Programación Dinámica

Algoritmos

Algorítmica

Capítulo 4: Programación Dinámica

Tema 11: Algoritmos basados en P.D.

- El Problema del Camino Mínimo: Algoritmo de Floyd
- El Problema de la Mochila
- El Problema del Viajante de Comercio

Aplicación de la P.D. al Diseño de Algoritmos

La 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 sobre

- El Problema del Camino Mínimo
- El Problema de la Mochila
- El Problema del Viajante

Programación Dinámica

Algoritmos

El Problema de la Mochila Booleano

- El Problema de la Mochila (PM) es un ejemplo clásico de problema n-etápico y por tanto de PD.
- Tenemos n objetos y una mochila. El objeto i tiene un peso w_i y la mochila tiene una capacidad M.
- Si metemos en la mochila el objeto i, $1 \leq i \leq n$, se genera un beneficio de valor $p_i x_i$
- El objetivo es llenar la mochila de modo que se maximice el beneficio que produce el peso total de los objetos que se transportan, con la limitación de la capacidad de valor M.

$$\begin{aligned} \text{Max:} \quad & \textstyle\sum_{1\leq i\leq j} p_i x_i \\ \text{Sujeto a:} \quad & \textstyle\sum_{1\leq i\leq j} w_i x_i \leq M \\ & x_i = 0,1;\ 1\leq i\leq n \end{aligned}$$

1. Naturaleza n-etápica: El Problema de la Mochila 0-1

- En el PM su solución puede verse como el resultado de una sucesión de decisiones:
  - Tenemos que decidir los valores de x_i, $1 \leq i \leq n$.
- Así, primero tomaríamos una decisión sobre x_1, luego sobre x_2, después sobre x_3, etc.
- El problema es claramente n-etápico
- Una sucesión óptima de decisiones, verificando las restricciones del problema, será aquella que maximice la función objetivo.
- Conocemos la versión fraccional (enfoque greedy)

2. Comprobación del P.O.B.: Problema de la Mochila 0-1

- Notamos M(1,j,Y) al siguiente problema,

$$\text{Max: } \sum_{1\leq i\leq j} p_i x_i$$
$$\text{Sujeto a: } \sum_{1\leq i\leq j} w_i x_i \leq Y$$
$$x_i = 0,\ 1;\ 1\leq i\leq j$$

entonces el clásico problema de la mochila 0-1 se representa por M(1,n,M).

- Sea $y_1, y_2, \ldots, y_n$ una sucesión óptima de valores 0-1 para $x_1, x_2, \ldots, x_n$.
- Si $y_1 = 0$, entonces $y_2, \ldots, y_n$ debe ser una sucesión óptima para el problema M(2,n,M).
- Si no lo es: $y_1, y_2, \ldots, y_n$ no es una sucesión óptima de M(1,n,M).

2. Comprobación del P.O.B.: Problema de la Mochila 0-1

- Si $y_1 = 1$, entonces $y_2, \ldots, y_n$ debe ser una sucesión óptima para $M(2,n, M\text{-}w_1)$.
- Si no lo fuera, sería porque habría otra sucesión 0-1, $z_2, z_3, \ldots, z_n$ tal que

$$\sum_{2\leq i\leq n} w_i z_i \leq M - w_1 \ \text{ y } \sum_{2\leq i\leq n} p_i z_i > \sum_{2\leq i\leq n} p_i y_i$$

  y por tanto la sucesión $y_1, z_2, z_3, \ldots, z_n$ es una sucesión para el problema de partida con mayor valor, lo que es contradictorio.
- Por tanto puede aplicarse el POB, ya que se verifica.

3. Construccion de una ecuación recurrente: Problema de la Mochila 0-1

- Consideremos ahora el Problema de la Mochila 0-1.
- Sea $f_j(y)$ el valor de una solución óptima del problema Mochila (j+1, n, y).
- Claramente $f_0(M)$ es el valor de una solución óptima de Mochila (1,n,M).
- Las posibles decisiones para x_1 son 0 o 1 ($D_1 = \{0,1\}$).
- A partir del POB se sigue que

$$f_0(M) = \mathrm{Max}\ \{f_1(M),\ f_1(M-w_1) + p_1\}$$

Existen muchísimos algoritmos para resolver este problema, que es NP completo. Ver Horowitz-Sahni, 1978

Los algoritmos que lo resuelven son exponenciales. No se conocen para él algoritmos polinomiales.

Un enfoque de solución para el Problema de la Mochila 0-1

- Podemos obtener una solución para el Problema de la Mochila tomando una sucesión de decisiones sobre las variables $x_1, \ldots, x_n$.
- Una decisión sobre la variable x_i supone decidir que valor (0 o 1) ha de tomar.
- Supongamos que las decisiones sobre las x_i se hacen en el orden $x_n, \ldots, x_1$.
- Tras una decisión sobre x_n nos podemos encontrar ante una de estas dos posibilidades:
  - la capacidad restante de la mochila es M, y no se ha producido incremento de beneficio, o bien
  - la capacidad restante es $M - w_n$ y hemos aumentado el beneficio en p_n.

Un enfoque de solución para el Problema de la Mochila 0-1*

- Sea $f_j(X)$ el valor de una solución óptima del problema Mochila(1,j,X). Como se verifica el Principio del Óptimo, obtenemos,

$$f_n(M) = \text{Max } \{f_{n-1}(M),\ f_{n-1}(M-w_n) + p_n\ \}$$

que para i > 0 arbitrario, se generaliza a,

$$f_i(X) = \text{Max } \{f_{i-1}(X),\ f_{i-1}(X-w_i) + p_i\}$$

- Esta ecuación puede resolverse para $f_n(M)$ teniendo en cuenta que $f_0(X) = 0$ para todo X, y que $f_i(x) = \infty$, $x < 0$.
- Entonces se pueden calcular sucesivamente f_1, f_2, …, f_n.

* Horowitz-Sahni, 1978

El Problema del Camino Mínimo: Naturaleza n-etápica:

- El Problema del Camino Mínimo es otro ejemplo clásico de problema de PD
- En este caso, para encontrar ese camino desde un vértice i a otro j en un grafo G, veríamos etapa por etapa que vértice debe ser el segundo, el tercero, etc. hasta alcanzar el j.
- Una sucesión óptima de decisiones proporcionará entonces el camino de longitud mínima.
- Por tanto, este problema también tiene naturaleza n-etápica

2. Comprobación del P.O.B.: Problema del Camino Mínimo

- Sea $i, i_1, \dots i_k, j$ el camino mínimo desde i hasta j.
- Comenzando con el vértice inicial i, se ha tomado la decisión de ir al vértice i_1.
- Como resultado, ahora el estado del problema esta definido por el vértice i_1, y lo que se necesita es encontrar un camino desde i_1 hasta j.
- Está claro que la sucesión $i_1, i_2, \dots i_k, j$ debe constituir un camino mínimo entre i_1 y j. Si no:
- Sea $i_1, r_1, \dots r_q, j$ un camino mas corto entre i_1 y j
- Entonces $i, i_1, r_1, r_2, \dots r_q, j$ es un camino entre i y j que es mas corto que el camino $i, i_1, i_2, \dots i_k, j$.
- Como eso es una contradicción, se verifica el POB y también puede aplicarse a este problema.

3. Construcción de una ecuación recurrente: Caminos Mínimos

- Sea A_i el conjunto de los vértices adyacentes al vértice i.
- Para cada vértice $k \in A_i$ sea Γ_k el camino mínimo desde k hasta j.
- Entonces el camino mas corto desde i hasta j es el mas corto de los caminos del conjunto $\{i, \Gamma_k / k \in A_i\}$
- La recurrencia es trivial en este caso:

$$D_k(i,j) = \mathrm{Min}\ \{D_{k-1}(i,j), D_{k-1}(i,k) + D_{k-1}(k,j)\}$$

4. Caminos mínimos: Algoritmo de Floyd

- Sea G = (N,A) un grafo dirigido en el que N es su conjunto de nodos y A el de sus arcos. Cada arco tiene asociada una longitud, no negativa.
- El problema consiste en determinar el camino de longitud mínima que conecta **cualquier** par de nodos del grafo.
- Supondremos que los nodos están numerados de 1 a n, N = {1,2,…,n} y que la matriz L da la longitud de cada arco, de modo que L(i,i) = 0, L(i,j) ≥ 0 si i es distinto de j, y L(i,j) = ∞ si no existe el arco (i,j).
- El POB se aplica del siguiente modo:
  - Si k es un nodo en el camino mínimo que une i con j, entonces la parte de ese camino que va de i hasta k, y la del que va de k hasta j, es también óptima.

4. Caminos mínimos: Algoritmo de Floyd

- El Algoritmo consiste en lo siguiente:
- Construimos una matriz D que da la longitud del camino mínimo entre cada par de nodos.
- El algoritmo comienza asignando a cada casilla de D, la correspondiente de L y, entonces, realiza n iteraciones.
- Tras la iteración k, D da la longitud de los caminos mínimos que solo usan como nodos intermedios los del conjunto {1,2,...,k}.
- Después de n iteraciones tendremos, por tanto la solución buscada.

4. Caminos mínimos: Algoritmo de Floyd

- En la iteración k, el algoritmo tiene que chequear, para cada par de nodos (i,j), si existe o no un camino que pase a través de k que sea mejor que el actual camino mínimo que solo pasa a por nodos del conjunto {1,2,...,k-1}.
- Sea D la matriz después de la k-ésima iteración. El chequeo puede expresarse como,

$$D_k(i,j) = \text{Min}\ \{D_{k-1}(i,j),\ D_{k-1}(i,k) + D_{k-1}(k,j)\}$$

- donde hemos hecho uso del POB para calcular la longitud del camino mas corto que pasa a través de k.

4. Caminos mínimos: Algoritmo de Floyd

```
Procedimiento Floyd
Begin
        For i := 1 to n do
             For j := 1 to n do
                  D[i,j] := L[i,j];
        For i := 1 to n do
                  D[i,i] := 0;
        For k := 1 to n do
           For i := 1 to n do
              For j := 1 to n do
                          If D[i,k] + D[k,j] < D[i,j]
                          Then D[i,j] := D[i,k] + D[k,j]
  End;
```

Robert W. Floyd (1936-2001)

4. Caminos mínimos: Algoritmo de Floyd

- El algoritmo consume un tiempo $O(n^3)$.
- También podemos usar el algoritmo de Dijkstra, entonces aplicaríamos ese algoritmo n veces, eligiendo un nodo diferente como origen cada vez.
- Si queremos usar la versión de Dijkstra que trabaja con una matriz de distancias, el tiempo de cálculo total está en n x $O(n^2)$, es decir en $O(n^3)$.
- El orden es el mismo que para el algoritmo de Floyd, pero la simplicidad de este supone que, en la práctica, probablemente sea mas rápido.

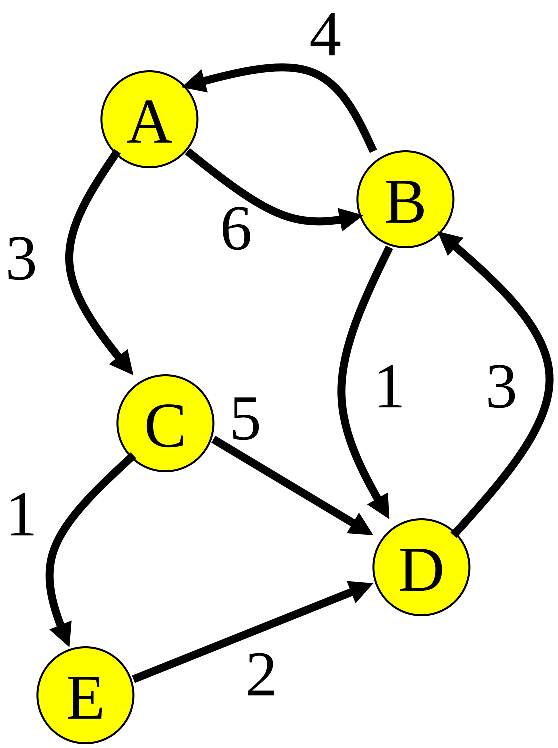
4. Caminos mínimos: Algoritmo de Floyd

- Si queremos saber por donde va el camino mas corto

```
Procedimiento Floyd-Warshall
Begin
    For i := 1 to n do
          For j := 1 to n do begin
              D[i,j] := L[i,j];
              P[i,j] := 0
          End;
    For i := 1 to n do
          D[i,i] := 0;
    For k := 1 to n do
         For i := 1 to n do
          For j := 1 to n do
                If D[i,k] + D[k,j] < D[i,j] then begin
                    D[i,j] := D[i,k] + D[k,j];
                   P[i,j] := k
                End
End;
```

```
Procedimiento Camino
Begin
   k := P[i,j];
   If k = 0 then Return;
   Path (i,k);
   Writeln (k);
   Path (k,j)
End;
```

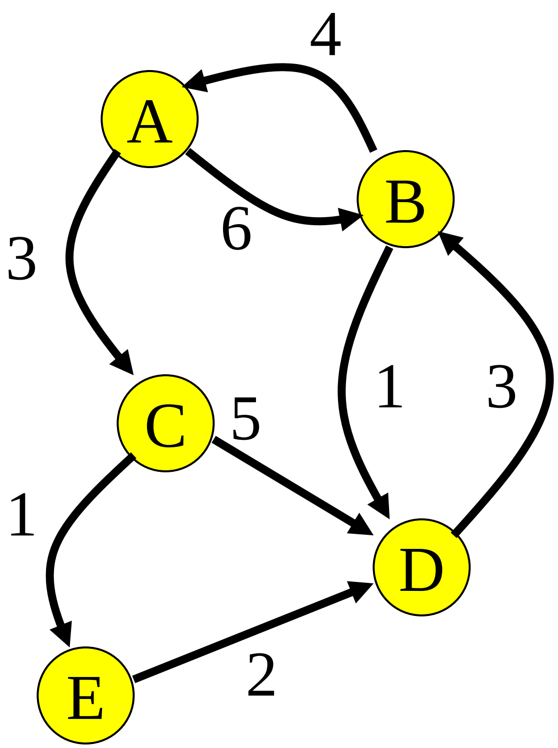
Ejemplo Algoritmo de Floyd



	A	B	C	D	E
A	0	6	3	∞	∞
B	4	0	∞	1	∞
C	∞	∞	0	5	1
D	∞	3	∞	0	∞
E	∞	∞	∞	2	0

Matriz de Distancias Inicial

Ejemplo Algoritmo de Floyd



	A	B	C	D	E
A	0	6	3	6	4
B	4	0	7	1	8
C	12	6	0	3	1
D	7	3	10	0	11
E	9	5	12	2	0

Matriz de Distancias Final

Programación Dinámica

Algoritmos

El Problema del Viajante de Comercio

El Problema del Viajante de Comercio

- Dado un grafo con longitudes no negativas asociadas a sus arcos, queremos encontrar el circuito mas corto posible que comience y termine en un mismo nodo, es decir, un camino cerrado que recorra todos los nodos una y solo una vez y que tenga longitud mínima (el Circuito Hamiltoniano Mínimo)
- Sea G = (N,A) un grafo dirigido,

  N = {1,2,...,n}, y L_{ij} la matriz de distancia,
  - $L(i,i) = 0$,
  - $L(i,j) \geq 0$ si i es distinto de j, y
  - $L(i,j) = \infty$ si no existe el arco (i,j).

El Problema del Viajante de Comercio

- Suponemos, sin perdida de generalidad, que el circuito comienza y termina en el nodo 1. El problema es claramente n-etápico
- El circuito, está constituido por un arco (1, j), seguido de un camino de j a 1 que pasa exactamente una vez a través de cada nodo de N - {1, ,j}.
- Si el circuito es óptimo, es decir, de longitud mínima, entonces ese es el camino de j a 1 y vale el Principio del Óptimo.

El Problema del Viajante de Comercio

- Sea $S \subseteq N - \{1\}$ un conjunto de nodos y consideremos un nodo mas $i \in N - S$
- Está permitido que $i = 1$ solo si $S = N - \{1\}$.
- Definimos el valor $g(i,S)$ para cada índice i, como la longitud del camino mas corto desde el nodo i al nodo 1 que pasa exactamente una vez a través de cada nodo de S.
- Usando esta definición,

$$g(1, N - \{1\})$$

- es la longitud de un circuito óptimo.

El Problema del Viajante de Comercio

- Por el POB vemos que

$$g(1, N-\{1\}) = \mathrm{Min}_{2\le j\le n} [L_{1j} + g(j, N - \{1,j\})]$$

- Mas generalmente, si i no es igual a 1, el conjunto S no es vacío y además es distinto de N - {1} e $i \notin S$,

$$g(i,S) = \mathrm{Min}_{j\in S} [L_{ij} + g(j, S - \{j\})]$$

- Además,
- $g(i,\varnothing) = L_{i1}$, $i = 2,3,\ldots n$
- Por tanto, los valores de g(i,S) se conocen cuando S es vacío

El Problema del Viajante de Comercio: Solución operativa

- Podemos aplicar

$$g(i,S) = \text{Min}_{\, j\in S} \, [L_{ij} + g(j, S - \{j\})]$$

para calcular g en todos los conjuntos S que contienen exactamente un nodo (que no es el 1).

- Luego aplicamos la misma formula para calcular g en todos los conjuntos S que contienen dos nodos (distintos del 1), y así sucesivamente.
- Cuando se conoce el valor de $g[j,N - \{1,j\}]$ para todos los nodos j, excepto para 1, utilizamos

$$g(1, N-\{1\}) = \text{Min}_{2\leq j\leq n} \, [L_{1j} + g(j, N - \{1, j\})]$$

- para calcular $g(1,N - \{1\})$, y definitivamente resolver el problema

El Problema del Viajante de Comercio: Tiempo de ejecución

- El tiempo que consumirá este algoritmo se hallará a partir de las expresiones anteriores.
- Para calcular $g(j,\varnothing)$ hay que hacer n-1 consultas de una tabla.
- Para calcular $g(i,S)$

$$g(i,S) = \mathrm{Min}_{j \in S} [L_{ij} + g(j, S - \{j\})]$$

- Hay que calcular todas las $g(i,S)$ tales que $1 \leq |S| = k \leq n-2$, lo que supone realizar,

$$(n-1) \times C_{n-2,k} \times k$$

adiciones.

El Problema del Viajante de Comercio: Tiempo de ejecución

- De esas $(n-1) \times C_{n-2,k} \times k$ adiciones,

  - n-1 corresponden a los posibles valores que puede tomar la variable i,
  - k provienen de los valores que puede tomar la variable j,
  - y las combinaciones restantes son todos los conjuntos que podemos formar de n-2 elementos tomados de k en k.

- Calcular g(1, N-{1}) implica hacer n-1 adiciones.
- Estas operaciones pueden usarse como referencia para calcular la eficiencia del algoritmo, y así el tiempo que se lleva el algoritmo en cálculos es,

$$O[2(n-1) + \sum_{k=1..(n-2)} (n-1)\times k\times \mathbf{C}_{n-2,k}] = O(n^2 2^n)$$

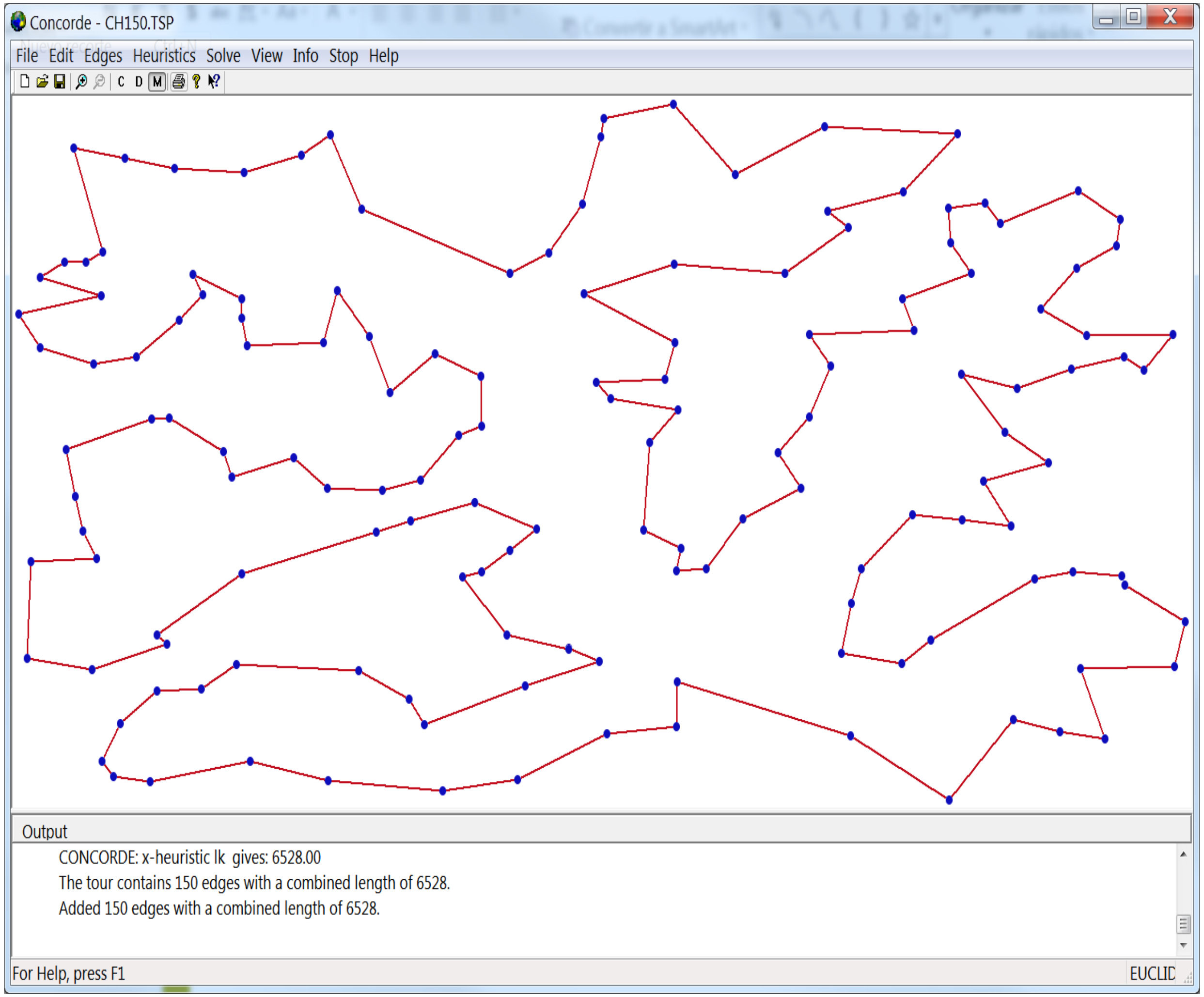
ya que

$$\sum_{k=1..r} k \times \mathbf{C}_{r,k} = r2^{r-1}$$

Programación Dinámica

Algoritmos

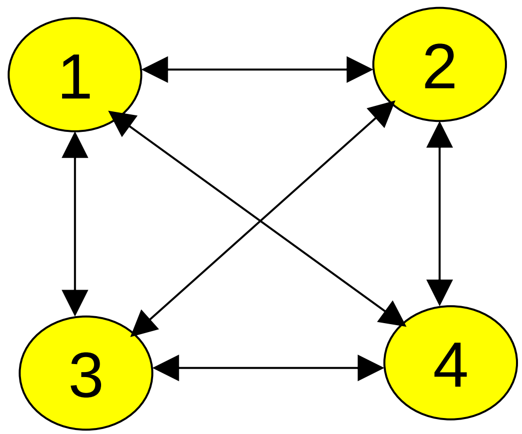
El Problema del Viajante de Comercio



http://www.math.uwaterloo.ca/tsp/concorde/index.html

El Problema del Viajante de Comercio: Ejemplo

- Consideremos el grafo



$$\begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 8 & 9 & 0 \end{bmatrix}$$

$g(2, \emptyset) = c_{21} = 5$; $g(3, \emptyset) = c_{31} = 6$; $g(4, \emptyset) = c_{41} = 8$

Y luego,

$g(2,\{3\}) = c_{23} + g(3, \emptyset) = 15$; $g(2,\{4\}) = 18$

$g(3,\{2\}) = 18$; $g(3,\{4\}) = 20$

$g(4,\{2\}) = 13$; $g(4,\{3\}) = 15$

El Problema del Viajante de Comercio: Ejemplo

- Calculamos g(i,S) para conjuntos S de cardinal 2, $i \neq 1$, $1 \notin S$ e $i \notin S$:

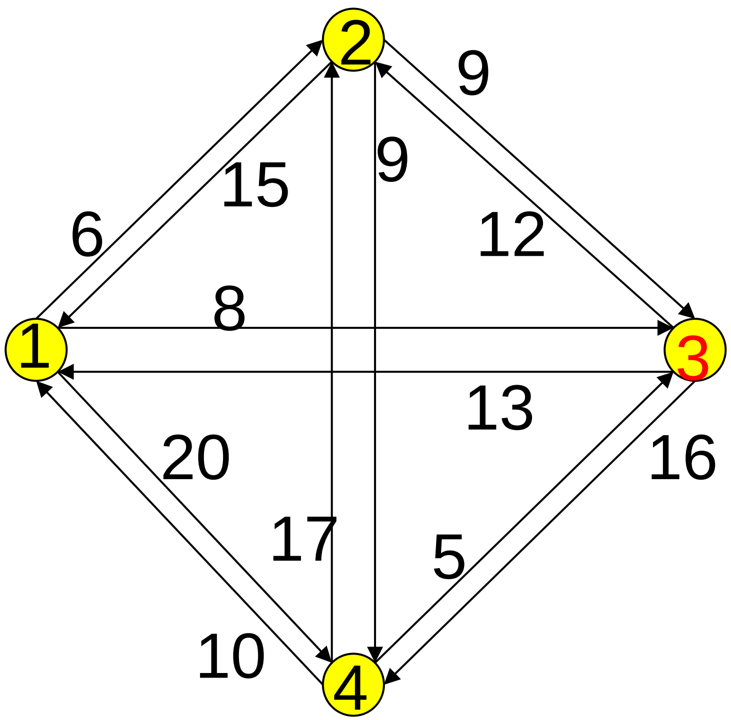
$$g(2, \{3,4\}) = \text{Min } [c_{23} + g(3,\{4\}),\ c_{24} + g(4,\{3\})] = 25$$
$$g(3, \{2,4\}) = \text{Min } [c_{32} + g(2,\{4\}),\ c_{34} + g(4,\{2\})] = 25$$
$$g(4, \{2,3\}) = \text{Min } [c_{42} + g(2,\{3\}),\ c_{43} + g(3,\{2\})] = 23$$

- y finalmente,

$$g(1,\{2,3,4\}) = \text{Min } [c_{12} + g(2, \{3,4\}),\ c_{13} + g(3, \{2,4\}),\ c_{14} + g(4, \{2,3\})] =$$
$$= \text{Min } \{35,40,43\} = 35$$

que es la longitud del circuito buscado.

El Problema del Viajante de Comercio: Ejemplo de enfoque atrasado



Sea k = 3.

$g(3,\{1,2,3,4\}-\{3,1\})$ es el camino mas corto de 3 a 1 que pasa por 2 y 4

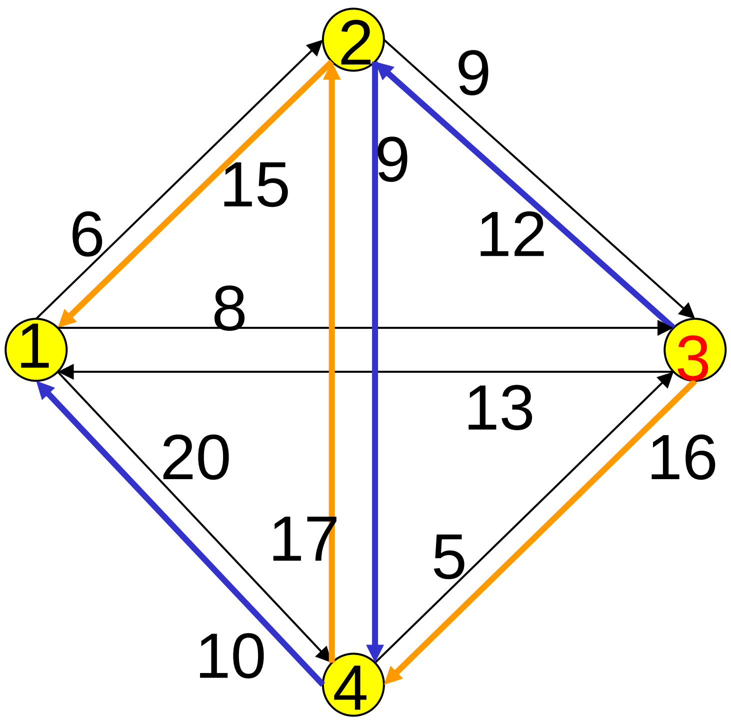
Y cual es ese camino?

Simplemente es el mas corto de los posibles:

$$c_{3i}+g(i,V-\{3,1,i\})$$

con i = 2 y 4.

El Problema del Viajante de Comercio: Ejemplo de enfoque atrasado



Así,

$c_{32} + g(2,\{4\}) =$

$= 12 + (9+10)$

$= 31$

o

$c_{34} + g(4,\{2\}) =$

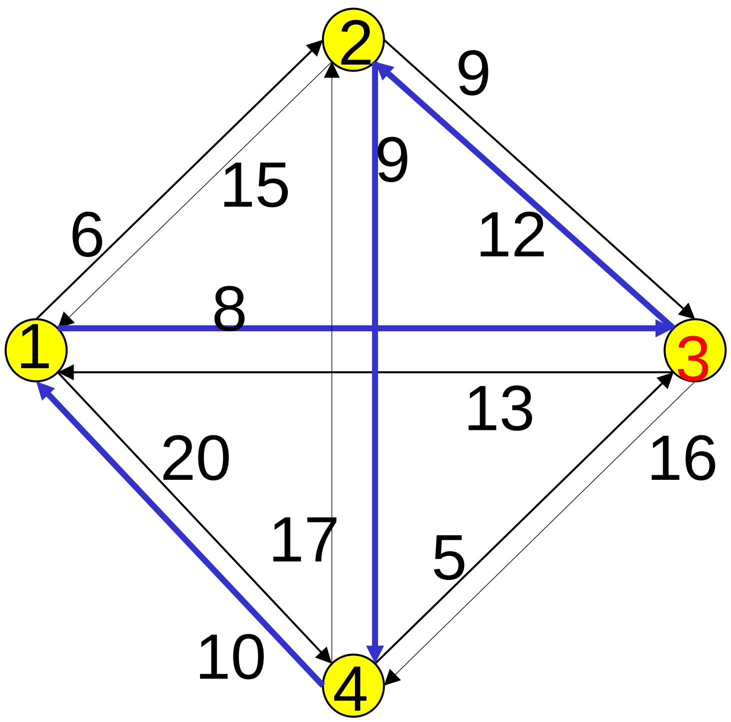
$= 16 + (17+15)$

$= 48$

Luego

$g(3,\{2,4\}) = 31.$

El Problema del Viajante de Comercio: Ejemplo de enfoque atrasado



Así g(1,{1,2,3,4}-1) podría ser

c_{13} +g(3,{2,4}) =

= 8 + 31

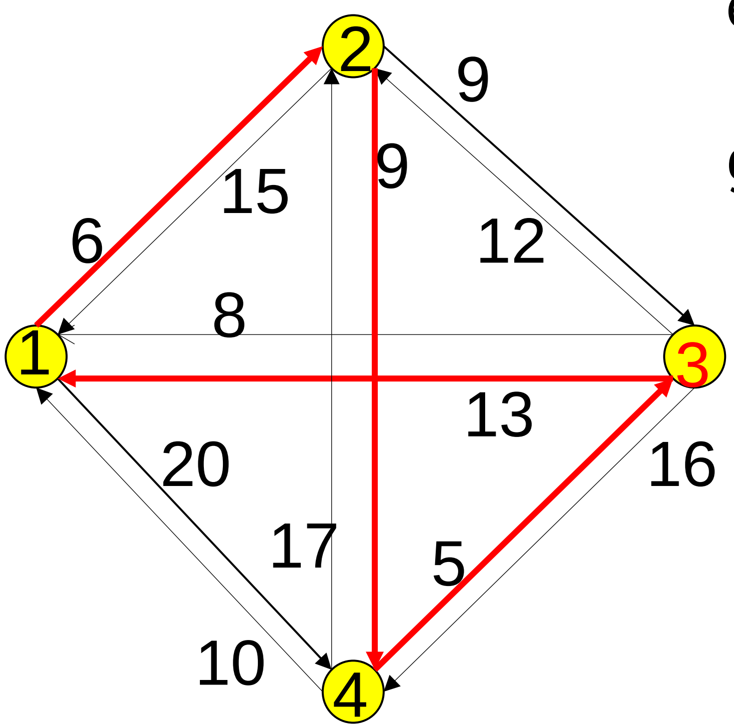
= 39.

Pero para resolver el problema necesitamos encontrar también

g(2,{3,4}) y g(4,{2,3})

y escoger el menor.

El Problema del Viajante de Comercio: Ejemplo de enfoque atrasado



Definitivamente, la solución es:

$$\begin{aligned} g(1,\{2,3,4\}) &= c_{12} + g(2,\{3,4\}) \\ &= 6 + 27 \\ &= 33. \end{aligned}$$

El Problema del Viajante de Comercio: Ejemplo de enfoque atrasado

$$g(1,\{2,3,4\})$$

$$c_{12}+g(2,\{3,4\}) \qquad c_{13}+g(3,\{2,4\}) \qquad c_{14}+g(4,\{2,3\})$$

$$c_{23}+g(3,\{4\}) \quad c_{24}+g(4,\{3\}) \quad c_{32}+g(2,\{4\}) \quad c_{34}+g(4,\{2\}) \quad c_{42}+g(2,\{3\}) \quad c_{43}+g(3,\{2\})$$