Algorítmica

Capítulo 4: Programación Dinámica

Tema 10: Programación Dinámica. El Principio del Óptimo

- El enfoque de la P.D.
- Principio del Óptimo de Bellman
- Algoritmos basados en P.D.
- PD frente a otros enfoques

Programación Dinámica

Un método de diseño de algoritmos que se puede usar cuando la solución del problema se puede ver como el resultado de una sucesión de decisiones.

Es una técnica que principalmente se aplica en Problemas de Optimización

Hoy día vive un resurgimiento por sus aplicaciones en los problemas de reconocimiento de secuencias de ADN

Richard Bellman

El origen de la P.D.

- Una forma de resolver un problema es caracterizar su solución mediante las soluciones de sus subproblemas.
- Esta idea, cuando se emplea recursivamente, produce métodos eficientes de solución.
- La hemos aplicado anteriormente en el caso de los algoritmos Divide y Vencerás, Backtracking o Branch and Bound. Incluso en Greedy.
- Pero en muchos casos, dado un problema de tamaño n solo puede obtenerse una caracterización efectiva de su solución en términos de la solución de los subproblemas de tamaño n-1.
- En estos casos, la Programación Dinámica (PD) proporciona algoritmos eficientes.

El origen de la P.D.

- La PD es un técnica típica del campo de la Teoría de Control. Entre los métodos mas frecuentemente usados para el diseño de sistemas de control están,

  1.- El Cálculo de Variaciones,

  2.- El Principio de Máximo de Pontryaguin, y

  3.- La Programación Dinámica.

- Cada uno de estos tres métodos está ligado a alguna formulación conocida de la Mecánica Clásica: Los primeros, con la ecuación de Euler-Lagrange, los segundos con el Principio de Hamilton y los terceros con la teoría de Hamilton-Jacobi

- En particular, la PD se aplica a la resolución de **Problemas de Decisión Markovianos (n-etápicos)** usando **Ecuaciones de Recurrencia**

Problemas n-etápicos

- Supongamos que se dispone de una cantidad inicial de dinero x para invertir en un negocio de alquiler de ordenadores. Este dinero se usará para comprar ordenadores para tratamiento de textos (PC) y para conexiones a la red (CR).
- Supongamos que se gasta **y** en la compra de PC y, el resto, se dedica a la compra de CR.
- Lo que producen al año los PC es función de la cantidad inicial invertida, **y**, e igual a **g(y)** el primer año. Así mismo, lo que producen al año los CR es función del capital restante, **x-y**, e igual a **h(x-y)** el primer año.
- La política de la empresa determina que al final de cada año, todos los ordenadores usados de uno y otro tipo se cambien.

Problemas n-etápicos

- El valor del cambio de todos los PC es una función del total de dinero invertido en PC e igual a **ay** el 1er año, **0<a<1**.
- También, el valor del cambio de los CR es una función igual a **b(x-y), 0<b<1**.
- Hay que tomar una serie (sucesión) de decisiones optimas para que el beneficio total a lo largo de **N** años sea máximo.

$$f_1(x) = \text{Max}_{0 \le y \le x} \{g(y) + h(x-y)\}$$

$$f_N(x) = \text{Max}_{0 \le y \le x} \{g(y) + h(x-y) + f_{N-1}[ay + b(x-y)]\}$$

A partir de $f_1(x)$, obtenemos $f_2(x)$, luego $f_3(x)$,... . En cada etapa, se obtiene la decisión óptima a tomar al comienzo de la jª etapa.

El valor que maximiza las funciones entre corchetes en la ecuación es la decisión óptima a tomar al comienzo de la operación del año N.

Con un **método directo**, hay que resolver simultáneamente las N ecuaciones que se obtienen haciendo cero las derivadas parciales

El Principio del Óptimo

- Supongamos un problema de decisión 1-etápico sobre un sistema cualquiera (PC, grafo, robot, ...)

  El rendimiento (x^1) de un PC depende, entre otras variables de la temperatura (m)

- Sea x la variable que da las decisiones que vamos tomando, y supongamos que el sistema se encuentra inicialmente en un estado x^1.
- Si tomamos una decisión m_1 , y el estado del sistema pasa de x^1 a x^2, el cambio se habrá hecho por medio de una transformación como,

$$x^2 = g(x^1, m_1)$$

- Esto producirá una respuesta, o recompensa,

$$R_1 = r(x^1, m_1)$$

- Lo que queremos es elegir m_1 de manera que se maximice la respuesta, es decir, el valor de R_1

El Principio del Óptimo

- Esta claro que la solución a ese problema de decisión uni-etápico no tiene dificultad, porque el máximo de la respuesta esta dado por,

$$f_1(x^1) = \text{Max}_{\ m_1} \{r(x^1, m_1)\}$$

- La decisión que da el máximo valor de la salida, se llama una decisión óptima, o estrategia de control óptima.

• En un proceso de decisión bi-etápico, si el estado del sistema se transforma, primero, de x^1 a x^2 por medio de $x^2 = g(x^1, m_1)$, y después, de x^2 a x^3 por $x^3 = g(x^2, m_2)$, esta sucesión de operaciones produce una única salida, o recompensa total

$$R_2 = r(x^1, m_1) + r(x^2, m_2)$$

El Principio del Óptimo

- La máxima recompensa estará dada por,

$$f_2(x^1) = \text{Max}_{m_1, m_2} \{r(x^1, m_1) + r(x^2, m_2)\}$$

- La función de recompensa total ahora se maximiza sobre la política $\{m_1, m_2\}$.
- A la política que maximiza a R_2 se le llama política óptima.
- Como fácilmente puede verse, manejar un proceso de decisión bi-etápico es mas complicado que manejar el uni-etápico anterior.
- La dificultad y la complejidad aumentan conforme lo hace el número de etapas del problema en consideración.

El Principio del Óptimo

- Para un proceso N-etápico, el problema consiste en elegir aquella política N-etápica $\{m_1, m_2 \ldots, m_N\}$ tal que maximice la recompensa total,

$$R_N = \sum_j r(x^j, m_j) \ , \ \ j \in J = \{1,2,\ldots,N\}$$

- El sistema se transforma de x^1 a x^2 por $x^2 = g(x^1, m_1)$, de x^2 a x^3 por $x^3 = g(x^2, m_2)$,..., y desde x^{N-1} hasta x^N por $x^N = g(x^{N-1}, m_{N-1})$.

- La recompensa máxima de este proceso N-etápico es

$$f_N(x^1) = \text{Max}_{\{m_j\}} \{ \sum_j r(x^j, m_j) \}, \ j \in J$$

- La política $\{m_j\}$ que determina $f_N(x^1)$ es la política óptima, o estrategia de control óptima. En este caso, las m_j forman una política de control N-etápica.

El Principio del Óptimo

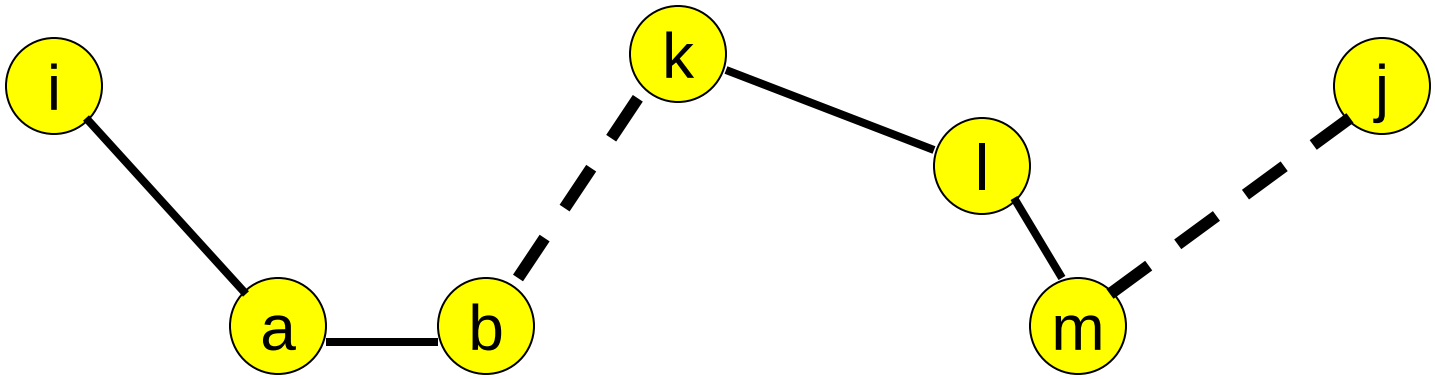
- Llevar a cabo el procedimiento de maximización por un método directo, requiere la resolución de las N ecuaciones simultaneas que se obtienen igualando a cero las derivadas parciales, con respecto a m_j, $j \in J$, de las cantidades entre los corchetes.
- Esto puede ser formidablemente complicado en tiempo.
- Para resolver un problema de decisión optimal con un gran número de etapas es necesario un procedimiento sistemático que mantenga el problema en niveles tratables.
- Tal procedimiento sistemático de resolución puede obtenerse haciendo uso del principio fundamental de la PD, **el Principio del Óptimo de Bellman**

El Principio del Óptimo

- **Enunciados:**
- Dado el estado actual del sistema, la política óptima (la sucesión de decisiones) que hay que definir para las etapas restantes es independiente de la política adoptada en las etapas precedentes.
- Una política es óptima si en un período dado, cualesquiera que sean las decisiones precedentes, las decisiones que quedan por tomar constituyen una política óptima independientemente de los resultados de las decisiones precedentes.
- Una política óptima solo puede estar formada por subpolíticas óptimas.

El Principio del Óptimo

Dado el estado actual del sistema, la política óptima (la sucesión de decisiones) que hay que definir para las etapas restantes es independiente de la política adoptada en las etapas precedentes.



El Principio del Óptimo

En el anterior ejemplo había que elegir aquella política N-etápica $\{m_1, m_2 \ldots, m_N\}$ tal que maximizara la recompensa total,

$$R_N = \sum_j r(x^j, m_j) \ , \ j \in J = \{1,2,\ldots,N\}$$

Ahora, por el **POB,** la recompensa total del mismo proceso de decisión puede escribirse como,

$$R_N = r(x^1, m_1) + f_{N-1}[g(x^1, m_1)]$$

siendo $r(x^1, m_1)$ la recompensa inicial y $f_{N-1}[g(x^1, m_1)]$ la recompensa máxima asociada a las restantes (últimas) N-1 etapas.

Así, la recompensa máxima estará dada por,

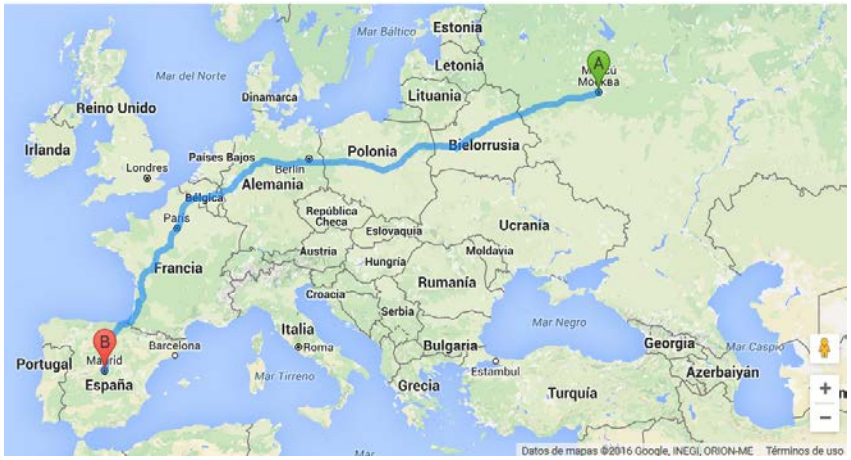
$$f_N(x^1) = \text{Max}_{m_1} \{r(x^1, m_1) + f_{N-1}[g(x^1, m_1)]\}, \ N \geq 2$$

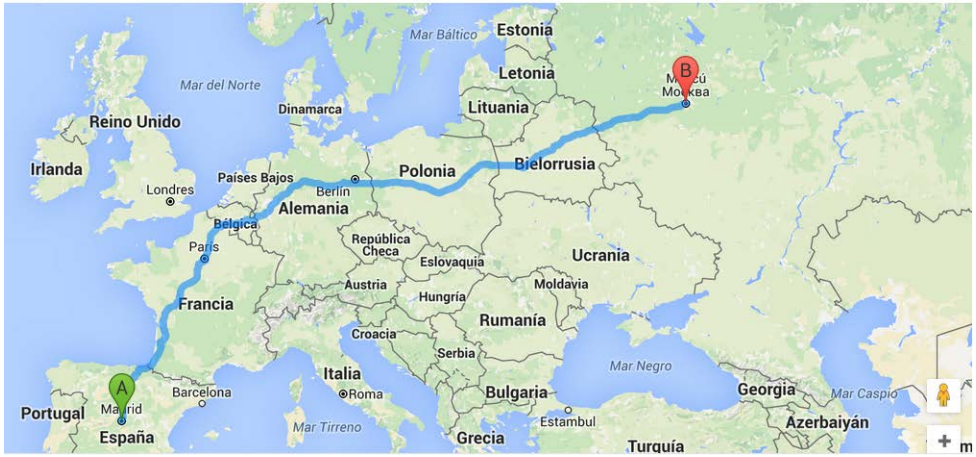
$$f_1(x^1) = \text{Max}_{m_1} \{r(x^1, m_1)\}, \ N = 1$$

El Método de la P.D.

- Para abordar la resolución de un problema con PD, habrá que comprobar,
- La **naturaleza N-etápica** del problema bajo consideración, para caracterizar la estructura de una solución óptima.
- Comprobar el **cumplimiento del POB** (condición necesaria para su aplicación).
- **Plantear una ecuación de recurrencia** que represente la forma de ir logrando etapa por etapa la solución óptima.
- **Encontrar la solución** óptima resolviendo problemas encajados para construir la solución.
- Con esos ingredientes, la solución del problema podrá construirse desde el primer estado hasta el último (**enfoque adelantado**), o bien a la inversa (**enfoque atrasado**)

El Método de la P.D.: Enfoques





El POB puede decirse que es conmutativo porque puede aplicarse hacia delante o hacia atrás.

La propiedad de una subpolítica de ser óptima, es independiente de que el estado inicial del proceso se considere a la izquierda del primer estado de esa subpolitica (**enfoque adelantado**), o a la derecha del mismo (**enfoque atrasado**).

El Método de la P.D.: Enfoque Adelantado

- Con el Enfoque Adelantado, el POB se aplicaría así:
- Supongamos S_0 el estado inicial del problema, y que hay que tomar n decisiones d_i , $1 \leq i \leq n$.
- Sea $D_1 = \{r_1, r_2, \ldots, r_j\}$ el conjunto de los posibles valores que podría tomar d_1.
- Sea S_i el estado del problema tras tomar la decisión r_i, $1 \leq i \leq j$.
- Sea Γ_i una sucesión de decisiones óptimas con respecto al estado del problema S_i.
- Cuando se verifica el POB, una sucesión de decisiones óptimas con respecto a S_0 es la mejor de las sucesiones de decisiones $r_i\Gamma_i$, $1 \leq i \leq j$.

P.D. vs. Enfoque Greedy

Programación Dinámica

- Se progresa etapa por etapa con sub-problemas que se diferencian entre si en una unidad en sus tamaños.
- Se generan muchas sub-sucesiones de decisiones
- Hay un gran uso de recursos (memoria).
- En cada etapa se comparan los resultados obtenidos, con los precedentes: Siempre obtienen la solución óptima.

Enfoque Greedy

- Se progresa etapa por etapa con sub-problemas que no tienen por que coincidir en tamaño.
- Solo se genera una sucesión de decisiones.
- La complejidad en tiempo suele ser baja.
- Como en cada etapa se selecciona lo mejor de lo posible sin tener en cuenta las decisiones precedentes, no hay garantía de obtener el óptimo.

P.D. vs. Divide y Vencerás

Programación Dinámica

- Se progresa etapa por etapa con sub-problemas que se diferencian en tamaño en una unidad.
- Se generan muchas sub-sucesiones de decisiones.
- Es aplicable cuando los sub-problemas comparten sub-problemas.
- Como los problemas están encajados, no se repiten cálculos.
- Siempre obtienen la solución óptima.

Divide y Vencerás

- Divide el problema en subproblemas independientes, los resuelve recursivamente y combina sus soluciones para obtener la solución total.
- Los sub-problemas no tienen ninguna característica común de tamaño, ni se sabe su número “a priori”.
- Como los problemas no tienen que ser independientes, pueden repetirse cálculos.
- Siempre se obtiene la solución óptima.

P.D. vs. Fuerza Bruta

- La solución se puede buscar enumerando todas las posibles sucesiones de decisiones.
- Se evalúa cada una de ellas, y se toma la mejor respecto al objetivo que se esté usando.
- La PD haciendo uso explícito del POB reduce la cantidad de cálculos que hay que hacer, evitando la enumeración de algunas sucesiones que posiblemente nunca puedan ser óptimas.

- Mientras que el número total de sucesiones de decisión diferentes es exponencial en el número de decisiones (si hay n elecciones para cada una de las d decisiones, entonces hay d^n posibles sucesiones de decisiones) **los algoritmos de PD suelen tener eficiencia polinomial**

Un ejemplo sencillo: el problema de la división óptima

- Se quiere dividir una cantidad positiva c en n partes de forma que el producto de esas n partes sea máximo.
- El problema es muy simple y podría resolverse fácilmente con métodos convencionales, pero nos sirve para ilustrar el método de la PD ya que puede considerarse n-etápico, y verifica el POB
- Sea
  - $f_n(c)$ el máximo producto alcanzable,
  - x el valor de la primera subdivisión, y
  - (c-x) el valor de las (n-1) partes restantes.

Un ejemplo sencillo: el problema de la división óptima

- De acuerdo con el POB, la ecuación funcional que describe este problema es,

$$f_n(c) = \text{Max}_{\,0\le x\le c}\,\{x f_{n-1}(c-x)\},\ n \ge 2$$

$$f_1(c) = c \text{ y } f_1(c-x) = c - x,\ n = 1$$

- Así, para n = 2,

$$f_2(c) = \text{Max}_{\,0\le x\le c}\,\{x f_1(c-x)\} = \text{Max}_{0\le x\le c}\,\{x(c-x)\}$$

- El máximo se obtiene para $x = c/2$.
- Así la política óptima, i.e., la subdivisión óptima para este proceso de decisión bi-etápico es $\{m_j\} = \{c/2, c/2\}$, siendo el máximo valor del producto

$$f_2(c) = (c/2)^2$$

Un ejemplo sencillo: el problema de la división óptima

- Si n = 3, el máximo producto alcanzable es,

$$f_3(c) = \text{Max}_{0\le x\le c}\{xf_2(c-x)\} = \text{Max}_{0\le x\le c}\{x(c-x)^2/4\}$$

y el valor máximo de x es x = c/3.

- La subdivisión de la parte restante, 2c/3, es un proceso de decisión bi-etápico.
- De acuerdo con el caso anterior, esa parte se divide en c/3 y c/3. Así la subdivisión óptima para el proceso 3-etápico es

$$\{m_j\} = \{c/3, c/3, c/3\}$$

siendo el valor máximo del producto

$$f_3(c) = (c/3)^3$$

Un ejemplo sencillo: el problema de la división óptima

- Todo esto nos lleva a conjeturar que la solución del problema, cuando n = k, sería

$$\{m_j\} = \{c/k, c/k, \ldots, c/k\}$$

siendo el máximo valor del producto

$$f_k(c) = (c/k)^k.$$

lo que puede demostrarse por inducción.

- De acuerdo con el POB, se sigue que,

$$f_{k+1}(c) = \text{Max}_{\,0\le x\le c}\ \{x f_k(c-x)\} = \text{Max}_{\,0\le x\le c}\ \{x(c-x)^k/k\ \}$$

lo que nos lleva a que x = c/(k+1), siendo el máximo valor del producto,

$$f_{k+1}(c) = [c/(k+1)]^{k+1}$$

Un ejemplo sencillo: el problema de la división óptima

- Así, por inducción matemática, la subdivisión óptima del proceso n-etápico puede obtenerse como

$$\{m_j\} = \{c/n, c/n, \ldots, c/n\}$$

con un valor máximo del producto

$$f_n(c) = (c/n)^n$$

- A pesar de ser un problema resoluble con otros métodos mas conocidos, el poder formularlo como uno de Programación Dinámica es lo que realmente da la clave sobre la gran utilidad de este enfoque.

Aplicacion de la P.D. al Diseño de Algoritmos

- PD se aplica en cuatro fases
  - Naturaleza n-etápica del problema
  - Verificación del POB
  - Planteamiento de una recurrencia
  - Cálculo de la solución (enfoque adelantado o atrasado)
- Las desarrollaremos en primer lugar sobre
  - Problema del Camino Mínimo
  - Problema de la Mochila