Algorísmica Avançada
Complexitat

Sergio Escalera

Ramificació i poda

- La **clau**: funció de **prioritat i cota**
- Problema del viatjant de comerç
  - → Passar per tots els nodes minimitzant el cost de la ruta, passant només un cop per cada node i acabant en el node de sortida (circuit ***hamiltonià***)

Ramificació i poda

- Formalització:
  - Sean $G=(V,A)$ un grafo orientado,
    $V=\{1,2,\ldots,n\}$,
    $D[i,j]$ la longitud de $(i,j)\in A$,
    $D[i,j]=\infty$ si no existe el arco (i,j).
  - El circuito buscado empieza en el vértice 1.

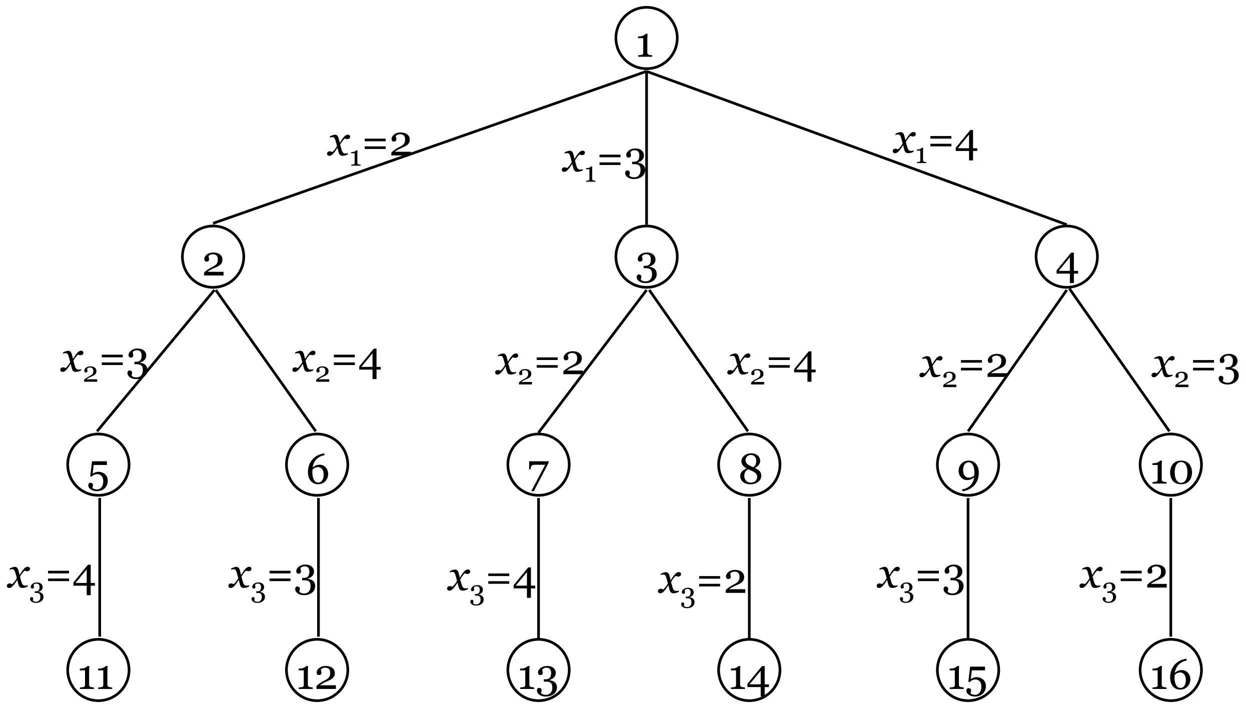
  - Candidatos:
    $E = \{ 1,X,1 \mid X$ es una permutación de $(2,3,\ldots,n) \}$
    $|E| = (n-1)!$

  - Soluciones factibles:
    $E = \{ 1,X,1 \mid X = x_1,x_2,\ldots,x_{n-1}$, es una permutación de $(2,3,\ldots,n)$ tal que $(i_j,i_{j+1})\in A$, $0<j<n$, $(1, x_1) \in A$, $(x_{n-1},1) \in A\}$

  - Funcion objetivo:
    $F(X)=D[1,x_1]+D[x_1, x_2] + D[x_2, x_3]+\ldots+D[x_{n-2}, x_{n-1}]+D[x_n,1]$

Ramificació i poda

- Representació per |V|=4



Ramificació i poda

- Definición de una *cota*(X,k) muy sencilla:
  - Suma de aristas ya escogidas
  - $cota(X,k)=D[1,X[1]]+\Sigma_{i=1..k-2}\, D[X[i],X[i+1]]$
- Ejemplo: (n=5)

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix}$$

Ramificació i poda

- Si se elige *t* como el mínimo de los elementos de la fila (columna) *i*-ésima y se resta *t* de todos los elementos de esa fila (columna), la fila resultante es **reducida**.
- La cantidad total *L* restada de filas y columnas es una cota inferior de la longitud de un hamiltoniano de longitud mínima

Reducción de la matriz, $L = 25$.

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix} \quad \begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

Ramificació i poda

- Cálculo de la cota inferior para los nodos distintos de la raíz y de las hojas:
  - Sea A la matriz de distancias reducida para el nodo y.
  - Sea x un hijo de y que corresponda a incluir el arco (i,j) en el recorrido y que no sea hoja.
  - La matriz B reducida para x, y por tanto cota(x), se calcula de la siguiente forma:

1. Cambiar todos los elementos de la fila i y de la columna j de A por ∞.
   - Esto evita el incluir más arcos que salgan de i o lleguen a j.
2. Cambiar el elemento (j,1) de A por ∞
   - Esto evita considerar el arco (j,1).
3. B es la matriz que se obtiene al reducir todas las filas y columnas de la matriz resultante (excepto aquéllas formadas sólo por "∞").

- Si r es el valor total restado en el paso (3): cota(x)=cota(y) + D[i,j] + r

Ramificació i poda

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix}$$

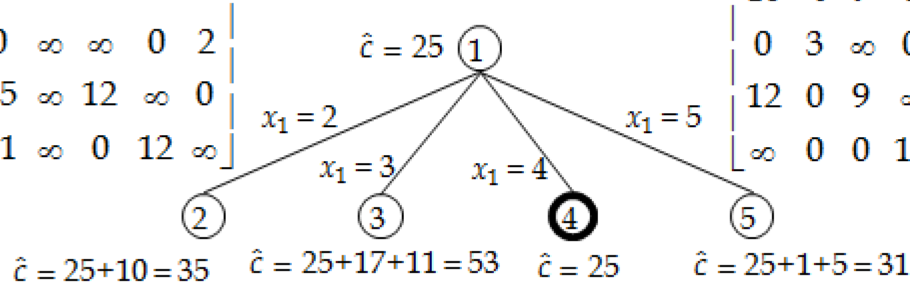
Grafo original.

$$\begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

Matriz reducida, $L = 25$.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & 2 & 0 \\ 0 & \infty & \infty & 0 & 2 \\ 15 & \infty & 12 & \infty & 0 \\ 11 & \infty & 0 & 12 & \infty \end{bmatrix}$$

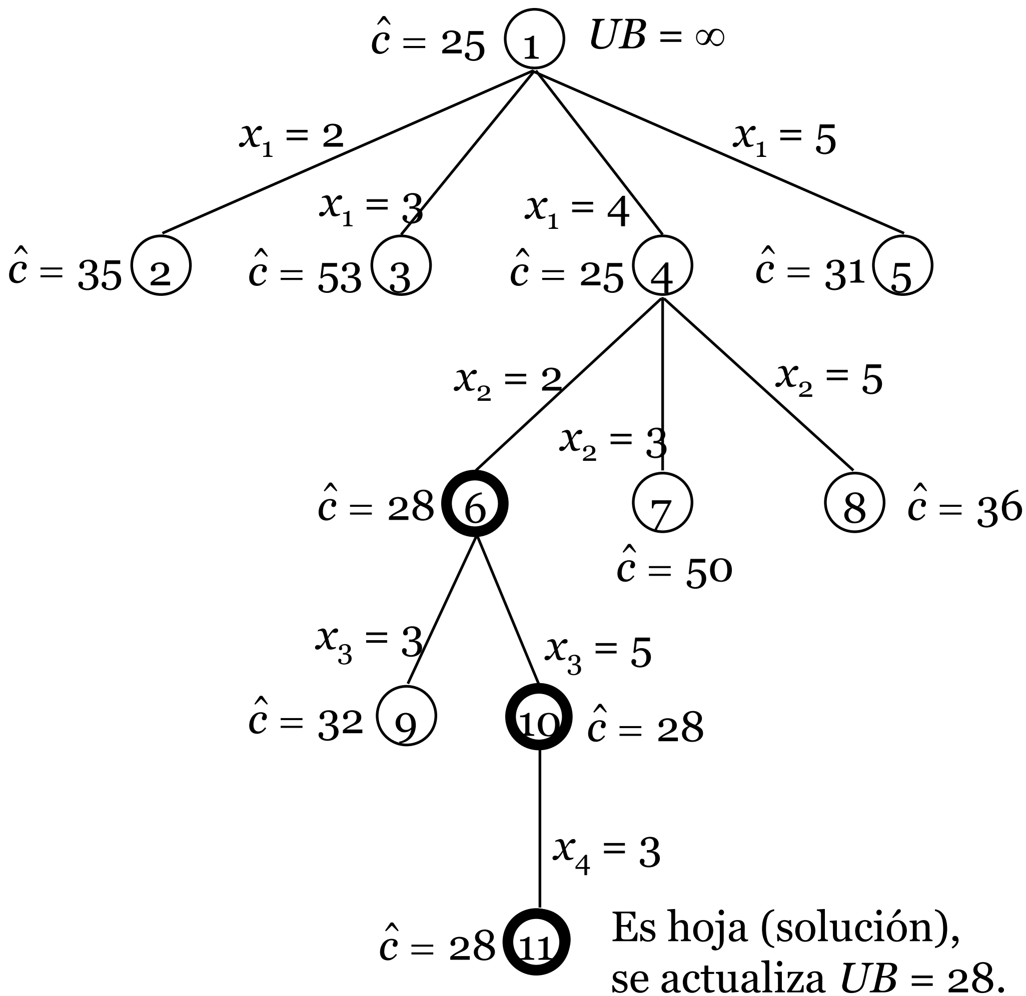
$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 10 & \infty & 9 & 0 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 12 & 0 & 9 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix}$$



$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 4 & 3 & \infty & \infty & 0 \\ 0 & 0 & \infty & 12 & \infty \end{bmatrix}$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ 0 & 3 & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{bmatrix}$$

Ramificació i poda



El siguiente nodo en curso sería el 5, pero cota(5)>UB luego el algoritmo termina y el hamiltoniano mínimo es 1,4,2,5,3,1.

Complexitat

- Fins ara hem vist algorismes que es poden resoldre en temps polinòmic (n, n^2, n^3, etc)
- No obstant, hi ha problemes on no existeix encara cap algorisme polinòmic per resoldre'ls, i hem de fer servir solucions exponencials (que no són molt més útils que la cerca exhaustiva)

Complexitat

- Exemple

  SATISFIABILITY

  $$(x \vee y \vee z)\ (x \vee \overline{y})\ (y \vee \overline{z})\ (z \vee \overline{x})\ (\overline{x} \vee \overline{y} \vee \overline{z})$$

  conjunctive normal form (CNF)

- Podem veure que aquest exemple de SAT no té solució, però quan no és evident, necessitem testejar el conjunt exponencial de 2^n possibles solucions per a *n* variables.

Complexitat

- Exemple

SATISFIABILITY

$$(x \vee y \vee z)\ (x \vee \overline{y})\ (y \vee \overline{z})\ (z \vee \overline{x})\ (\overline{x} \vee \overline{y} \vee \overline{z})$$

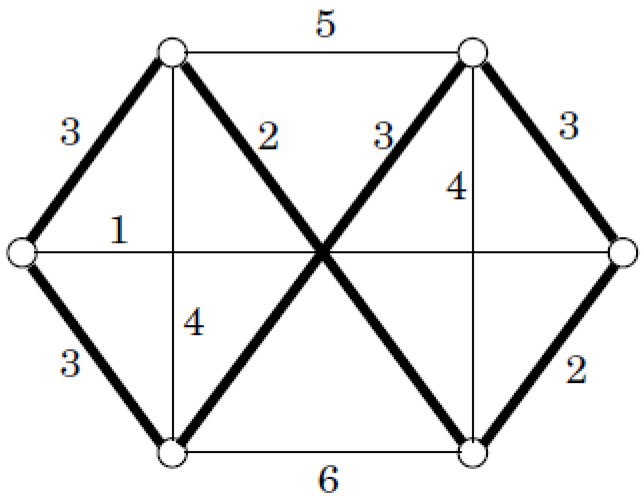
conjunctive normal form (CNF)

- El problema s'ha estudiat durant més de 50 anys:
  - Si totes les clàusules contenen com a molt un literal positiu (Horn) existeix solució
  - Si totes les clàusules contenen 2 literals (2SAT) es pot resoldre per teoria de grafs per components connexos
  - **Si permetem que hi hagi 3 literals (3SAT) estem de nou en un problema difícil de solucionar**

Complexitat

- **The traveling salesman problem (TSP)**

•Volem recórrer tots els vèrtex només un cop amb el mínim cost possible.
•És un problema de **cerca** i el podem resoldre en temps polinomial amb els algorismes dels capítols anterior.
•Però no sabem si el resultat és **òptim**!! Podem incloure restricció de que el camí sigui com a molt de cost ***b***
•Amb programació dinàmica es pot fer (també ho hem vist amb ramificació i poda), però així encara la complexitat és exponencial!



Complexitat

- Definicions $C(I, S)$
- Problema de **cerca**: Si I correspon a les dades de la instància del problema a solucionar i S és una possible solució, C retorna si es satisfà o no el problema de cerca en un temps lineal acotat a $|I|$
- El conjunt dels problemes de cerca s'anomena ***NP***
- El conjunt de tots els problemes de cerca que es poden resoldre en temps polinomial a $|I|$ s'anomena ***P***
- Un problema de cerca C a NP és ***NP-complet*** si qualsevol altre problema de cerca es pot reduir a C en temps polinomial. (NP-complet són els problemes més difícils de resoldre de NP).

Complexitat

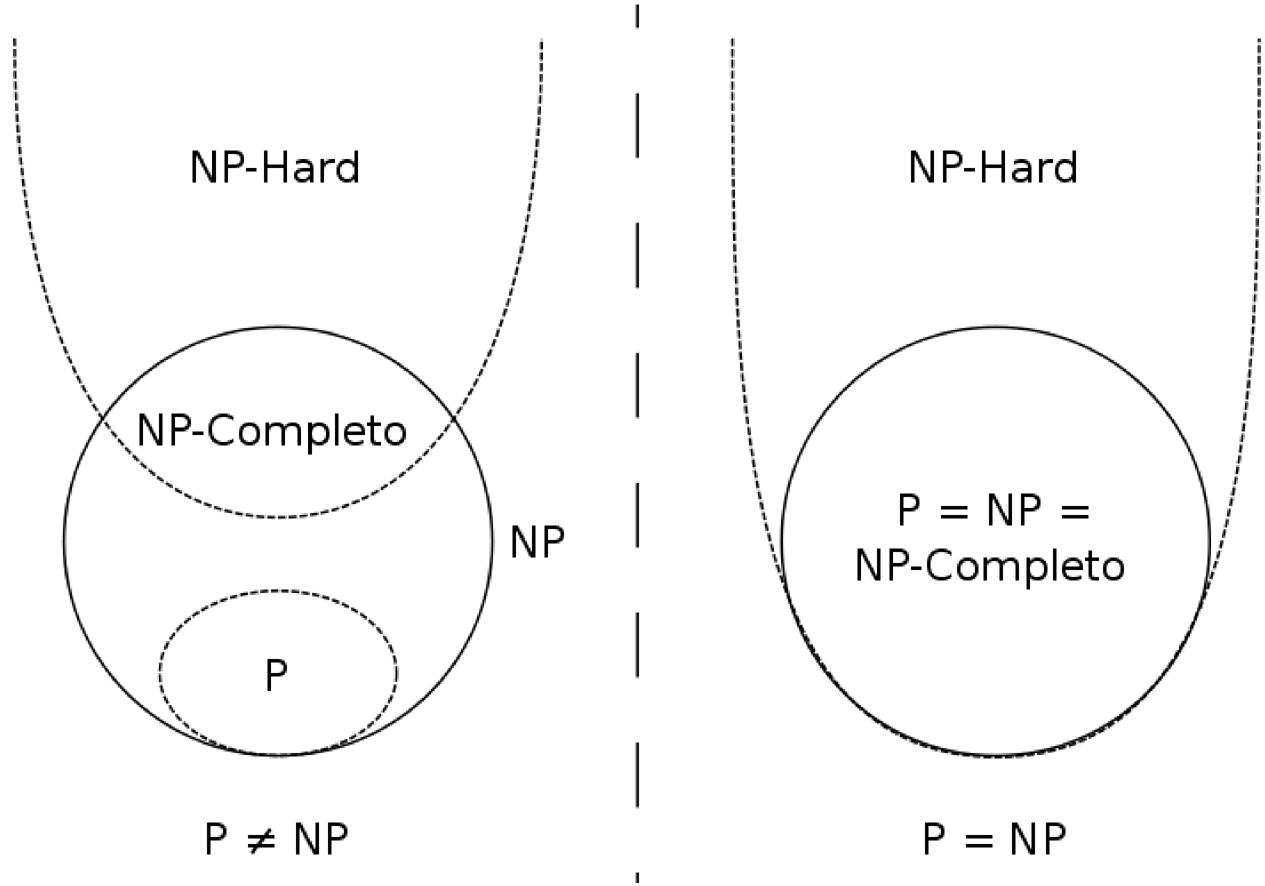
$\mathbf{P \neq NP?}$

- Es creu que sí...
- S'ha demostrar?
  - No!
- Per què?
  - Perquè la demostració també és NP-complet ☺

Complexitat

- NP-Hard
  - → Són aquells tipus de problemes que es poden **reduir** als més complexes de NP (ex. SAT3) en temps polinòmic. Cobreixen els problemes de decisió, de cerca o optimització.
  - D'aquesta forma, si trobem una solució en temps polinòmic per SAT3 també trobem una solució polinòmica per a tot problema NP (i llavors P=NP).

Complexitat



Complexitat

- Al menys els algorismes NP tenen una solució, encara que la complexitat sigui exponencial.
- Existeixen problemes sense solució.

- Exemple:
  - Donada una funció polinomial de diferents variables

$$x^3yz + 2y^4z^2 - 7xy^5z = 6,$$

  - Hi ha valors enters per a *x*, *y*, *z* que la solucionen?

Complexitat

- Com treballem amb els problemes NP-complet?
  - En el món laboral molts dels problemes que sorgeixen no es cobreixen amb les solucions algorísmiques explicades en aquest curs
  - És més, encara que ho siguin és difícil veure l’equivalència, i requereix pràctica
  - Però quan podem veure que el problema és NP-complet, com el solucionem?

Complexitat

- Com treballem amb els problemes NP-hard?
  - Podem fer ús d'algorismes aproximats

→ Sabem que el resultat és pràcticament equivalent a l'òptim, i en el pitjor cas servirà per cobrir els nostres propòsits

→ També podem incloure heurístiques basant-nos en la nostra experiència que serviran per guiar la cerca i acostar l'espai de possibilitats

Complexitat

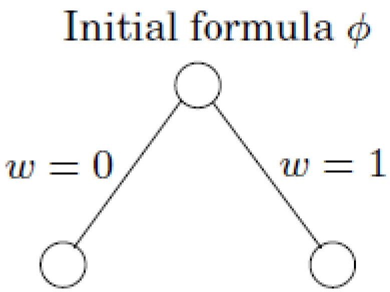
- Exemple: ramificació i poda (ja ho hem vist)
  - Fórmula booleana

$$\phi(w, x, y, z)$$

  - Clàusules

$$(w \vee x \vee y \vee z),\ (w \vee \overline{x}),\ (x \vee \overline{y}),\ (y \vee \overline{z}),\ (z \vee \overline{w}),\ (\overline{w} \vee \overline{z})$$

  - Fem un arbre de solucions parcials!

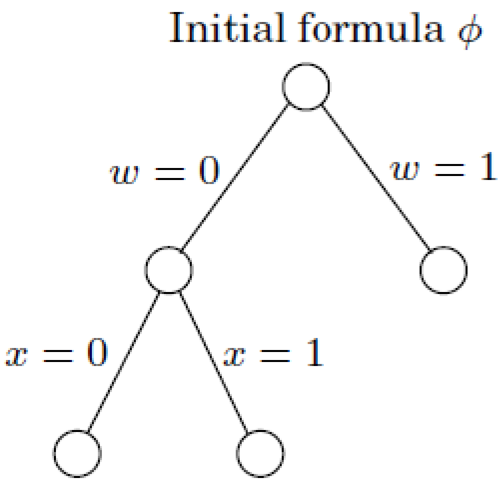


Complexitat

- Exemple: backtracking
  - Continuem ramificant
  - Ara podem podar per que no satisfem una clàusula:

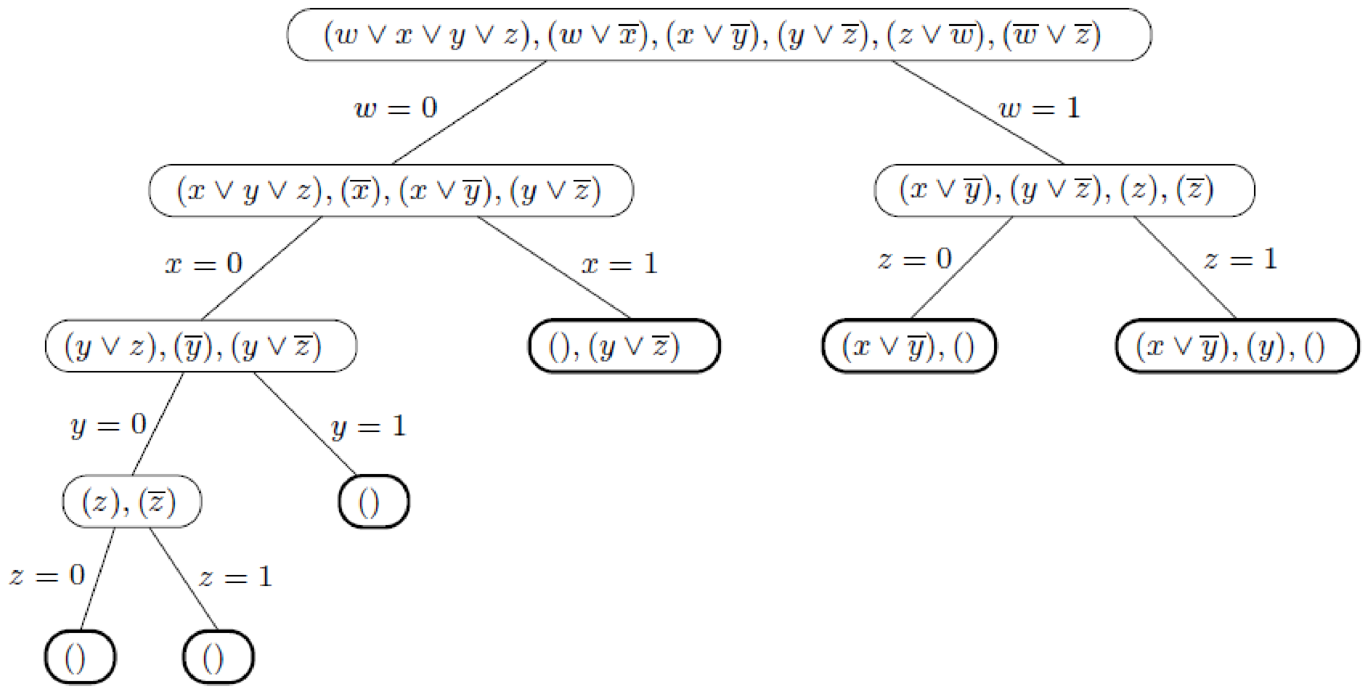
$$w = 0, x = 1 \qquad (w \vee \overline{x})$$

  - I retornem (backtracking) als altres 2 nodes actius



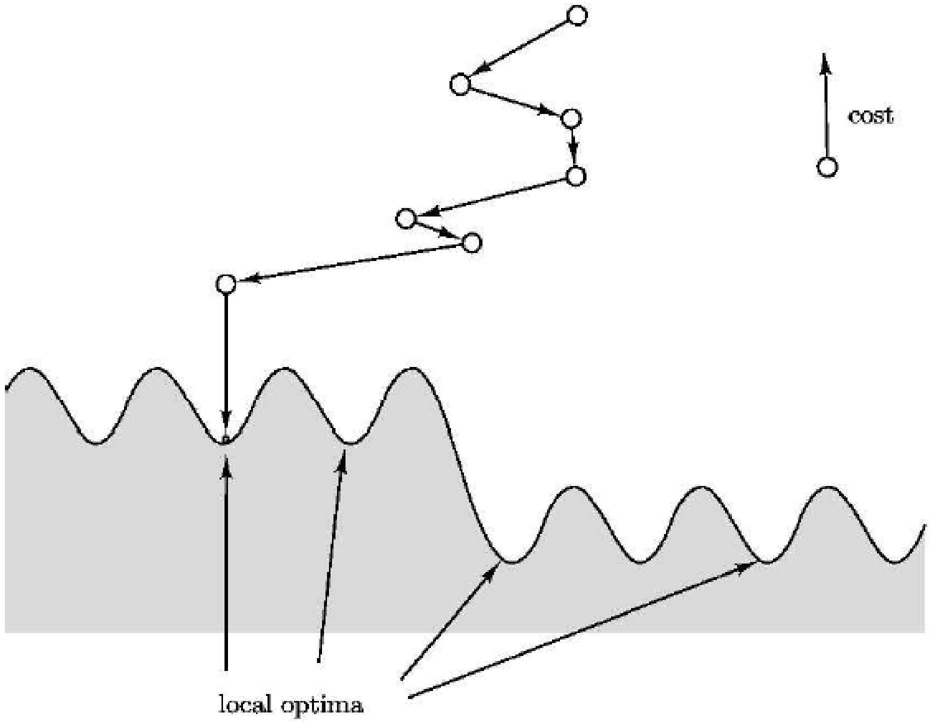
Complexitat

- Exemple: backtracking
  - Hem reduit considerablement el conjunt de possibilitats per arribar més ràpidament a una solució del problema



Complexitat

- Exemple d'execució d'una heurística per a cerca local per a trobar una solució aproximada



Complexitat

- El problema de les solucions locals:
  - → Amb alguns d'aquests algorismes podem arribar a solucions (mínims locals) que no són els òptims
  - → Una possible solució son les execucions aleatòries per corregir algunes solucions que arriben als mínims locals

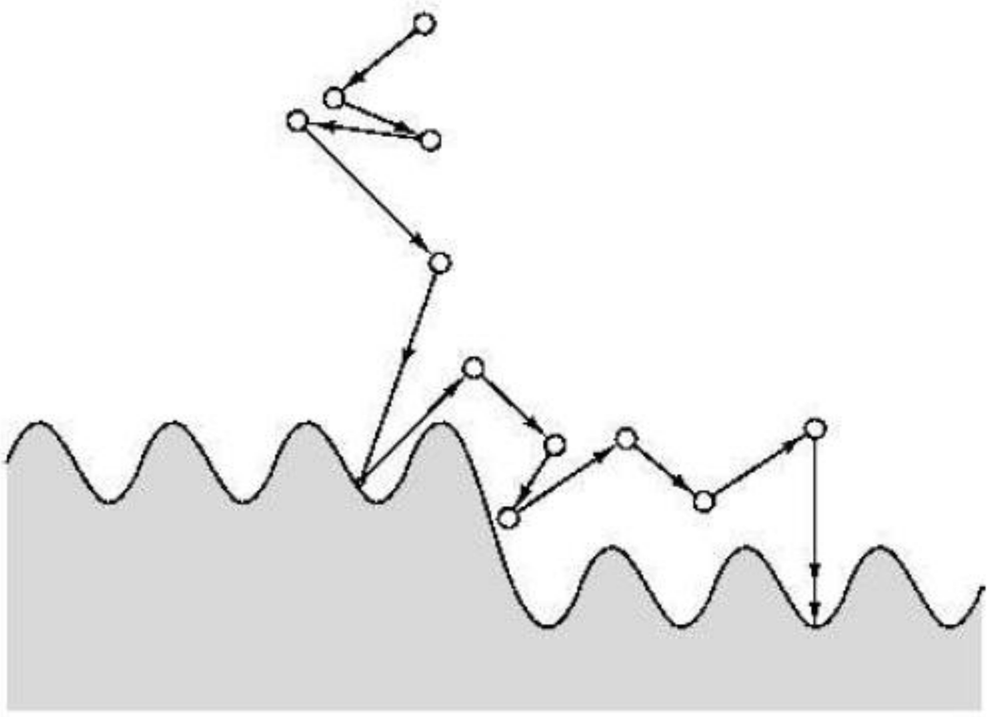
Complexitat

- Exemple: simulating annealing
  - → El problema del exemple anterior es que ens podem moure constantment al voltant d'un mateix mínim local. Una possible solució es permetre en certs moments salts en la cerca suficientment grans

```
let s be any starting solution
repeat
   randomly choose a solution s' in the neighborhood of s
   if Δ = cost(s') − cost(s) is negative:
      replace s by s'
```

Complexitat

- Exemple: simulating annealing



Programació dinàmica

- Exercici

```
for i = 0,1,2,...,m:
    E(i,0) = 2i
for j = 1,2,...,n:
    E(0,j) = j
for i = 1,2,...,m:
    for j = 1,2,...,n:
        E(i,j) = min{E(i-1,j)+2, E(i,j-1)+1, E(i-1,j-1)+diff(i,j)}
return E(m,n)
```

		A	L	G	O	R	I	S	M	I	C	A
	0	1										
A	2	0										
V												
A												
N												
Ç												
A												
D												
A												

$$\gamma(i,j,k) = d(i,j,k) + \min\{\gamma((i-1,j-1),(i-1,j),(i,j-1)\times\{1,\ldots,K\})\}$$

Programació dinàmica

1 1 1 OO 1 O 1

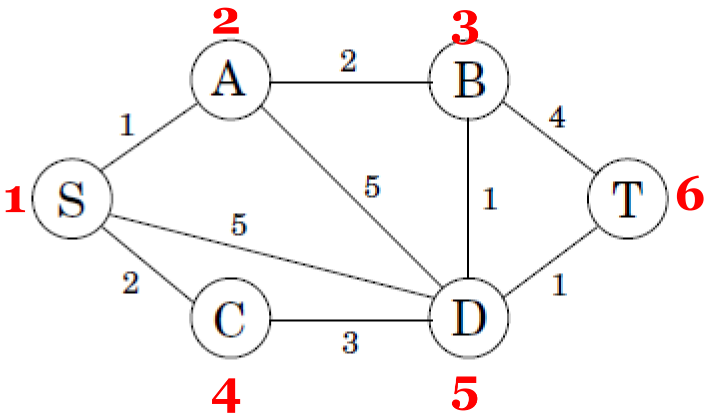
O O 1

1 1 1

	0	1	2	3	4	5	6	7	8	9
1	1	0	0	0	0	0	0	0	1	1
1	2	0	0	0	0	0	0	0	1	1
1	3									
1	4									
O	5									
O	6									
O	7									
1	8									

Programació dinàmica

- Floyd-Warshall




```
for i = 1 to n:
    for j = 1 to n:
        dist(i, j, 0) = ∞

for all (i, j) ∈ E:
    dist(i, j, 0) = ℓ(i, j)
for k = 1 to n:
    for i = 1 to n:
        for j = 1 to n:
            dist(i, j, k) = min{dist(i, k, k − 1) + dist(k, j, k − 1),  dist(i, j, k − 1)}
```

Ramificació i poda - Exercici

- Fes l'arbre de ramificació i poda de la següent taula (seguint el problema de l'exemple anterior). Numera els passos i actualitza els valors de les cotes.

	A	B	C	D	E
a	1	13	3	18	2
b	3	5	9	20	6
c	5	10	2	17	5
d	7	2	10	21	10
e	9	10	15	16	4

Ramificació i poda - Exercici

- Fes l'arbre de ramificació i poda de la següent taula (seguint el problema de l'exemple anterior). Numera els passos i actualitza els valors de les cotes.

	A	B	C	D
a	1	13	2	18
b	3	5	9	20
c	5	10	12	17
d	7	2	10	13