Programación Dinámica

Manuel Maurette e Ignacio Ojea

Junio de 2006

Agradecimientos:

Nuestro reconocimiento a la Dra. Susana Puddu, por su compromiso con la labor docente y con los alumnos y, especialmente, al Dr. Fabio Vicentini por su tesón en la difusión de la matemática aplicada y la generosidad con que brinda sus conocimientos a los estudiantes.

Índice

1. Introducción

1.1. Resumen del Trabajo

La Programación Dinámica es un método de optimización de extraordinaria versatilidad. Si bien fue desarrollada especialmente para la resolución de problemas en Procesos de Decisión en Múltiples Pasos, diferentes investigaciones han mostrado que las mismas ideas pueden utilizarse en otro tipo de problemas de matemática aplicada, e incluso pueden ser útiles en el planteo de algunas cuestiones teóricas. Habiendo surgido en los inicios de la época de las computadoras, la Programación Dinámica fue, además, concebida con un ojo puesto en esta potente herramienta. La *Ecuación Funcional* que se obtiene, para cada problema, a través del uso del *Principio de Optimalidad* de Bellman permite, con mayor o menor esfuerzo dependiendo del caso, establecer una recurrencia que es, en sí misma, un algoritmo que resuelve el problema en cuestión.

El objetivo de esta monografía es brindar un panorama relativamente amplio de las aplicaciones de la Programación Dinámica, de manera que resulte accesible para cualquier estudiante de Licenciatura, incluso para aquellos que no estén familiarizados con las áreas específicas de dichas aplicaciones. Persiguiendo este fin, procuramos, en la medida en que el espacio lo permitió, exponer todos los pasos de cada razonamiento y los elementos teóricos básicos para su comprensión.

Atendiendo a la utilidad principal de la Programación Dinámica, esto es: la resolución de problemas aplicados con el auxilio de las computadoras; nuestro trabajo se centra en la exposición y resolución de algunos ejemplos clásicos, a través de los cuales intentamos mostrar las ideas que pone en juego la técnica de la Programación Dinámica, su versatilidad y, también, sus limitaciones.

Teniendo en cuenta que temas como, por ejemplo, el Cálculo de Variaciones o los Procesos Estocásticos, difícilmente sean abordados en las materias regulares de una carrera de Licenciatura, preferimos dar prioridad a los problemas discretos y determinísticos, que requieren menos conocimientos teóricos previos para su comprensión, y dejar las aplicaciones de la Programación Dinámica en estas áreas para el final. Dadas las limitaciones de extensión, debimos, a nuestro pesar, reducir los últimos temas abordados a su mínima expresión.

1.2. Historia

Durante la Segunda Guerra Mundial la investigación matemática se extendió hacia zonas que hasta entonces le habían sido ajenas. Si bien la participación de la ciencia, y de la matemática en particular, en los enfrentamientos bélicos, puede remontarse a la organización, por parte de Arquímedes, de las defensas de Siracusa, lo cierto es que, hasta la Segunda Guerra, no habían existido políticas consecuentes de aplicación específica de la matemática a problemas de importancia en esta materia.

En realidad, este fenómeno comenzó en los años previos al estallido de la guerra. Alemania, Inglaterra, Estados Unidos y la U.R.S.S. formaron equipos de investigación, cuyos trabajos fueron la base de muchos de los inventos que aparecieron en funcionamiento durante la guerra (el radar, por ejemplo) y que abrieron las nuevas ramas de la matemática que se desarrollarían enormemente después de 1945.

La primera gran disciplina que surgió a partir del abordaje matemático de los problemas específicos de la guerra fue, seguramente, la Investigación Operativa[1]. El término *Operations Research* fue utilizado por primera vez en Inglaterra, en 1941. Las investigaciones realizadas en los centros de Investigación Operativa de la Royal Air Force y otros organismos militares británicos permitieron, entre otras cosas, incrementar la eficacia de la los patrullajes aéreos en busca de submarinos alemanes, y consecuentemente, la cantidad de submarinos dañados o hundidos.

Rápidamente se hizo evidente que las mismas técnicas utilizadas en el ámbito militar podían servir en otras áreas de aplicación. En los años posteriores a la Guerra se abrieron nuevos temas de investigación y se plantearon nuevos problemas, que fueron abordados desde una perspectiva matemática. Entre estos nuevos temas se encontraba la teoría de los Procesos de Decision en Múltiples Pasos, que Richard Bellman (1920 - 1984) abordó alrededor de 1952, y para los cuales fue pensada originalmente la Programación Dinámica.

Después de desarrollar el método en el área específica de los problemas de decisión discretos, Bellman y sus colaboradores se dedicaron a la ardua tarea de formular diferentes problemas en los términos de la Programación Dinámica. Como resultado de esta labor, encontraron que las ideas centrales del método, en particular, el Principio de Optimalidad, podían ser aplicadas satisfactoriamente en muchos de los problemas abordados. Descubrieron también las limitaciones de esta técnica y hallaron modos de sobreponerse a ellas, para algunos problemas puntuales.

La Programación Dinámica es, hoy en día, un recurso imprescindible de Matemática Aplicada y, también, una importante herramienta teórica.

[1] En rigor, una traducción más exacta sería *Investigación en Operaciones*.

2. Programación Dinámica Discreta

2.1. El Problema del Camino de Mínimo Costo

2.1.1. Grafos

Llamamos *grafo* a un par de conjuntos V y E, de los cuales el primero contiene los *vértices* o *nodos*, mientras que el segundo es el conjunto de las *ramas* o *arcos* y está formado por pares de elementos de V que consideramos conectados entre sí. La notación usual para un grafo es $G = (V, E)$.

Nos interesa particularmente considerar el caso en que los elementos $(u, v) \in E$ son pares *ordenados*. Cuando esto sucede, el grafo se dice *dirigido* y las ramas se notan: $u \to v$. A modo de ejemplo, La Fig. 1 representa un grafo en donde $V = \{1, 2, 3, 4\}$ y $E = \{(1, 2); (4, 2); (3, 1)\}$.

Figura 1: Ejemplo de Grafo

En un grafo dirigido un camino del vértice u al vértice v es una sucesión de ramas $u \to u_1, u_1 \to u_2, ..., u_{k-1} \to v$ que conectan u con v.

Por último, un grafo se dice acíclico si no forma ciclos, es decir, si no existe ningún camino que comience y termine en el mismo vértice.

2.1.2. El problema

Sea $G = (V, E)$ un grafo dirigido y acíclico, donde cada arco $u \to v$ tiene asociado un costo $c_{uv} \in \mathbb{R}$, y donde el costo de un camino se computa sumando los costos de las ramas que lo componen.

El hecho de que sea acíclico implica que todo camino finaliza en un vértice del que no sale ninguna flecha, al que llamamos *terminal*. Dado un vértice cualquiera, $u \in V$, el problema consiste en hallar un camino de costo mínimo que parta de u y finalice en un vértice terminal. Un camino con estas características se llama *camino óptimo* que parte de u.

Esta formulación matemática sirve de modelo para diversos problemas. El ejemplo más sencillo es el de una red de carreteras que conectan una localidad de origen con otra de destino, pasando por varias localidades intermedias. En este caso, el *costo* de

un tramo de ruta podría ser su longitud. Es a partir de este ejemplo que el problema suele presentarse con el nombre de problema del camino más corto.

Debe tenerse en cuenta que tanto V como E pueden tener una enorme cantidad de elementos, por lo que la búsqueda de un camino de costo mínimo representa un auténtico problema. Por otra parte y como veremos luego en un ejemplo, una política *codiciosa* a corto plazo, que tome en cada nodo la rama que resulte menos costosa, no conduce, generalmente, a la construcción de un camino de costo mínimo.

Antes de encarar la resolución del problema, observemos que, puesto que existen finitos caminos, debe existir al menos uno de costo mínimo. Es decir: nuestro problema tiene solución.

2.1.3. El planteo con Programación Dinámica

Sea V_0 el conjunto de todos los vértices terminales de G. Consideremos ahora el grafo $G - V_0$ que resulta de eliminar de G los vértices de V_0 y las ramas que inciden en ellos. Sea ahora V_1 el conjunto de los vértices terminales de $G - V_0$. Análogamente, definimos:

$$V_i := \text{conjunto de los vértices terminales de } G - (\bigcup_{j=0}^{i-1} V_j)$$

Puesto que la cantidad de vértices es finita, existe un n para el cual $V = \bigcup_{i=0}^{n} V_i$, y como $V_i \cap V_j = \emptyset$ si $i \neq j$, concluimos que $\{V_j\}_{j=1}^{n}$ es una partición de V. Por otra parte:

$$u \in V_i \quad \wedge \quad u \to v \quad \Longrightarrow \quad v \in \bigcup_{j=0}^{i-1} V_j$$

Ahora bien, si

$$u \to v \to v_2 \to ... \to v_{k-1} \to v_k \qquad v_k \text{ terminal}$$

es un camino óptimo partiendo del vértice u, entonces, necesariamente,

$$v \to v_2 \to ... \to v_{k-1} \to v_k$$

es un camino óptimo partiendo del vértice v. Es decir: las *colas* de un camino óptimo son, a su vez, óptimas. Esta brillante, y aparentemente sencilla, observación recibe el nombre de *Principio de Optimalidad* y es la clave de la Programación Dinámica. Veamos cómo, a partir de ella, se llega a la solución del problema.

Sea $f : V \longrightarrow \mathbb{R}$ la función que asigna a cada vértice $u \in V$ el costo de un camino óptimo que parte de dicho vértice. El Principio de Optimalidad puede expresarse en términos de esta función, que es una incógnita de nuestro problema. En efecto, si, como antes, v es el nodo que sigue a u en un camino óptimo:

$$f(u) = c_{uv} + f(v)$$

Además, para cualquier otro w tal que $u \to w$ tendremos:

$$f(u) \leq c_{uw} + f(w)$$

Lo cual nos conduce a establecer la siguiente *ecuación funcional*:

$$f(u) = \min_{w:u \to w} c_{uw} + f(w)$$

Consideramos ahora la *política óptima*, esto es, la función $p : V \setminus V_0 \longrightarrow V \setminus V_n$ que asigna a cada vértice u el nodo v que le continúa en un camino óptimo; es decir, $p(u)$ es el argumento que realiza el mínimo en la *ecuación funcional*. Está función podría no estar bien definida, si para algún u existiera más de un camino óptimo. Sin embargo, si esto ocurriera, podríamos elegir arbitrariamente alguno de los vértices posibles, resolviendo el inconveniente. Cabe aclarar que para algunos problemas puede determinarse un criterio específico para esta elección.

Así planteado el problema, nuestro objetivo es hallar, para todo $u \in V$ los valores $f(u)$ y $p(u)$. La función f nos da el costo del camino mínimo, mientras que p nos permite construirlo, de la siguiente manera:

$$u \to p(u) = u_1 \to p(u_1) = u_2 \to ... \to p(u_{k-1}) = u_k \in V_0$$

La ecuación funcional nos permite encontrar f y p recursivamente y *de atrás para adelante*; en primer lugar tenemos:

$$f(u) = 0 \qquad \forall u \in V_0$$

Luego, si $u \in V_1$ y $u \to w$ entonces $w \in V_0$. Por lo tanto:

$$f(u) = \min_{w:u \to w} c(u, w) + f(w) = \min_{w:u \to w} c(u, w) \qquad \forall u \in V_1$$

$$p(u) = \arg\min_{w:u \to w} c(u, w) + f(w) = \arg\min_{w:u \to w} c(u, w) \qquad \forall u \in V_1$$

Y así, habiendo calculado $f(u)\ \forall u \in \bigcup_{j=0}^{i-1} V_j$ se tiene que:

$$f(u) = \min_{w:u \to w} c(u, w) + f(w) \qquad \forall u \in V_i$$

$$y(u) = \arg \min_{w:u \to w} c(u,w) + f(w)$$

El planteo recursivo no sólo permite hallar una solución del problema sino que constituye la base para el diseño de un algoritmo de implementación bastante sencilla.

Veamos como funciona el método en un ejemplo:

2.1.4. Ejemplo

Consideremos el grafo de la Fig. 2. El problema es hallar el camino óptimo desde el vértice inicial 1 hasta el terminal 7, donde los valores en las flechas indican el costo $c(u,v)$ de ir de un vértice u a otro v. Sean f y p las funciones definidas anteriormente.

Figura 2: Hallar el camino de mínimo costo.

En primer lugar, podemos definir los V_i fácilmente:

$$V_0 = \{7\}, V_1 = \{5,6\}, V_2 = \{4\}, V_3 = \{2,3\}, V_4 = \{1\}$$

Entonces, procedamos a calcular las funciones recursivamente, comenzando por el vértice terminal:

$$f(7) = 0$$

$$f(6) = c(6,7) = 2 \qquad p(6) = 7$$

$$f(5) = c(5,7) = 3 \qquad p(5) = 7$$

$$f(4) = \text{mín}\{c(4,5)+f(5), c(4,6)+f(6), c(4,7)+f(7)\} = \text{mín}\{1+3, 4+2, 6\} = 4 \qquad p(4) = 5$$

$$f(3) = \text{mín}\{c(3,4) + f(4), c(3,6) + f(6)\} = \text{mín}\{1+4, 5+2\} = 5 \qquad p(3) = 4$$

$$f(2) = \text{mín}\{c(2,4) + f(4), c(2,5) + f(5)\} = \text{mín}\{3+4, 5+3\} = 7 \qquad p(2) = 4$$

$$f(1) = \text{mín}\{c(1,2) + f(2), c(1,3) + f(3)\} = \text{mín}\{1+7, 2+5\} = 7 \qquad p(1) = 3$$

Notar que $f(1) = 7$ nos proporciona el costo del camino óptimo. El camino, propiamente dicho se obtiene con la función p :

$$1 \to p(1) = 3 \to p(3) = 4 \to p(4) = 5 \to p(5) = 7$$

Comparemos este algoritmo a otro, por ejemplo, un algoritmo codicioso, es decir uno que comenzando desde el primer vértice, elija el próximo siguiendo la rama de costo menor inmediato. En ese caso obtenemos:

$$1 \to 2 \to 4 \to 5 \to 7$$

Cuyo costo es 8, o sea mayor al mínimo. Otra manera de resolver este problema sería usando fuerza bruta, es decir, numerar todos los posibles caminos que parten del 1 y elegir el menor. Veamos que esto requiere excesivos cálculos. Para esto, deberíamos enumerar todos los caminos, $1 \to 2 \to 5 \to 7$, $1 \to 3 \to 4 \to 6 \to 7$, etc. En total son 8, 4 de 3 ramas y 4 de 4. A cada uno de estos habría que calcular su costo, es decir, la suma de las ramas que lo componen. Para los caminos que tiene 3 ramas harán falta 3 sumas y 4 para los de 4. En total hacen 28 sumas. Luego de tener todos, hay que comparar los 8 números y quedarse con el menor, usando el algoritmo *merge sort* se necesitarían, en el peor de los casos, 24 comparaciones (Con el *bubble sort*, que es el algoritmo básico para la comparación se necesitarían 28 comparaciones). Lo que hace un total de 52 operaciones básicas. Con el algoritmo de programación dinámica hicimos 9 sumas y 6 comparaciones. Es decir 15 operaciones básicas, más de un 70 % menos. A medida que crece la complejidad del problema más se ve el poder de la programación dinámica.

2.2. El método general

El problema de hallar el camino de costo mínimo en un grafo dirigido es un caso particular de lo que se llama Problema de Decisión en Múltiples Pasos (*Multistage Decision Process Problem*). La formulación general de este problema es la siguiente:

Un *proceso* es examinado periódicamente, a tiempos $t = 0, 1, ...$ Como resultado de dicho examen se obtiene el valor de una variable (o, eventualmente, un vector de variables) x que sirve para juzgar la situación del proceso. Al par $u = (t, x(t))$ lo llamamos un *estado*. Luego de cada observación de x debe ejecutarse una acción correctiva, tomada de un conjunto de posibles decisiones $D(u)$. La elección de una decisión particular $d(u)$ genera una *transformación* T que da como resultado un nuevo estado: $v = (t + 1, \tilde{x}) = T(u, d)$, y que tiene asociado un costo $c(u, d)$. Una función que indique para cada *estado* u una decisión específica a tomar recibe el nombre de *política*. Se quiere hallar una política de manera que la suma de los costos de las transformaciones engendradas por las sucesivas decisiones resulte mínima. A una política de estas características se la llama *política óptima*.

En el problema del camino de mínimo costo, los nodos del grafo juegan el papel de *estados*. A su vez, para cada $u \in V$, el conjunto $D(u)$ de las posibles *decisiones* está formado por los vértices v tales que $(u, v) \in E$, mientras que las *transformaciones* engendradas por una decisión son justamente las ramas $(u, v) \in E$, y el costo de la transformación es el costo de la rama. Una política es, en este caso, una regla que nos dice a qué nodo pasar si nos encontramos en un vértice dado cualquiera, siendo la política óptima aquella que nos da, para cada $u \in V$ un camino de costo mínimo

que finalice en un vértice terminal.

Si nos centramos en el caso en que el proceso finaliza en un tiempo N, la analogía entre el planteo general y el caso particular del problema del camino de mínimo costo en un grafo dirigido resulta inmediata, y nos permite pensar que las sucesivas transformaciones forman un camino, pasando de un estado a otro, en un tiempo posterior.

Figura 3: Problema de Control

Siguiendo esta idea, tenemos que, si llamamos d a la política utilizada, comenzando en un estado u_0 se forma un *camino* - como lo muestra la Figura 3- dado por:

$$u_1 = T(u_0, d(u_0))$$

$$u2 = T(u_1, d(u1))$$

$$...$$

$$u_{k+1} = T(u_k, d(u_k) \qquad u_k = (k, x(k))$$

$$...$$

$$u_N = T(u_{N-1}, d(u_{N-1}))$$

Continuando con la analogía, llamamos f_d a la función que asigna a cada estado $u = (t, x)$ el costo del *camino* hasta un estado terminal inducido por la política d.

La observación crucial sobre la que llamamos la atención en el ejemplo es la siguiente:

2.2.1. Principio de optimalidad

Una política óptima tiene la propiedad de que cualquiera sean el estado y la decisión iniciales, las decisiones siguientes constituyen una política óptima con respecto al estado resultante de la primera decisión.

Si llamamos f a la función de costo correspondiente a la política óptima, el principio de optimalidad puede expresarse de la siguiente manera:

$$f(u_t) = c(u_t, d(u_t)) + f(T(u_t, d(u_t)))$$

2.2.2. Ecuación Funcional

Al igual que en el ejemplo, el Principio de Optimalidad permite establecer la siguiente *Ecuación Funcional*:

$$f(u) = \min_{d \in D(u)} c(u, d(u)) + f_d(T(u, d(u)))$$

A partir de esta ecuación, puede resolverse el problema de manera recursiva, de un modo análogo al utilizado para el problema de camino de costo mínimo.

2.3. Ejemplos

Seguramente la mayor dificultad que se presenta al encarar la resolución de un problema de Optimización es la de encontrar una formulación matemática apropiada. En el caso de la Programación Dinámica, que se basa en el uso del Principio de Optimalidad para el planteo de la Ecuación Funcional, el elemento principal de una buena formulación es la noción de *estado*, íntimamente ligada a la idea de *secuencia*, puesto que, en casa paso del proceso se pasa de un estado a otro *posterior*, desde el cual no es posible regresar. Esta concepción *temporal* no siempre está implícita en el planteo verbal del problema. Encontrar la manera apropiada de expresarla puede representar un importante esfuerzo. Como se verá más adelante, para decidir qué es lo que debe considerarse *estado* es necesario determinar cuál es la información esencial con que se debe contar a cada paso para estar en condiciones de tomar una decisión. Esto haría pensar qué deben hallarse *primero* los sucesivos pasos y luego los estados. Sin embargo, el estrecho vínculo que une ambas nociones -la de estado y la de secuencia- obliga a pensar en ambas cosas a la vez, no siendo, generalmente, posible determinar la secuencia independientemente del estado, ni viceversa. Por otra parte, una vez establecidos ambos y explicitada la *función óptima* f que calcula el mínimo (o máximo) buscado, deben asignársele a esta función los valores correspondientes a los *estados terminales*. En el caso del camino de mínimo costo esto resultaba muy sencillo, pues si u era un nodo terminal, se tenía trivialmente que $f(u) = 0$. En la mayor parte de los problemas esto no es así y la determinación de los valores *de borde* o *extremos*, representa también un importante escollo en el camino hacia la solución.

Para mostrar la naturaleza de estas dificultades y el modo en que pueden resolverse, exponemos a continuación algunos ejemplos clásicos, enunciándolos primero verbalmente y mostrando luego cómo pueden ser planteados matemáticamente.

2.3.1. Asignación de un Recurso

Supongamos que contamos con una cierta cantidad (limitada) de un determinado *recurso*, sea este dinero, máquinas, agua, combustible o materia prima de cualquier tipo. Este recurso puede ser utilizado para diferentes *actividades*, cada una de las cuales produce un determinado retorno, que depende tanto de la actividad como de la cantidad del recurso invertida en ella. Para fijar ideas, supongamos que el recurso en cuestión es *dinero*. Supondremos, además, que: (a) Los retornos de las diferentes actividades pueden ser medidos en una unidad común, que podemos pensar que es la unidad *pesos*, lo que nos permite cambiar la palabra retorno por *ganancia*; (b) La ganancia de una actividad es independiente de la cantidad del recurso que se haya invertido en las otras; (c) La ganancia total se calcula sumando las ganancias proporcionadas por todas la actividades.

El problema consiste en hallar los montos que deben invertirse en cada actividad de manera que la ganancia total sea máxima.

El planteo matemático

Sea P la cantidad de dinero con la que contamos. Numeremos $1, 2, ..., N$ a las diferentes actividades, cada una de las cuales tiene asignada una *función de ganancia*: $g_1, g_2, ..., g_N$. Si x_i es la cantidad de pesos que se asigna a la actividad i, $g_i(x_i)$ será la ganancia proporcionada por esta actividad. Razonablemente, podemos suponer que las funciones g_i son crecientes. Nuestro problema consiste, entonces, en minimizar la función de ganancia total:

$$G(x_1, x_2, ..., x_N) = g_1(x_1) + g_2(x_2) + ... + g_N(x_N)$$

Sujetos a las condiciones:

$$x_1 + x_2 + ... + x_N = P$$
$$x_i \geq 0 \qquad i = 1, 2, ..., N$$

La formulación con Programación Dinámica

En este caso, si bien el tiempo no figura entre los ingredientes del problema, resulta más o menos sencillo encararlo de manera *secuencial*, pensando que primero se asigna una cierta cantidad x_1 a la actividad 1, luego una cantidad x_2 a la actividad 2, y así sucesivamente.

Ahora bien, al comenzar la asignación, es decir, en el momento de decidir el valor x_1, estamos limitados por las restricciones $0 \leq x_1 \leq P$. Una vez fijado x_1 el monto total con el que contamos habrá disminuido a $P - x_1$, por lo que las restricciones para la determinación de x_2 serán $0 \leq x_2 \leq P - x_1$. Siguiendo este razonamiento,

cuando se hayan determinado los valores $x_1, x_2, ..., x_k$, las restricciones de la asignación correspondiente a la actividad $k+1$ serán: $0 \leq x_{k+1} \leq P - (x_1 + x_2 + ... + x_k)$.

El Principio de Optimalidad nos dice que el valor x_{k+1} de una asignación óptima para las N actividades con un monto inicial P corresponde, a su vez, a una asignación óptima de las actividades $k+1, ..., N$ con un monto inicial $z = P - (x_1 + ... + x_k)$. La información esencial con la que debemos contar a cada paso es, entonces, el número de la actividad sobre la cual estamos decidiendo y la cantidad de pesos que restan distribuir. Por lo tanto, un *estado* deberá ser un par (k, z), con k el número de la actividad que debemos asignar y z el dinero disponible. Llamando f a la función óptima, la Ecuación Funcional del problema es:

$$f(k, z) = \max_{0 \leq x_k \leq z} g_k(x_k) + f(k+1, z - x_k)$$

Observemos que los montos de dinero siempre pueden considerarse enteros, pues en el caso de que sea posible asignar una fracción de peso a alguna actividad, puede cambiarse la unidad de medida de pesos a centavos. Esto hace que el número de estados posibles sea finito, pues: $k = 1, 2, ..., N$ y $z = 0, 1, 2, ..., P$, siendo N y P números enteros dados.

Lo único que resta, entonces, para establecer la recurrencia que resuelva el problema es determinar los valores de f para los estados terminales. Hecho esto, la Ecuación Funcional nos permitirá hallar f en todos los otros estados. En este caso, los estados terminales son los de la forma (N, z) con $z = 1, 2, ..., P$, es decir, aquellos que corresponden a la situación que sólo quedan z pesos para asignar a la actividad N. Puesto que las funciones g_i son crecientes, resulta natural que la totalidad del dinero sea asignado a esta última actividad. Esto es:

$$f(N, z) = g_N(z)$$

A partir de aquí, para $k = N-1, N-2, ..., 1$ y para $z = 1, 2, ..., P$ utilizamos la Ecuación Funcional y calculamos:

$$f(k, z) = \min_{0 \leq x_k \leq z} g_k(x_k) + f(k+1, z - x_k)$$

$$p(k, z) = \arg \min_{0 \leq x_k \leq z} g_k(x_k) + f(k+1, z - x_k)$$

Donde p es la *política óptima*, que nos da las asignaciones que maximizan la ganancia:

$$x_1 = p(1, P)$$

$$x_2 = p(2, P - x_1)$$

...

$$x_N = p(N, P - (x_1 + ... + x_{N-1}))$$

Complejidad

Contemos el número de operaciones que hacen falta para completar la recursión. En primer lugar el cálculo de f para los estados terminales requiere de $P + 1$ operaciones pues la única variable es z que toma valores enteros entre 0 y P. Luego, para cada (k, z) fijo se realizan z comparaciones, y este cálculo debe hacerse para $0 \leq z \leq P$, lo que da un total de $\frac{P(P-1)}{2}$ operaciones. Finalmente, k se mueve entre 1 y $N-1$, por lo que, la complejidad general del proceso será de $\mathcal{O}(NP^2)$. Esto es lo que se llama un algoritmo *pseudo-polinomial*: el número de operaciones es un polinomio en la cantidad de variables (N) y el *valor* de un de sus parámetros (P).

2.3.2. Multiplicación de Matrices

Se tienen N matrices $A_1, A_2, ..., A_N$ y $N + 1$ números naturales $r_0, ..., r_N$ tales que $A_i \in \mathbb{R}^{r_{i-1} \times r_i}$. Se desea calcular el producto:

$$A_1 \cdot A_2 \cdot ... \cdot A_N$$

Observamos que el número de operaciones necesarias para calcular $(A_1A_2)A_3$ no es el mismo que el número de operación que demanda $A_1(A_2A_3)$. El problema consiste, entonces, en decidir cómo deberían ubicarse los paréntesis para el obtener el producto total en un mínimo número de operaciones.

El planteo con Programación Dinámica.

Lo que buscamos es, en primer lugar, un k tal que la asociación

$$(A_1A_2...A_k)(A_{k+1}A_{k+2}...A_n)$$

resulte óptima. Y luego de esto, queremos encontrar un k' y un k'' que optimicen

$$(A_1A_2...A_{k'})(A_{k'+1}A_{k'+2}...A_k)(A_{k+1}A_{k+2}...A_{k''})(A_{k''+1}A_{k''+2}...A_n)$$

y así sucesivamente.

Según este planteo, a cada paso de nuestro problema nos concentramos en un subconjunto $\{A_j\}_{j=i}^{k} \subseteq \{A_j\}_{j=1}^{n}$, teniendo como datos las dimensiones dadas por los números $r_0, ..., r_n \in \mathbb{N}$. Entonces, la información esencial que necesitamos para encarar cada uno de los *subproblemas* está dada por el número de la primera y de la última

matriz del subconjunto. Luego, un estado será un par (i,k) y corresponderá a la situación en que debamos asociar el producto $A_iA_{i+1}...A_k$. En este caso, lo política óptima será una función $j = j(i,k)$, $i \leq j \leq k$, que al par (i,k) le asigna la ubicación óptima del paréntesis: $(A_iA_{i+1}...A_j)(A_{j+1}...A_k)$. Así, si definimos:

$f(i,k) =$ mínimo número de operaciones para obtener $A_iA_{i+1}A_k \qquad (1 \leq i \leq k \leq n)$

Por el Principio de Optimalidad tenemos que:

$$f(i,k) = f(i,j) + f(j+1,k) + r_{i-1}r_jr_k$$

En donde $r_{i-1}r_jr_k$ es el número de operaciones del producto $(A_iA_{i+1}...A_j)(A_{j+1}A_{j+2}...A_k)$. De aquí obtenemos la Ecuación Funcional:

$$f(i,k) = \min_{i \leq j \leq k} f(i,j) + f(j+1,k) + r_{i-1}r_jr_k \qquad (1 \leq i \leq j \leq j \leq n)$$

Naturalmente, la función $j(i,k)$ que asigna a cada par (i,k) el argumento j que minimiza la ecuación, nos brinda la solución del problema:

$$j_{11} = j(1,n)$$

$$j_{21} = j(1,j_{11}) \qquad j_{22} = j(j_{11},n)$$

$$j_{31} = j(1,j21) \qquad j_{32} = j(j_{21},j_{11}) \qquad j_{33} = j(j_{11},j_{22}) \qquad j_{34} = j(j_{22},n)$$

Y así siguiendo.

Dejamos el cálculo de la complejidad en manos del lector.

2.3.3. El Problema de la Carga

Una fábrica que produce N artículos debe cargar un contenedor con algunos de sus productos, pudiendo poner en él diferentes cantidades de cada uno. Cada unidad del producto i tiene un peso $p_i \in \mathbb{Z}$ y un valor $v_i \in \mathbb{Z}$. El peso total de la carga no puede superar el límite $P \in \mathbb{Z}$. El problema consiste en hallar las cantidades que deben cargarse de cada producto de manera que el valor total de la carga sea máximo, y su peso no supere el límite. Para la resolución supondremos que la cantidad disponible de cada artículo es ilimitada.

Planteo con Programación Dinámica

Si la variable x_i designa la cantidad de unidades del artículo i que se pondrá en el contenedor, nuestro problema puede plantearse como uno de programación lineal entera:

$$\text{máx} \sum_{i=1}^{N} v_ix_i$$

Sujeto a las restricciones:

$$\sum_{i=1}^{N} p_i x_i \leq P$$
$$x_i \geq 0 \qquad x_i \in \mathbb{Z}$$

Para plantear este problema por programación dinámica, supondremos que la carga se hace artículo por artículo. Análogamente a lo que ocurría en el caso de la Asignación de un Recurso, los estados son binomios (i, z) con $(1 \leq i \leq N)$ y $0 \leq z \leq P$ el peso restante a distribuir. Los cambios de estado están dados por los arcos $(i, z) \rightarrow (i+1, z-p_i x_i)$ donde x_i es la cantidad que se decide cargar del artículo i. Por el Principio de Optimalidad tenemos la Ecuación Funcional:

$$f(i, z) = \max_{0 \leq x \leq \lfloor \frac{z}{p_i} \rfloor} v_i x + f(i+1, z - p_i x) \qquad (1 \leq i \leq N)(0 \leq z \leq P)$$

También de manera análoga al caso de la Asignación de un Recurso definimos la función f en los estados terminales poniendo en el contenedor tanta cantidad del artículo N como sea posible:

$$f(N, z) = v_N \lfloor \frac{z}{p_N} \rfloor$$

Con esto estamos en condiciones de establecer la recursión que resuelve el problema. Para $i = N-1, N-2, ..., 1$ y $0 \leq z \leq P$ calculamos:

$$f(i, z) = \max_{0 \leq x \leq \lfloor \frac{z}{p_i} \rfloor} v_i x + f(i+1, z - p_i x)$$

$$x(i, z) = \arg \max_{0 \leq x \leq \lfloor \frac{z}{p_i} \rfloor} v_i x + f(i+1, z - p_i x)$$

La función x es la política óptima que nos permite reconstruir la solución:

$$x_1 = x(1, P)$$
$$x_2 = x(2, P - x_1 p_1)$$
$$...$$
$$x_i = x(i, P - (x_1 p_1 + ... + x_{i-1} p_{i-1}))$$

Complejidad

El cálculo de la complejidad para este problema es prácticamente idéntico al caso de Asignación de un Recurso, obteniéndose el mismo resultado. El número de operaciones es de $\mathcal{O}(NP^2)$. Veremos que el hecho de que el algoritmo sea *pseudo-polinomial* compromete la resolución numérica del problema en el caso en que se agreguen algunas complicaciones.

2.4. El Problema de la Dimensión.

Si añadimos a la restricción de peso una restricción de volumen, es decir, si introducimos valores w_i que representan el volumen de una unidad del artículo i, y un valor W para el volumen del contenedor, nuestro problema quedaría formulado de este modo:

$$\text{máx} \sum_{i=1}^{N} v_i x_i$$

Sujeto a:

$$\sum_{i=1}^{N} p_i x_i \leq P$$

$$\sum_{i=1}^{N} w_i x_i \leq W$$

$$x_i \geq 0 \qquad x_i \in \mathbb{Z}$$

Razonando como antes, un estado será ahora una terna (i, z, w) con i el artículo considerado, z el peso que aún es posible cargar y w el volumen restante. La Ecuación Funcional queda, entonces:

$$f(i, z, y) = \max_{0 \leq x \leq \text{mín}\{\lfloor \frac{z}{p_i} \rfloor, \lfloor \frac{y}{w_i} \rfloor\}} v_i x + f(i+1, z - p_i x, y - w_i x)$$

Los valores de f en un estado terminal serán:

$$f(N, z, w) = v_N \cdot \text{mín}\{\lfloor \frac{z}{p_N} \rfloor, \lfloor \frac{w}{w_N} \rfloor\}$$

Es posible, entonces, establecer la recursión para resolver el problema. Hay, sin embargo, un inconveniente. El valor W juega en este caso un papel idéntico al de P. Resulta, por lo tanto, bastante sencillo comprobar que la complejidad de un algoritmo construido en base a la recursión dada por Programación Dinámica es de $\mathcal{O}(NP^2W^2)$. Esto hace que el tiempo de ejecución aumente bastante si los valores de P y W son grandes. Pero no es sólo este el problema.

En el caso en que hay solamente una restricción de peso, en cada paso del algoritmo se calculan los valores de f para los estados (i, z) con i fijo y $0 \leq z \leq P$. Esto hace un total de $P+1$ datos por paso. Una vez completado el proceso, la cantidad de estados en que hemos calculado f es $N(P+1)$. Agregando la restricción de volumen, cada paso del algoritmo calcula f en *todos* los estados (i, z, w) con i fijo y $0 \leq z \leq P$, $0 \leq w \leq W$. Así, al finalizar la recurrencia, deberán guardarse en memoria $N(P+1)(W+1)$ datos,

cosa que puede resultar imposible para valores no muy grandes de P y W, incluso con computadoras modernas.

Esto muestra una de las limitaciones más grandes de la Programación Dinámica: el problema de la *dimensión*. Si el número de restricciones es grande, la complejidad y, sobre todo, el espacio en memoria, crecen demasiado. Sin embargo, puede adaptarse el método para casos con este, en que la dimensión, si bien impide el uso directo de la Programación Dinámica, no es demasiado grande.

2.5. Multiplicadores de Lagrange

El método de los Multiplicadores de Lagrange[2] es un recurso muy útil para resolver problemas de minimización o maximización de funciones de varias variables sujetas a restricciones. Describiremos brevemente su uso en el Cálculo antes de mostrar la manera de utilizarlos en Programación Dinámica.

Supongamos que queremos hallar los extremos de una función diferenciable de dos variables $F(x, y)$ sobre todos los (x, y) en una curva descripta por la ecuación $G(x, y) = 0$. De no existir esta restricción procederíamos a calcular las derivadas parciales de F respecto de x y de y, igualándolas a 0. Pero este procedimiento deja de ser útil al imponer la restricción, pues no es necesario que las derivadas parciales se anulen para que existan extremos *sobre los puntos de la curva*. Para resolver este problema, definimos una función H:

$$H(x, y) = F(x, y) + \lambda G(x, y)$$

Donde λ es un Multiplicador de Lagrange. Si ahora buscamos los extremos de H de la manera usual tendremos:

$$\frac{\partial H}{\partial x} = \frac{\partial F}{\partial x} + \lambda \frac{\partial G}{\partial x} = 0$$

$$\frac{\partial H}{\partial y} = \frac{\partial F}{\partial y} + \lambda \frac{\partial G}{\partial y} = 0$$

A partir de estas ecuaciones podemos obtener x e y en términos de λ y luego usar la restricción $G(x, y) = 0$ para determinar el valor de λ. Por supuesto, no siempre es posible obtener una solución a través de este método, puesto que los despejes de $x = x(\lambda)$, $y = y(\lambda)$ y luego el cálculo del valor de λ pueden resultar impracticables.

[2]Expuesto por Joseph Louis Lagrange (1736-1813) en su obra *Mécanique Analytique*

2.5.1. Los Multiplicadores de Lagrange en Programación Dinámica

Tomemos un problema general de optimización definido del siguiente modo: sean g y h dos funciones definidas sobre un conjunto finito $\mathbb{S}$ y W un número natural. Queremos hallar:

$$\text{mín}\, g(x)$$

Sujetos a las restricciones:

$$h(x) \leq W$$

$$x \in \mathbb{S}$$

Observemos que el Problema de la Carga con restricciones de peso y volumen puede expresarse de este modo, siendo:

$$x = (x_1, ..., x_N)$$

$$g(x) = \sum_{i=1}^{N} v_i x_i$$

$$h(x) = \sum_{i=1}^{n} w_i x_i$$

$$\mathbb{S} = \{x : \sum_{i=1}^{N} p_i x_i \leq P\}$$

Para utilizar los multiplicadores de Lagrange pensaremos que la restricción:

$$h(x) \leq W$$

juega el papel que cumplía, en el caso de la minimización o maximización de una función diferenciable, la restricción $G(x, y)$=0. Plantearemos entonces, el siguiente problema auxiliar:

$$\text{mín}\, g(x) + \lambda h(x)$$

Sujeto a:

$$x \in \mathbb{S}$$

Hemos reducido el número de restricciones, de dos que teníamos originalmente, a una. Veamos que es posible hallar un λ tal que la solución del segundo problema sea también solución del primero:

Teorema 1 *Si para todo $\lambda > 0$, la función $g(x) + \lambda h(x)$ tiene un mínimo global en $\mathbb{S}$ que se alcanza en un elemento $x_\lambda \in \mathbb{S}$. Entonces:*

1. *x_λ es solución de:*

$$\begin{aligned} &\text{mín}\, g(x) \\ &h(x) \leq h(x_\lambda) \\ &x \in \mathbb{S} \end{aligned}$$

2. *$h(x_\lambda)$ decrece cuando λ crece*

Demostración:

1) Sea $x \in \mathbb{S}$. Por hipótesis tenemos que:

$$g(x_\lambda) + \lambda h(x_\lambda) \leq g(x) + \lambda h(x)$$

$$g(x_\lambda) - g(x) \leq \lambda(h(x) - h(x_\lambda))$$

Luego, como $h(x) \leq h(x_\lambda)$ vale $\lambda(h(x) - h(x_\lambda)) \leq 0$. Entonces:

$$g(x_\lambda) - g(x) \leq 0 \Longrightarrow g(x_\lambda) \leq g(x)$$

2) Sea $0 < \lambda < \mu$ por hipótesis vale que:

$$g(x_\lambda) + \lambda h(x_\lambda) \leq g(x_\mu) + \lambda h(x_\mu)$$

$$g(x_\mu) + \mu h(x_\mu) \leq g(x_\lambda) + \mu h(x_\lambda)$$

Sumando ambos términos y reordenando queda

$$(\mu - \lambda)(h(x_\lambda) + h(x_\mu)) \geq 0$$

Y como $\mu > \lambda$, $h(x_\lambda) \geq h(x_\mu)$. ■

La primera parte de este teorema muestra que la resolución del problema auxiliar brinda también la solución de un tercer problema que tiene la misma estructura que el original, que deseamos resolver. La única diferencia está en que, en lugar de la restricción $h(x) \leq W$, tenemos otra, $h(x) \leq h(x_\lambda)$. Buscamos, entonces, un valor de λ tal que $h(x_\lambda) = W$. Para esto, utilizamos la segunda parte del teorema, que nos dice que $h(x_\lambda)$ es una función decreciente del multiplicador de Lagrange λ. Esto nos permite hallar el valor de λ a través de un algoritmo de bisección:

Tomamos un valor inicial λ_0 y calculamos $h(\lambda_0)$. Si $h(\lambda_0) < W$ tomamos $\lambda_1 = \frac{\lambda_0}{2}$. Si, en cambio $h(\lambda_0) > W$, definimos un λ_1 mayor que λ_0, por ejemplo: $\lambda_1 = \frac{3}{2}\lambda_0$, y reiteramos el procedimiento para λ_1.

Por supuesto, hemos resulto el inconveniente del espacio necesario para alojar la información en memoria, reduciendo el problema a uno con una sola restricción. Sin embargo, es necesario correr el algoritmo varias veces, con diferentes valores de λ, para encontrar finalmente la solución de problema original.

3. Aplicación al Cálculo de Variaciones

Hasta aquí hemos trabajado con problemas en los que había que minimizar o maximizar una función de varias variables, sujeta a ciertas restricciones. En el Cálculo de Variaciones consideramos problemas similares, pero que involucran funciones de infinitas variables, es decir, funciones de funciones, que reciben el nombre de *funcionales*.

El problema más antiguo del Cálculo de Variaciones consistía en hallar la figura de área máxima entre todas las que tienen perímetros iguales[3]. Varios matemáticos griegos de la antigüedad trabajaron en este problema pero fue Jacob Bernoulli, el mayor, (1654-1705) quién dio la solución (el círculo), demostrándo su validez. Uno de los hermanos de Jacob, Johannes Bernoulli (1667 - 1748) lanzó, como desafío a los matemáticos del mundo, uno de los problemas que más importancia histórica tuvieron en el desarrollo de esta teoría: el problema de la *Braquistocrona*. Dados dos puntos en el espacio, la braquistocrona es la curva que los une, a lo largo de la cuál el descenso de una partícula por acción de la gravedad se realiza en menor tiempo. Este problema había inquietado ya a Galileo, que no había podido resolverlo, hecho este bastante natural pues hacía falta contar con el Cálculo como herramienta matemática para encontrar la solución. Aparentemente, el desafío de Johannes Bernoulli iba dirigido particularmente a Newton. Un año después de planteado el problema aparecieron varias soluciones, entre ellas una de Leibniz, una de Jacob Bernoulli, y una anónima. Según se cuenta, al ver la breve resolución anónima, Johannes habría exclamado algo así como *Reconozco al león por su garra*. En efecto, el autor de esta solución había sido Newton. Por lo demás, el resultado del problema es sorprendente: la braquistocrona es una cicloide[4].

Los funcionales correspondientes a estos ejemplos se expresan en términos de una

[3]Este tipo de restricción es la que, en términos modernos, corresponde a un problema de *isoperímetros*.

[4]La curva que describe un punto fijo en una rueda cuando esta se desplaza.

determinada integral que involucra a la función buscada. Abordaremos aquí un problema general, con un funcional de la forma:

$$J(f) = \int_a^b G(x, f(x), f'(x))dx$$

Buscaremos una función f que haga mínimo este funcional, imponiendo como única restricción un valor inicial $f(a) = c$.

3.1. El Planteo Formal con Programación Dinámica

Definamos:

$$\bar{J}(a, c) = \min_{f:f(a)=c} J(f) = \min_{f:f(a)=c} \int_a^b G(x, f(x), f'(x))dx$$

Por la aditividad de la integral tenemos que:

$$\bar{J} = \min_{f:f(a)=c} (\int_a^{a+h} G(x, f, f')dx + \int_{a+h}^b G(x, f, f')dx)$$

Y aplicando el Principio de Optimalidad:

$$\bar{J} = \min_f (\int_a^{a+h} G(x, f, f')dx + \bar{J}(a + h, c(f))$$

con $c(f) = f(a + h)$. Llegado el momento haremos tender h a cero, pero primero debemos escribir las cosas de manera apropiada. En primer lugar:

$$\int_a^{a+h} G(x, f, f')dx = G(a, c, f'(a))h + o(h)$$

$$c(f) = f(a + h) = c + f'(a)h + o(h^2)$$

Por lo cual:

$$\bar{J}(a, c) = \min_v (G(a, c, v)h + \bar{J}(a + h, c + vh)) + o(h)$$

donde v juega el papel de $f'(a)$. Ahora, reescribiendo los términos apropiadamente y tomado límite con h tendiendo a 0, tenemos la llamada *Ecuación de Bellman*:

$$-\frac{\partial \bar{J}}{\partial a} = \min_v (G(a, c, v) + v\frac{\partial f}{\partial c})$$

Sólo en algunos (raros) casos es posible encontrar una solución explícita a un problema variacional. Normalmente no puede esperarse más que una descripción más o

menos satisfactoria de la solución a través de una ecuación en derivadas parciales. Euler, Lagrange, Weierstrass, Hamilton, Jacobi, Hilbert y otros importantes matemáticos abordaron estos problemas y establecieron diversas condiciones necesarias y suficientes para diferentes tipos de problemas, todas ellas a través de ecuaciones en derivadas parciales que involucran a la función incógnita. Por falta de espacio no abordaremos este tema en detalle. Sin embargo, llamaremos la atención sobre un punto importante: muchas de estas condiciones clásicas, que fueron halladas, en su momento, como consecuencia de un arduo trabajo, pueden ser deducidas fácilmente a partir de la Ecuación de Bellman.

3.2. Resolución Numérica de Problemas Variacionales

Dada la enorme dificultad que implica hallar una solución explícita de un problema variacional, luego de la aparición de las computadoras se ha intentado encontrar métodos que permitieran obtener aproximaciones numéricas de las soluciones. En esto también ha hecho un importante aporte la Programación Dinámica.

Consideremos, a modo el ejemplo, el problema de la Braquistocrona, que comentamos anteriormente. Tomemos dos puntos en el plano, $(0,0)$ y (a,b), $0 < a\ 0 > b$. Nuestra incógnita será una función $y = y(x)$, que para cada punto x dará la altura correspondiente, de manera que la curva dada por $(x, y(x))$ será la que describa la trayectoria de la partícula. El funcional que debemos minimizar es el que expresa el tiempo de la caída dada la trayectoria y, es decir:

$$J(y) = \int_a^0 (\frac{1+y'^2}{2gy})^{\frac{1}{2}} dx$$

Donde g es la gravedad y la expresión $(1+y'^2)^{\frac{1}{2}}dx$ es el diferencial de arco. Definimos:

$f(x,y) =$ tiempo de caída desde (x,y) hasta (a,b) siguiendo una trayectoria óptima.

Entonces, aplicando el Principio de Optimalidad tenemos que:

$$f(x,y) = \min_{y'} \int_x^{x+\delta} (\frac{1+y'(s)^2}{2gy(s)})^{\frac{1}{2}} ds + f(x+\delta, y+y'\delta)$$

Lo cual puede aproximarse de la siguiente manera:

$$f(x,y) = \min_{y'} [(\frac{1+y'^2}{2gy})^{\frac{1}{2}} + f(x+\delta, y+y'\delta)]$$

A diferencia de lo que sucedía en el planteo formal (continuo), en este caso el valor δ del incremento en la variable x es un número pequeño *fijo*. Para discretizar el

problema, tomamos un valor ε para el incremento en la variable y y construimos a partir de ellos una grilla como la que muestra la Fig.4. Nuestra solución discreta será una sucesión de puntos de la grilla, empezando por el $(0,0)$ y terminando en el (a,b) que, unidos, formarán una poligonal que aproxime a la solución analítica.

Dadas las características del problema resulta natural suponer que la trayectoria óptima estará dada por una función *decreciente*, y que no tomará puntos fuera del rectángulo de la grilla. Hechas estas suposiciones, el problema numérico se resuelve del siguiente modo:

Sean $x_i = i\delta$, $i = 0, 1, ..., n = \frac{a}{\delta}$ las coordenadas x de los puntos de la grilla. Del mismo modo definimos $y_j = j\varepsilon$, $j = 0, 1, ..., m = \frac{b}{\varepsilon}$. Lo que buscamos no es otra cosa que un camino de *costo* mínimo entre los vértices $(x_0, y_0) = (0,0)$, $(x_n, y_m) = (a,b)$ del grafo formado por los nodos de la grilla, donde las ramas unen cada punto (x_i, y_j) con todos los puntos $(x_{i+1}, y_{j'})$, $j' \leq j$.

Figura 4: Grilla

La función f, que ya hemos definido, es la *función óptima* del planteo de Programación Dinámica, que nos permitirá establecer la recurrencia que resuelva el problema. Para ello, debemos definir el valor de f en el nodo terminal y determinar la manera de obtener la *política óptima*, que será la trayectoria. Lo primero es sencillo, pues, naturalmente, $f(a,b) = 0$. Lo segundo requiere un pequeño esfuerzo.

Dado un nodo (x_i, y_j) en la grilla, y teniendo el valor $f(x_i, y_j)$ queremos obtener el nodo (x_{i+1}, y_k) que le sigue en una trayectoria óptima. En los ejemplos estudiados hasta aquí, esto venía dado por el argumento que minimizaba f. En este caso, dicho argumento será un valor y', correspondiente a la derivada de la trayectoria y en el punto (x_i, y_j). Basta tomar, entonces, la aproximación de la derivada

$$y' = \frac{y_k - y_j}{x_{i+1} - x_i} = (k - j)\frac{\varepsilon}{\delta}$$

para poder conseguir el k que buscamos. Así, la política óptima viene dada por:

$$k(i,j) = \arg\min_{y'} f(x_i, y_j)\frac{\delta}{\varepsilon} + j$$

4. Programación Dinámica Estocástica

En las dos secciones anteriores, se analizaron problemas en los cuales dado un estado y una política, tanto el *payoff* (costo o ganancia según el caso) y el estado siguiente eran conocidos. Se podría decir entonces que se trataba de programación dinámica *determinista*. Si, en cambio, no se sabe con seguridad cuales serán estos, y en cambio tenemos alguna función de distribución de probabilidad se habla de programación dinámica *estocástica*. Las ideas básicas de determinar los estados, las etapas, las políticas y las ecuaciones funiconales siguen valiendo, simplemente toman una forma distinta. La aleatoriedad que aparece introduce naturalmente a un *Proceso Estocástico*. Matemáticamente, un proceso estocástico se define como un conjunto de variables aleatorias X_t cuya distribución varía de acuerdo a un parámetro, generalmente el tiempo. El proceso estocástico más sencillo es el *Proceso de Markov*, aquí una pequeña introducción y los problemas de decisión asociados a estos procesos.

4.1. Procesos de Decisión Markoviana

Procesos de Markov

Consideremos un sistema que en cualquier tiempo determinado está en uno de un número finito de estados $i = 1, 2, ..., N$, y asumamos que en tiempos discretos $t = 0, 1, ...$ el sistema cambia aleatoriamente de un estado a otro. Las transiciones de un estado a otro se rigen por una *matriz de transición* $P = (p_{ij})$ donde $p_{ij} =$ es la probabilidad de que el sistema esté en el estado j al tiempo $t + 1$ dado que estaba en el estado i en el tiempo $t \qquad (i, j = 1, 2, ...N)$

Consideramos aquí el caso en el que la matriz P no depende del tiempo, que es el caso más interesante de estudiar. Sean las siguientes funciones:

$x_t(i) =$ la probabilidad de que el sistema esté en el estado i en el tiempo t

Con $i = 1, 2, ..., N$ y $t = 0, 1, ...$ Un sencillo análisis de probabilidades nos dá:

$$x_{t+1}(j) = \sum_{i=1}^{N} p_{ij} x_t(i) \qquad x_0(i) = c_i \qquad j = 1, ..., N.$$

La teoría de los procesos de Markov se dedica a estudiar el comportamiento asintótico de las funciones $x_t(i)$ cuando $t \to \infty$. Si todas las p_{ij} son positivas, no es difícil probar que estas funciones convergen a unas xi que satisfacen la ecuación:

$$x(j) = \sum_{i=1}^{N} p_{ij} x(i) \qquad j = 1, ..., N.$$

Lo sorprendente es que los valores en el límite son independientes a los valores iniciales $x_0(i)$.

Problemas de Decisión

Extendamos ahora el concepto de un proceso a situaciones más generales en las cuales se toman decisiones en cada etapa. Supongamos que en cada tiempo la matriz de transición puede ser elegida entre un conjunto de tales matrices, y notemos a la matriz correspondiente a una política y como $P(y) = (p_{ij}(y))$. Supongamos también que no sólo hay un cambio de estado en cada etapa, pero también un retorno, que es una función que depende del estado inicial, del final y de la decisión. Sea $R(y) = (r_{ij}(y))$ la matriz de retornos. Un proceso de este tipo se llama un *Proceso de Decisión Markoviana*. El problema es elegir una secuencia de decisiones que maximicen el retorno esperado que se obtiene de un proceso de N etapas, dado el estado inicial.

Formulación Matemática

Usemos técnicas de programación dinámica para obtener una formulación analítica para el problema de decisión markoviana. Para $i = 1, 2, ...N$ y $n = 0, 1, ...$ sea $f(i, n)$ =retorno esperado de un proceso de n etapas, empezando en el estado i usando una política óptima. Notemos que n representa la *longitud* del proceso, a diferencia de t, que era usada antes para denotar el tiempo. El principio de optimalidad da como consecuencia la ecuación funcional:

$$f(i,n) = \max_y \sum_{j=1}^{N} p_{ij}(r_{ij}(y) + f(j, n-1)) \qquad (i = 1, 2, ..., N) \qquad (n = 1, 2, ...)$$

$$f(i, 0) = 0 \qquad (i = 1, 2, ..., N)$$

Una política óptima consiste de un vector $(y_n(1), y_n(2), ..., y_n(N))$

Veamos algunos simples ejemplos en los cuales se introduce la aleatoridad y cómo se resuelven mediante la programación dinámica.[5]

4.2. Ejemplos de Retorno Incierto

4.2.1. Distribución de un Producto

Consideremos una cadena de supermercados que ha adquirido 6 litros de leche de un distribuidor local. La cadena debe distribuir los 6 litros entre sus 3 mercados. Si un mercado vende un litro de leche la cadena recibe una ganancia de $2. Por cada

[5]Los ejemplos están basados en [7]

litro de leche sin vender la cadena sólo gana \$0.50. Desafortunadamente, la demanda de leche es incierta y está dada por la siguiente tabla:

Mercado	Cant. de Litros	Probabilidad
1	1	0.6
	2	0
	3	0.4
2	1	0.5
	2	0.1
	3	0.4
3	1	0.4
	2	0.3
	3	0.3

El objetivo de la cadena es maximizar la ganancia (su esperanza) a partir de los 6 litros de leche.

Notemos que es un problema similar a los que teníamos anteriormente, la diferencia es que no conocemos la ganancia con certeza. Podemos, sin embargo, determinar la ganancia esperada para cada entrega de leche en un mercado. Por ejemplo, la ganancia esperada de entregar 2 litros en el marcado 1 es 0.6(2.5)+0.4(4)=3.1 ya que la probabilidad de que demanden 1 litro es 0,6 y en este caso se ganarán \$2 por ese litro vendido más los \$0.50 del que quedó sin vender. La que demanden 2 litros es 0. Y la que demanden 3 es \$0.4 y de este modo se ganan los \$4. Podemos hacer esta misma cuenta para todas las posibilidades (omitimos la posibilidad de entregar más de 3 litros en un mercado pues tiene probabilidad 0):

Mercado	Cant. de Litros	Ganancia esperada
1	1	2
	2	3.1
	3	4.2
2	1	2
	2	3.25
	3	4.35
3	1	2
	2	3.4
	3	4.35

Ahora tenemos un problema determinístico. Un estado en este caso será un par (i, k) que representa que al mercado i se le entregan k litros. Sea $c(i, k)$=ganancia esperada al entregar k litros al mercado i (con $c(i, k) = c(i, 3)$ si $4 \leq k \leq 6$). Si llamamos $f(i, k)$ a la ganancia esperada máxima llegamos a la ecuación funcional:

$$f(3,k) = c(3,k) \qquad (0 \leq k \leq 6)$$

$$f(i,k) = \max_{0 \leq y \leq k} c(i,y) + f(i+1, k-y) \qquad (i = 1,2) \qquad (0 \leq k \leq 6)$$

Y llamamos $y(i,k)$ al y que maximiza en cada caso. La política óptima será:

$$y_1 = y(1,6); y_2 = y(2, 6-y_1); y_3 = y(3, 6-y_1-y_2)$$

Haciendo las cuentas, en este ejemplo, y de la misma manera que uno hacía en el caso determinista, se llega a que la máxima ganancia esperada será de 9,75, asignando 1 litro al primer mercado, 3 al segundo y 2 al tercero.

4.2.2. Valuación de una Opción

Opciones

Un problema común en el ámbito de las finanzas es el de la valuación de una opción. Una opción *call europea* (la más simple de todas) es un contrato entre dos partes que cumple las siguientes condiciones: En un tiempo T predeterminado en el futuro llamado *tiempo de expiración*, el *portador* puede (de aquí la palabra opción) comprar al *escritor* un predeterminado *activo* S (por ejemplo, una acción), llamado *activo subyacente*, o simplemente *subyacente* por una cantidad predeterminada K, llamado el *precio de ejercicio*. Otro tipo de opción es la llamada *opción americana*, que a diferencia de la europea, uno puede ejercer la opción no solo en el tiempo de ejercicio, sinó en cualquier tiempo entre el inicio del contrato y éste. En teoría se piensa en el tiempo como un continuo, pero en la práctica se discretiza. Se llama el *payoff* de la opción, al retorno de la misma en T. En el caso de las call europea y americana es máx$\{0, S-K\}$. Pues si $S-K > 0$ entonces uno ejerce la opción pagando K y luego vende el activo a S lo que representa una ganancia de $S-K$. En el caso que $S-K \geq 0$ no conviene ejercer la opción, pues se perdería $K-S$ y la opción no tiene valor.[6]

Ejemplo

Supongamos que tenemos la opción de comprar una acción de una empresa a \$150. Podemos ejercer la opción en cualquier día en los próximos 10 días. Notemos que es una opción call americana. El precio actual de la acción es de \$140. Tenemos de antemano un modelo, muy precario por cierto, del movimiento de la acción que dice

[6] Ver [10] para más información sobre valuación de opciones

lo siguiente: Sube \$2 con probabilidad 0.4 baja \$2 con probabilidad 0.5 y se queda igual con probabilidad 0.1. El valor de la opción, si la ejercemos con precio S será $S - 150$. Notemos que solamente ejerceremos la opción si el precio de ejercicio supera los 150. Podemos formular este problema con programación dinámica de la siguiente manera: Para cada día tendremos un estado i. El estado será el precio de la acción ese día. Sea $f(i, S)$ la máxima ganancia esperada ejerciendo la opción en el día i dado que el valor de la acción es S.

Entonces, usando herramientas de programación dinámica, la decisión óptima está dada por:

$$f(i, S) = \text{máx}\{S - 150, 0{,}4f(i + 1, S + 2), 0{,}1f(i + 1, S), 0{,}5f(i + 1, (S - 2)\},$$

$$(1 \leq i \leq T - 1) \qquad (140 - 2S10 \leq S \leq 140 + 2S10)$$

$$f(10, S) = \text{máx}\{0, S - 150\} \qquad (140 - 2S10 \leq S \leq 140 + 2S10)$$

Referencias

[1] Bell, E.T; *Los Grandes Matemáticos.*

[2] Bellman, Richard y Dreyfus, Stuart ; *Applied Dynamic Programming*; Princeton University Press, New Jersey, 1962.

[3] Dreyfus, Stuart; *Dynamic Programming and the Calculos of Variations*; Academic Press, New York,1965.

[4] Koo, Delia; *Elements of Optimization (With Applications in Economics and Business)*; Heildelberg Science Library, New York, 1977.

[5] Neimhauser, George; *Integer and Combinatorial Optimization*; Wiley, New York, 1988.

[6] Sasieni, M., Yaspan, A., Friedman L.; *Investigación de Operaciones. Métodos y Problemas*; AID, México, 1967.

[7] Trick, Michael; *A Tutorial on Stochastic Dynamic Programming*, http://mat.gsia.cmu.edu/

[8] Ventsel, Elena; *Investigación de Operaciones. Problemas, principios, metodología*; Editorial Mir, Moscú, 1983.

[9] Vicentini, Fabio , *Notas de Optimización*

[10] Wilmott, P., Howiston S., Deweynne J.; *The Mathematics of Finantial Derivatives. A Student Introduction*; Cambridge University Press, 1995