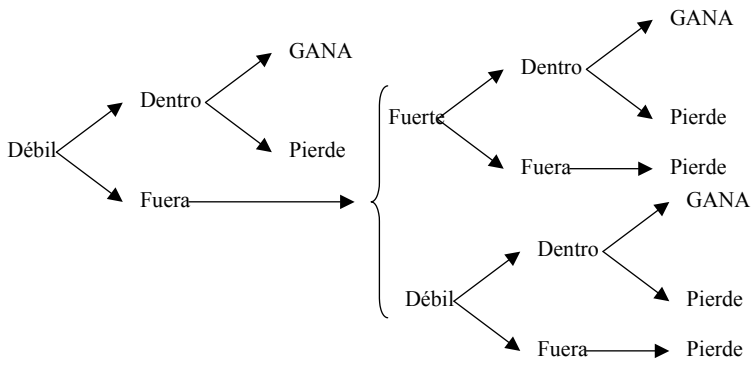
Política óptima: $x^*_1 \mapsto x^*_2 \mapsto x^*_3$, ***Valor total:*** $f_1(x_1, d^*_1(x_1))$

Jugada 1	***Jugada 2***	***Jugada 3***	***Apuesta***
1 Gana	***1 Gana***	***0***	***GANA***
	1 Pierde	***2 o 3 Gana***	***GANA***
		2 o 3 Pierde	***PIERDE***
1 Pierde	***1 Gana***	***2 o 3 Gana***	***GANA***
		2 o 3 Pierde	***PIERDE***
	1 Pierde	***-------***	***PIERDE***
	2 Gana	***1,2,3 o 4 Gana***	***GANA***
		1,2,3 o 4 Pierde	***PIERDE***
	2 Pierde	***-------***	***PIERDE***

EJEMPLO 2: Martina McEnroe tiene dos clases de saque: uno fuerte *(H)* y uno suave *(S)*. La probabilidad de que el saque fuerte caiga dentro es p_H, y la probabilidad de que el saque suave caiga dentro es p_S. Si el saque fuerte de Martina cae dentro, hay una probabilidad w_H que gane el punto. Si su saque suave cae dentro, hay una probabilidad w_S que gane el punto. Suponemos que $p_H < p_S$ y que $w_H > w_S$ (¿suposiciones lógicas?). La meta de Martina es maximizar la probabilidad de ganar un punto cuando sirve. Use la programación dinámica para ayudar a Martina a seleccionar una estrategia óptima de servicios. Recuérdese que si dos servicios caen fuera, Martina pierde el punto.

SOLUCIÓN: Para maximizar la probabilidad de Martina de ganar el punto, le daremos una recompensa de 1 si lo gana y una recompensa 0 si la pierde. También definiremos f_t *(t = 1, 2)* como la probabilidades con la que gane el punto si juega en forma óptima y está a punto de servir el *t*-ésimo saque.





Para determinar la estrategia óptima de servicio, avanzaremos hacia atrás, iniciando con f_2. Si Martina sirve fuerte la segunda vez, ganará el punto y una recompensa de 1, con probabilidad $p_H w_H$ (probabilidad condicionada $p(A \cap B) = p(A/B) \cdot P(B)$). Igualmente si sirve suave la segunda vez, su recompensa esperada es $p_S w_S$. Entonces tenemos que:

$$f_2 = max \begin{cases} p_H w_H \text{ (saque fuerte)} \\ p_S w_S \text{ (saque suave)} \end{cases}$$

por el momento supondremos que $p_S w_S > p_H w_H$. En este caso Martina debe sacar suave en el segundo servicio, es decir $f_2 = p_S w_S$.

Para determinar f_1 necesitamos ver la que sucede en el primer servicio. Si Martina saca fuerte en el primer servicio, gana una recompensa esperada igual a $p_H w_H + (1 - p_H) f_2$, pues la probabilidad de que el primer servicio cae dentro y Martina gana el punto es $p_H w_H$ con recompensa de 1, si lo pierde no gana recompensa, con probabilidad $1 - p_H$ el primer servicio cae fuera, el cual gana una recompensa de f_2 (jugar el segundo saque). Si Martina saca suave en el primer servicio, gana una recompensa esperada igual a $p_S w_S + (1 - p_S) f_2$, pues la probabilidad de que el primer servicio cae dentro y Martina gana el punto es $p_S w_S$ con recompensa de 1, si lo pierde no gana recompensa, con probabilidad $1 - p_H$ el primer servicio cae fuera, el cual gana una recompensa de f_2 (jugar el segundo saque). Así pues tenemos:

$$f_1 = max \begin{cases} p_H w_H + (1 - \mathrm{p_H}) \mathrm{f_2} \text{ (saque fuerte)} \\ p_S w_S + (1 - \mathrm{p_S}) \mathrm{f_2} \text{ (saque suave)} \end{cases}$$

Según esta ecuación, Martina debe sacar fuerte en el primer saque si:

$$p_H w_H + (1 - \mathrm{p_H}) \mathrm{f_2} \geq p_S w_S + (1 - \mathrm{p_S}) \mathrm{f_2}$$

en caso contrario, debe sacar suave, en el primer servicio. Como hemos supuesto que $p_S w_S > p_H w_H$, lo cual implica $f_2 = p_S w_S$, sustituyendo en la ecuación anterior nos queda:

$$p_H w_H \geq p_S w_S (1 + \mathrm{p_H} - \mathrm{p_S})$$

Dependiendo de los valores de las incógnitas, Martina sacará fuerte o suave el primer servicio. Para completar nuestro análisis, debemos tener en cuenta el caso en el que se verifica $p_S w_S \leq p_H w_H$. En esta ocasión, $f_2 = p_H w_H$, por tanto debe servir fuerte en el segundo saque. Para el primer saque, aplicando la fórmula y simplificando tenemos que debe servir fuerte si:

$$p_H w_H (1 + \mathrm{p_S} - \mathrm{p_H}) \geq p_S w_S$$

Al dividir ambos lados de la desigualdad entre $p_S w_S$, se demuestra que Martina debe servir fuerte la primera vez si:

$$\frac{p_H w_H (1 + \mathrm{p_S} - \mathrm{p_H})}{p_S w_S} \geq 1$$

Puesto que $p_H w_H \geq p_S w_S$ y $(1 + \mathrm{p_S} - \mathrm{p_H}) \geq 1$, ya que $p_S > p_H$, vemos que siempre se verifica la última desigualdad, es decir debe sacar fuerte en ambos servicios. Esto es razonable, pues si es óptimo sacar fuerte en el segundo servicio donde tiene más riesgo, debería ser óptimo sacar fuerte también el primero, porque el peligro de doble falta, que es el inconveniente del saque fuerte, es menor a continuación del primer servicio.

NOTA: Para usar la programación dinámica en un problema en el que la etapa está representada por el tiempo, se debe determinar el valor *T,* el número de etapas sobre las cuales se maximiza la ganancia esperada, o se minimiza los costos esperados. *T* se *llama amplitud del horizonte o longitud del horizonte*. En ocasiones, resulta difícil para quien toma decisiones determinar exactamente la amplitud del horizonte más adecuado. En realidad, cuando un tomador de decisiones encara un horizonte amplio y no está seguro de la amplitud, es más cómodo suponer que la amplitud del horizonte es infinita. Los problemas de programación dinámica probabilística con horizonte infinito se llaman ***procesos de decisión de Markov*** que se tratará en el Capítulo 5.