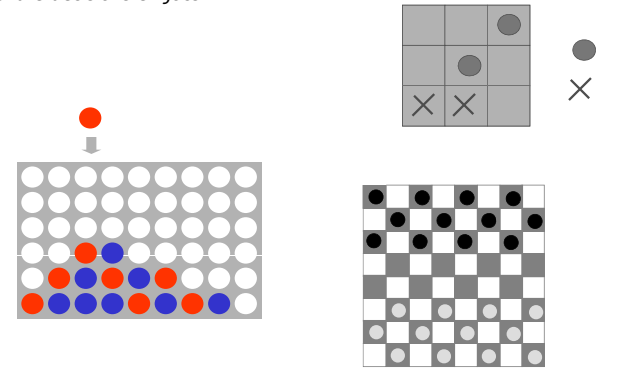


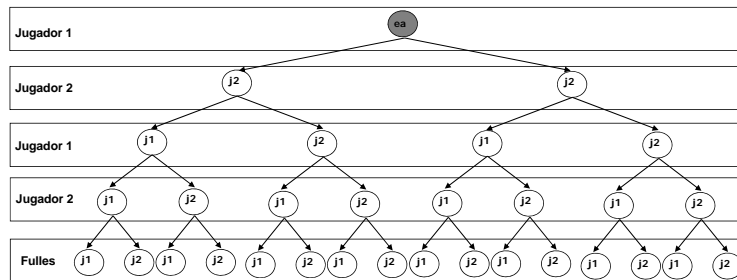
CERCA AMB ADVERSARIS  
Resolució de problemes de presa de  
decisió per exploració d'alternatives

### Exemples de Problemes

Prendre decisions en jocs



### Arbre de joc:



A les fulles podem avaluar amb 1 o 0:

- 1: Guanya el jugador 1 (Perd jugador 1)
- 0: Perd el jugador 1 (Guanya jugador 2)

**Problema:** L'arbre de joc té un creixement exponencial que el fa no assumible en molts de casos.

### Exemple: Escacs

Factor de ramificació mitjà  $b=35$

Nombre de moviments que fa un jugador en una partida = 50

Mida de l'arbre de joc  $35^{100} = 2,55 \cdot 10^{154}$

### Solució:

1. Fixar la profunditat de l'arbre a una profunditat predefinida, d'aquesta manera es pot controlar la mida de l'arbre.
2. Introduir una funció heurística que avaluï l'estat del joc a les fulles.

**Arbre de joc:** és una representació de coneixement que és un arbre semàntic amb les següents particularitats:

**Lèxic:** nivells maximitzadors (max) i nivells minimitzadors (min)

**Estructural:**

- El nivells de l'arbre tenen una etiqueta min o max associada.
- Els nivells min i max s'alternen des de l'arrel fins a les fulles.
- El nivell de les fulles no té cap etiqueta.

**Semàntica:** Node= Estat del joc  
Branca= Moviment d'una peça

**Procedimental:**

- Proc. Minimax.
- Proc. Alfa-beta.
- Proc. que calculen el mínim i el màxim d'una llista d'estats amb valors associats.

### Assumpcions del Minimax

1. Juguen alternativament un i altre
2. Hi ha una profunditat màxima prefixada
3. Hi ha una funció heurística que avalua l'estat del joc de manera simètrica per un i altre jugador.

$$f : \{Estat del joc\} \rightarrow [v_m + \delta, v_m - \delta]$$

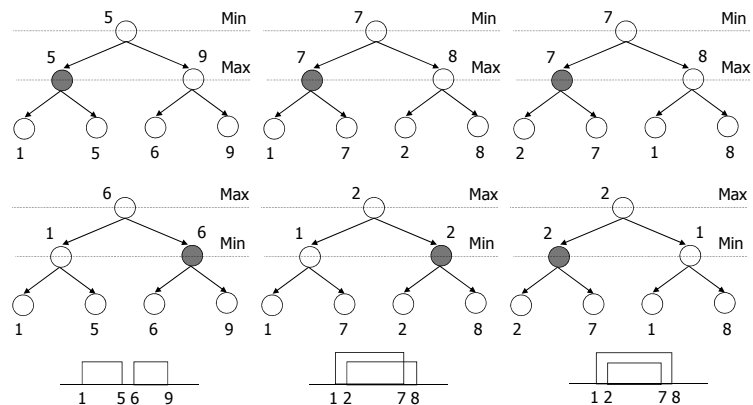
4. Ambdós jugadors trien les millors jugades per ells.

#### Passos generals de l'algorisme:

- Construir l'arbre de joc des de l'estat actual.
- Avaluar les fulles amb la funció heurística.
- Retornar els valors d'abaix cap adalt d'acord amb la funció de max o min de cada nivell.

### Comportament del Minimax

Exemples:



### Funció Minimax (EstatActual, jugador, profmax)

1. Si (profmax=0) llavors
  - a) Retornar ( [h(EstatActual) EstatActual] );
2. Sinó
  - a) E=Expandir(EstatActual); L=nil;
  - b) Per a cada (element, Ef, de E) fer  
V= Minimax(Ef, inv(jugador), profmax--);  
L= Insertar( [Cap(V), Ef], L);
  - c) FPer
  - d) Si (jugador=min) llavors Retornar(Minim(L));
  - e) Sinó Retornar(Maxim(L) );
  - f) Fsi
3. Fsi.

#### FFunció

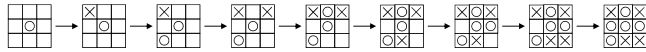
Inv(J): Retorna el jugador oposat a J.

Maxim(L): Retorna la subllista de L que conté com a primer element el valor màxim de tota la llista.

Minim(L): Retorna la subllista de L que conté com a primer element el valor mínim de tota la llista.

h(E), Expandir(E): Són funcions específiques per a cada joc.

**Exercici:** Suposem que volem construir un agent que jugui al tic-tac-toe, en el que l'objectiu és aconseguir tres peces formant una línia, vertical, horitzontal o diagonal, per exemple:



Construeix l'arbre de joc que construirà l'agent si aplica el minimax a profunditat màxima 2 a partir de la jugada:



Suposarem que: ○ és maximitzador i × és minimitzador, i que la funció heurística és la següent:

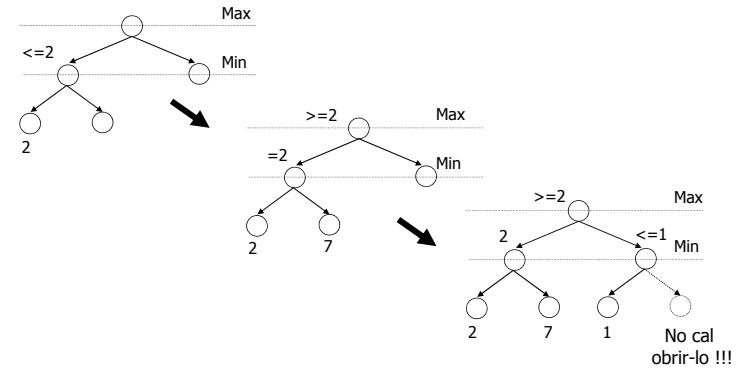
$$H(E) = h(\text{max}, E) - h(\text{min}, E),$$

on  $h(X, E) = \text{Nombre de línies que encara pot fer el jugador X sobre l'estat E}.$

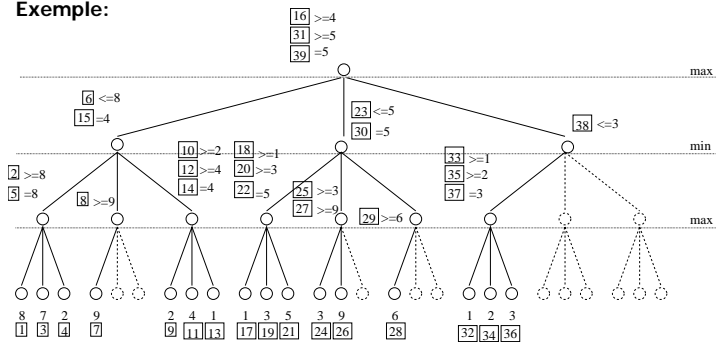
### Millora del minimax, PODA ALFA-BETA

**Idea:** No obrir camins que no poden afectar al resultat final del minimax.

**Exemple:** en recorre l'arbre de joc en profunditat tenim que:



**Exemple:**



□ → El número en el requadre representa l'ordre en que es va recorrent l'arbre

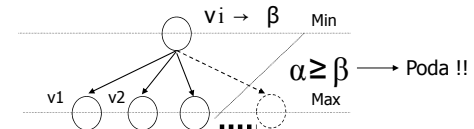
○ → Els nodes puntejats són els que no cal obrir perquè no afecten al resultat final

Exemple extret de: P.H. Winston, *Artificial Intelligence*, 3rd. Edition. Addison Wesley, MA 1992

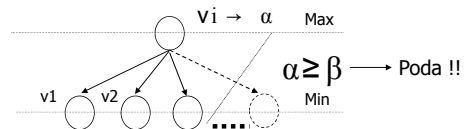
### Implementació de la poda alfa-beta:

$[\alpha, \beta]$ : representen les cotes inferior i superior fins al moment

En nivells minimitzadors:  $\beta$ : cota superior

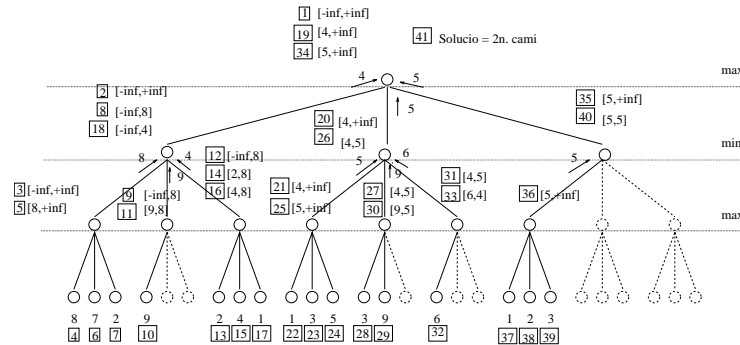


En nivells maximitzadors:  $\alpha$ : cota inferior



Inicialització:  $[\alpha, \beta] = (-\infty, +\infty)$

**Exemple:**



(Versió del mateix exemple amb intervals alfa-beta per representar les cotes de cada nivell)

**Funcio** AlfaBeta (EstatActual, alfa, beta, jugador, profmax, h)

1. **Sí** (profmax=0) **llavors**
  - a) **Retornar**( [ h(EstatActual), EstatActual ] );
2. **Sinó**
  - a) **E=Expandir**(EstatActual);
  - b) **Sí** (jugador=min) **llavors**;
    - i. **Fins que** (E buida) **o bé** (alfa >= beta) **fer**  
 $V = \text{AlfaBeta}(\text{Cap}(E), \text{alfa}, \text{beta}, \text{inv}(\text{jugador}), \text{profmax}--, h);$   
**Sí** (Cap(V) < beta) **llavors** beta = Cap(V); Ev = Cap(E); **fsi**  
 E = Cua(E);
    - ii. **FFinsque**
    - iii. **Retornar**( [beta, Ev] );
  - c) **Sinó** /\* jugador = max \*/
    - i. **Fins que** (E buida) **o bé** (alfa >= beta) **fer**  
 $V = \text{AlfaBeta}(\text{Cap}(E), \text{alfa}, \text{beta}, \text{inv}(\text{jugador}), \text{profmax}--, h);$   
**Sí** (Cap(V) > alfa) **llavors** alfa = Cap(V); Ev = Cap(E); **fsi**  
 E = Cua(E);
    - ii. **FFinsque**
    - iii. **Retornar**( [alfa, Ev] );
  - d) **Fsi**
3. **Fsi**

**FFuncio**

**Anàlisi dels algorismes:**

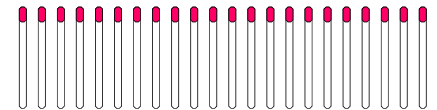
Criteri	Minimax	Poda Alfa-Beta
Temps	$b^p$	Pitjor cas: $b^p$ Millor cas: $2b^{\frac{p}{2}} - 1$ si p parell $b^{\frac{p+1}{2}} + b^{\frac{p-1}{2}} - 1$ si p senar
Espai	$b \cdot p$	$b \cdot p$
Complet		Sí (si no hi ha limitació de temps)
Òptim		No té sentit parlar-ne (Per exemple: Joc dels Ilumins)

**b:** factor de ramificació  
**p:** profunditat màxima

**El joc dels Ilumins i l'optimalitat**

**Exercici:** Suposem que ens plantegen fer un agent que jugui al següent joc contra un adversari:

"Considereu que sobre la taula hi ha 23 llumins i alternativament cada jugador pot agafar 1, 2 o 3 llumins. Perd el jugador que agafa el darrer llumí"



Digueu

1. Quin algorisme de cerca hauria de fer servir aquest agent?
2. Com serà l'arbre de cerca?, semàntica de nodes, branques, factor de ramificació ....
3. Quina funció heurística podria fer servir? ...

*Solució òptima: Deixar sobre la taula un número de llumins, n, tal que compleixi que  $n = 4 * x + 1$*

### Minimax amb limitació de temps: APROFUNDIMENT PROGRESSIU

**Idea:** Aplicar el minimax (poda alfa-beta) a totes les profunditats que es pugui des de una profunditat mínima que garanteixi tenir suficient temps i en endavant, s'acabarà quan s'acabi el temps, i es retornarà la jugada seleccionada amb l'arbre de major profunditat.

#### Algorisme:

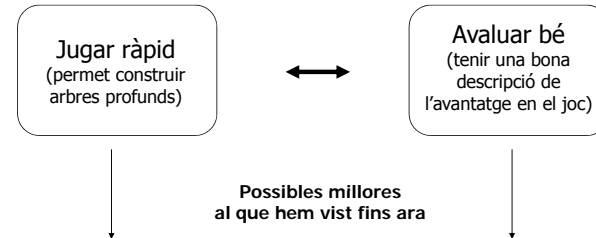
1. Inicialització  $P=P_{min}$ ;
2. Mentre (hi hagi temps) fer
  - (a) Apliquem PodaAlfa-Beta a profunditat  $P$ ;
  - (b) Guardem la solució del pas anterior a  $R$ ;
  - (c)  $P++$ ;
3. FMentre
4. Retornar( $R$ );

#### Complexitat addicional:

$$Cost\ Minimax\ (p=d) \approx (b-1) * Cost\ Aprof.\ Prog.(p= d-1)$$

**Qüestió:** Normalment no existeix optimalitat en jocs, ara bé, qué cal per a que un programa jugui bé ?

**Resposta:** normalment és el resultat d'aconseguir el màxim en dues coses:



- Representació eficient del taulell
- Guardar taules de transposicions

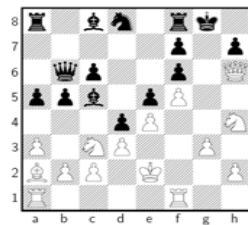
- Bona funció d'avaluació
- Control de l'efecte horitzó

### Representació del taulell

1. Matriu de sencers de 8x8

Exemple:

```
enum {RB, DB, TB, AB, CB, PB, RN, DN, TN, AN, CN, PN, VA};
int casillas[64];
```



TN	VA	AN	CN	VA	TN	RN	VA
VA	VA	VA	VA	VA	PN	VA	PN
VA	DN	PN	VA	VA	PN	VA	DB
PN	PN	AN	VA	PN	PB	VA	VA
VA	VA	VA	PN	PB	VA	VA	CB
PB	VA	CB	PB	VA	VA	PB	VA
AB	PB	PB	VA	RB	VA	VA	PB
TB	VA	VA	VA	VA	TB	VA	VA

### Representació del taulell

2. Mapes de bits per tipus de peces

Exemple:



Peons negres  
00000000  
0000101  
00100100  
11001000  
00010000  
00000000  
00000000  
00000000

Peons blancs  
00000000  
00000000  
00000000  
00000100  
00001000  
10010010  
01100001  
00000000

Cavalls blancs  
00000000  
00000000  
00000000  
00000000  
00000000  
00100000  
00000000  
00000000  
00000000

Cavalls negres  
00010000  
00000000  
00000000  
00000000  
00000000  
00000000  
00000000  
00000000  
00000000

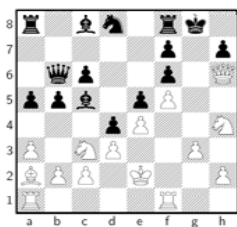
Alfils blancs  
00000000  
00000000  
00000000  
00000000  
00000000  
00000000  
10000000  
00000000

... fins a 12 mapes de bits

### Representació del taulell

3. Representacions híbrides (formades pels dos tipus anteriors).

Exemple:



Mapes de bits

Peces Negres

```
10110110
00000101
01100100
11101000
00010000
00000000
00000000
00000000
```

Peces blanques

```
00000000
00000000
00000001
00000100
00001001
10110010
11101001
10000100
```

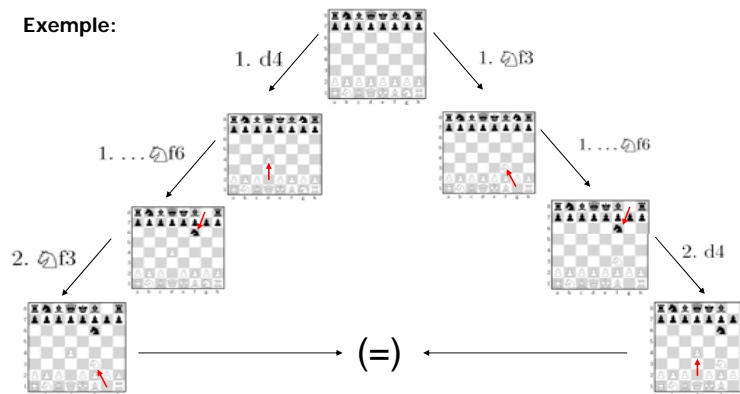
Matriu de sencers 8x8

TN	VA	AN	CN	VA	TN	RN	VA
VA	VA	VA	VA	VA	PN	VA	PN
VA	DN	PN	VA	VA	PN	VA	DB
PN	PN	AN	VA	PN	PB	VA	VA
VA	VA	VA	PN	PB	VA	VA	CB
PB	VA	CB	PB	VA	VA	PB	VA
AB	PB	PB	VA	RB	VA	VA	PB
TB	VA	VA	VA	TB	VA	VA	VA

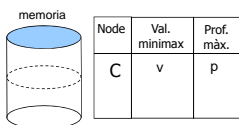
### Taules de transposicions

**Transposició:** Seqüències de moviments diferents que porten a una mateixa situació.

Exemple:

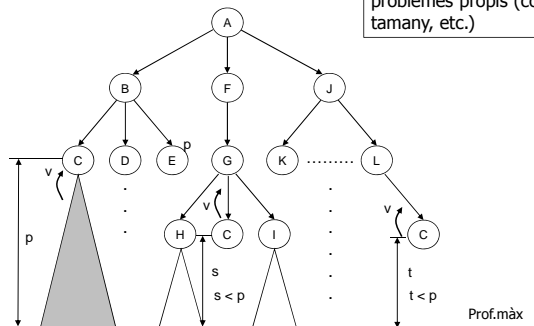


### Taules de transposicions



**Implementació,** implica la gestió d'una taula que guarda el valor resultat d'aplicar el minimax a una certa profunditat màxima.

Normalment és una taula hash amb tots els problemes propis (col.lisions, control del tamany, etc.)



### Funció d'avaluació

Depèn totalment del joc, normalment és un compromís entre diferents termes:

Exemple:

$$f(\text{estat}) = \sum_i w_i \times T_i(\text{estat})$$

ESCACS

**T<sub>1</sub>: Material**  
 Valors habituals:

Dama	9.50
Torre	5.00
Alfil	3.75
Cavall	3.50
Peó	1.00

- T<sub>2</sub>: Activitat**
- T<sub>3</sub>: Estructura dels peons**
- T<sub>4</sub>: Seguretat del rei**

### Funció d'avaluació

#### Exemple:

**DAMES** el 1963 A. Samuel defineix una funció d'avaluació amb capacitat d'aprenentatge que es convertirà en campió mundial

La funció d'avaluació estava formada per 26 termes diferents:

$$f(\text{estat}) = \sum_{i=1}^{26} w_i \times T_i(\text{estat})$$

$T_1$ : Terme que avalua l'avantatge de peces.

$T_2$ : Terme que avalua la mobilitat.

$T_3$ : Terme que avalua el control del centre.

$T_4$ : Terme que avalua la capacitat d'avançament.

$T_4$ : Terme que avalua les amenaces dobles.

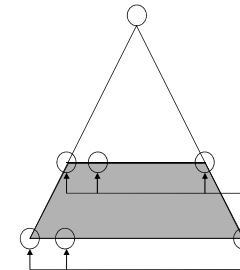
fins a  $T_{26}$

els paràmetres  $w_i$  eren calculats amb tècniques d'aprenentatge, per generalització a partir de fer jugar l'ordinador contra sí mateix.

### Control de l'efecte horitzó

L'efecte horitzó és produït pel fet d'aplicar el minimax a una profunditat fixada, i basat tot l'anàlisi amb el valor de la funció heurística de les fulles en aquesta profunditat.

#### Arbre de joc

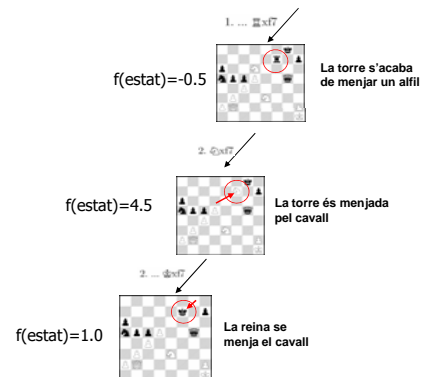


A **Profunditat no màxima** desconeixem totalment què pot passar en els moviments posteriors i la funció d'avaluació pot no ser capaç de considerar girs importants amb l'avantatge entre jugadors

**Exemple:** Intercanvis de peces en escacs, un jugador mata una peça a l'adversari i aleshores en el següent moviment l'adversari n'hi mata una a ell, els valors de la funció heurística a les fulles poden tenir fluctuacions importants i poc representatives de l'estat del joc en aquell punt.

A **Profunditat màxima** guanya un o l'altre jugador, l'avaluació és molt clara

**Exemple:** Fluctuacions de la funció heurística en funció de la profunditat a la que s'avalua l'estat del joc.



**Solució:**  
Heurística d'extensió singular  
(*Quiescence Search*)

Continuar aprofundint només en aquelles fulles que presenten situacions forçades (avaluacions molt diferents a les dels veïns) fins que la situació s'estabilitzi, i aleshores avaluar aplicant el minimax des d'aquell punt en amunt.