

Algorísmica Avançada

Complexitat

Sergio Escalera

A series of horizontal lines of varying lengths and colors (teal, light blue, and white) extending from the left edge of the slide towards the right, positioned below the author's name.

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

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

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

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

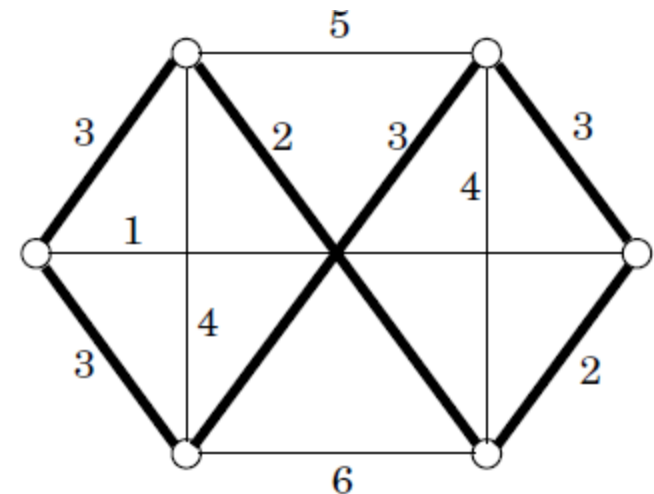
conjunctive normal form (CNF)

- 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 $\mathcal{C}(I, S)$
- 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

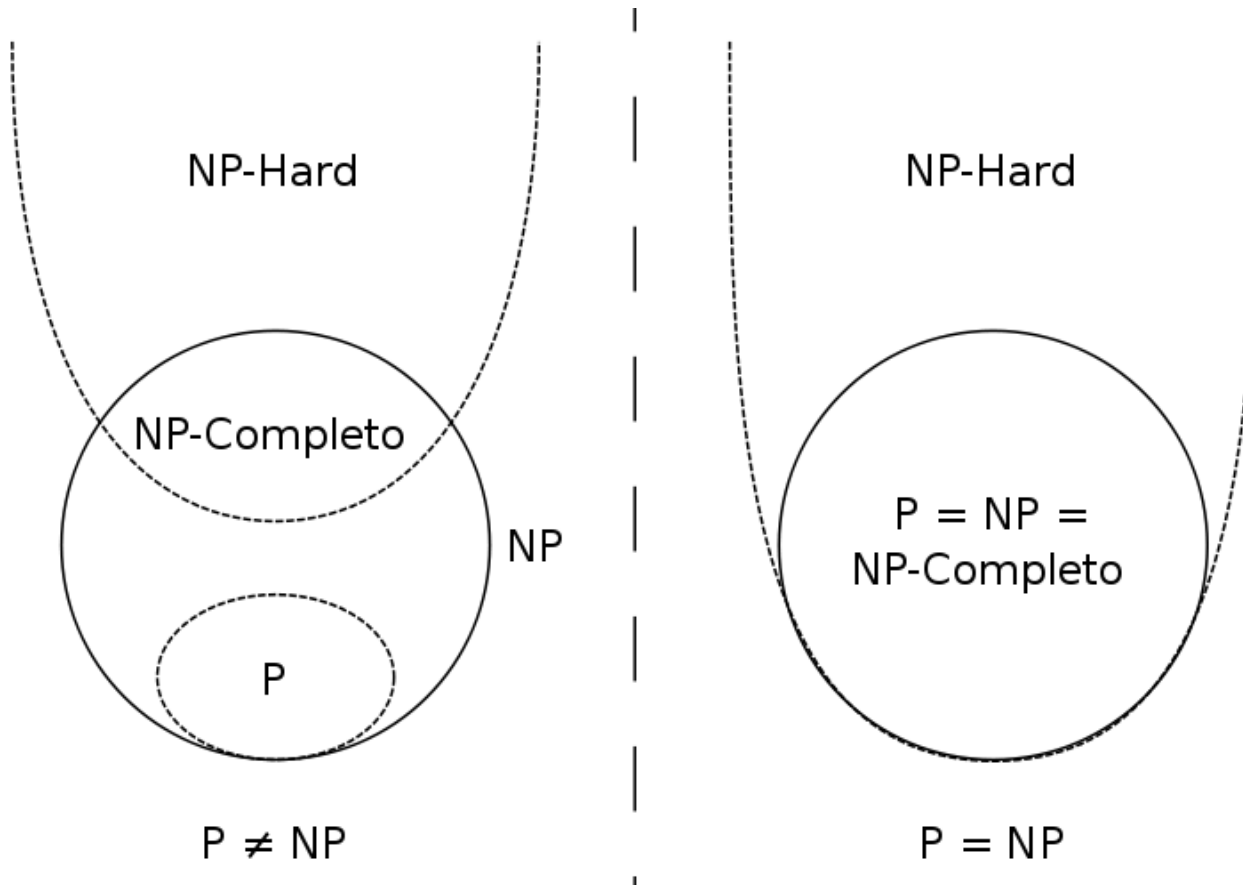
P \neq **NP**?

- 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 complexos 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

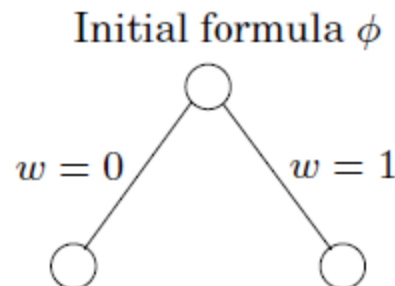
- Exemple: ramificació i poda (ja ho hem vist)
 - Fórmula booleana

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

- Clàusules

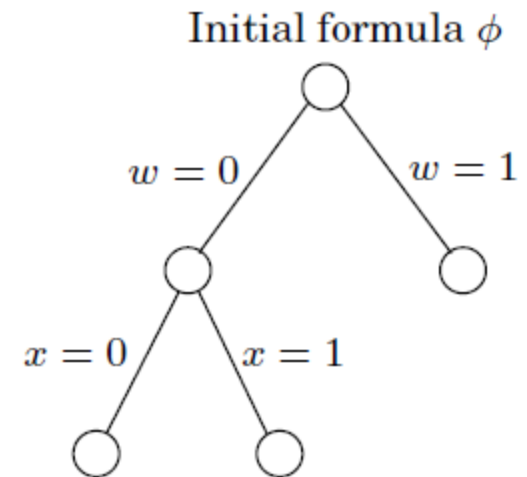
$$(w \vee x \vee y \vee z), (w \vee \bar{x}), (x \vee \bar{y}), (y \vee \bar{z}), (z \vee \bar{w}), (\bar{w} \vee \bar{z})$$

- Fem un arbre de solucions parcials!



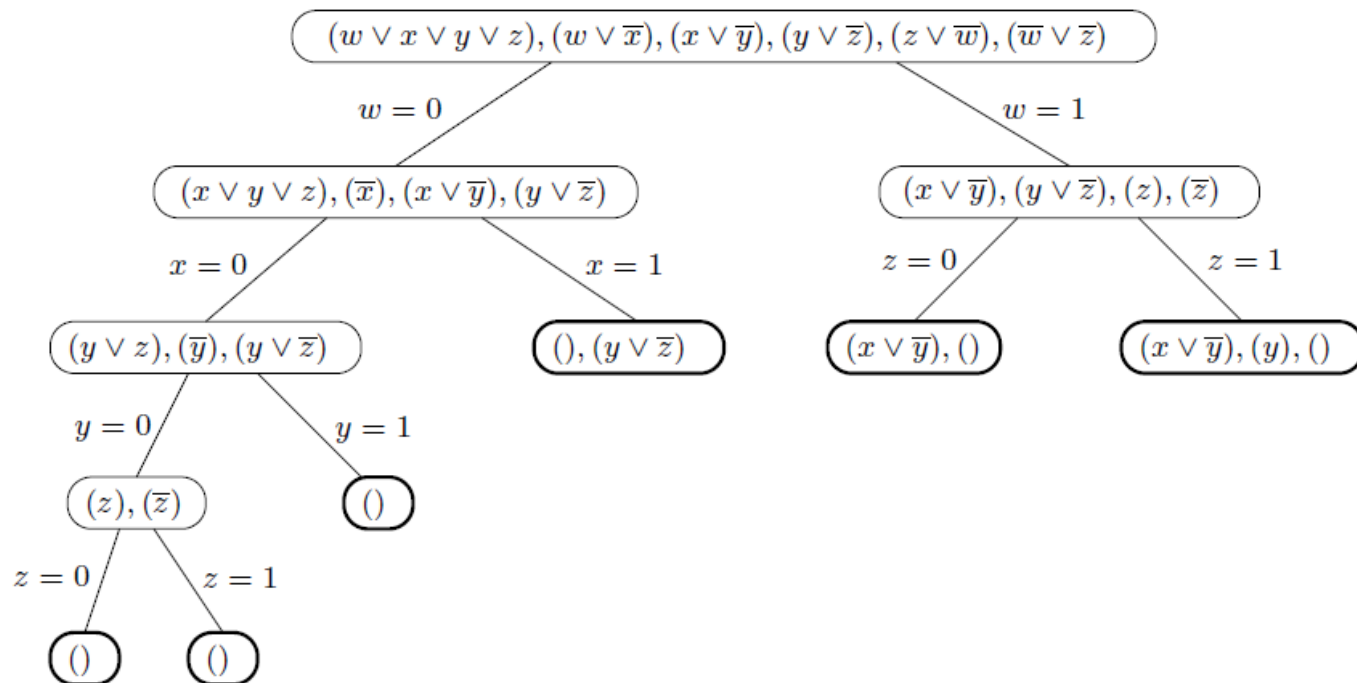
Complexitat

- Exemple: backtracking
 - Continuem **ramificant**
 - Ara podem **podar** per que no satisfem una clàusula:
$$w = 0, x = 1 \quad (w \vee \bar{x})$$
 - I retornem (**backtracking**) als altres 2 nodes actius



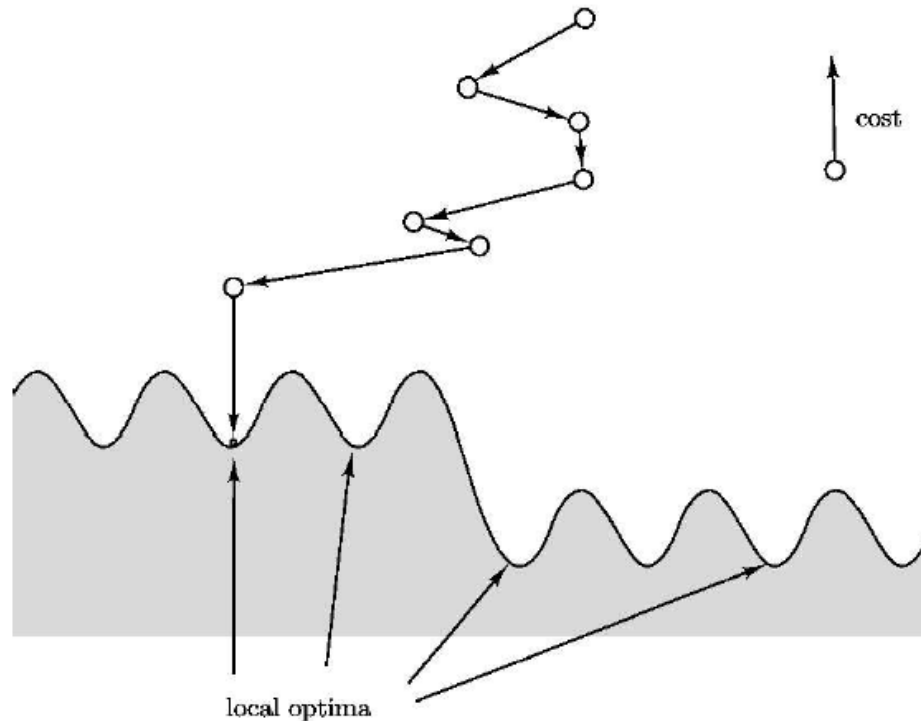
Complexitat

- Exemple: backtracking
 - Hem reduït 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

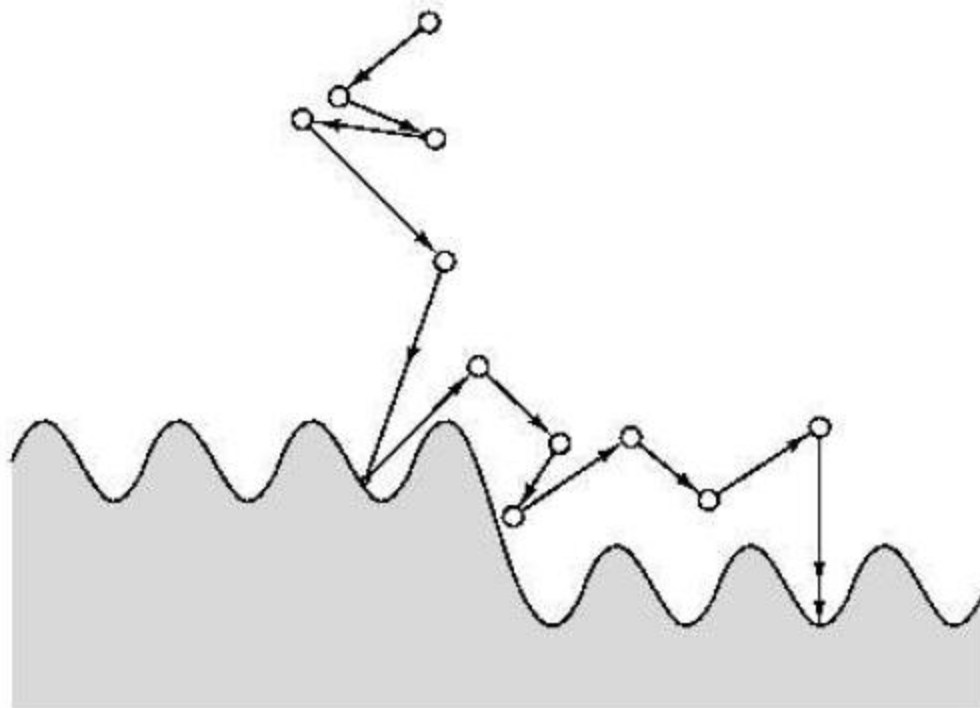
Complexitat

- Example: 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  $\Delta = \text{cost}(s') - \text{cost}(s)$  is negative:
    replace  $s$  by  $s'$ 
```

Complexitat

- Example: simulating annealing



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

Programació dinàmica

- Exercici

```

for  $i = 0, 1, 2, \dots, m$ :
     $E(i, 0) = 2i$ 
for  $j = 1, 2, \dots, n$ :
     $E(0, j) = j$ 
for  $i = 1, 2, \dots, m$ :
    for  $j = 1, 2, \dots, n$ :
         $E(i, j) = \min\{E(i-1, j) + 2, E(i, j-1) + 1, E(i-1, j-1) + \text{diff}(i, j)\}$ 
return  $E(m, n)$ 

```

Associar la paraula ALGORISME amb la paraula AVANÇADA fent ús d'aquesta inicialització i funció de programació dinàmica
 → Dibuixa l'associació de lletres final

[illegible]