UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Laurea triennale in Informatica

Fondamenti di Intelligenza Artificiale

Lezione 10 - Ricerca con avversari

Ricerca con Avversari

Ambienti Multi-Agente

Nella lezione sugli agenti intelligenti, abbiamo introdotto le proprietà degli ambienti e, in particolar modo, gli scenari in cui più agenti agiscono nello stesso ambiente. Questi sono noti come **ambienti multi-agente**.

In questo contesto, ogni agente deve considerare le azioni degli altri, così come gli effetti di tali azioni, per poter efficacemente quantificare il proprio benessere. Questo può dar luogo a *contingenze* nel processo di risoluzione di un problema.

Oggi iniziamo a discutere di **ambienti competitivi**, in cui gli obiettivi degli agenti sono in conflitto. Questo origina i problemi di ricerca con avversari, i quali vengono tipicamente indicati come *giochi*.

> La **teoria dei giochi** è una *branca dell'economia* che considera ogni ambiente multi-agente come un gioco, indipendentemente dal fatto che l'interazione sia cooperativa o competitiva, a patto che l'influenza di ogni agente sugli altri sia significativa.

Curiosamente, gli ambienti con quantità molto grande di agenti sono spesso considerate **economie** piuttosto che giochi.

Esistono diverse tipologie di giochi (o economie). Queste dipendono dal grado di osservabilità dell'ambiente, dal grado di determinismo che possiamo supporre, o altro.

Ricerca con Avversari

Teoria dei Giochi - Classificazione

Come detto, esistono diverse tipologie di giochi. Questi sono classificabili in vario modo, ma tipicamente sono due gli aspetti principali:

Classificazione 1: *Condizioni di Scelta*

Giochi con informazione "perfetta". Giochi in cui gli stati del gioco sono completamente espliciti agli agenti.

Giochi con informazione "imperfetta". Giochi in cui gli stati del gioco sono solo parzialmente esplicitati.

Classificazione 2: *Effetti della Scelta*

Giochi deterministici. Giochi in cui gli stati del gioco sono determinati unicamente dalle azioni degli agenti.

Giochi stocastici. Giochi in cui gli stati del gioco sono determinati anche da fattori esterni (ad esempio, tutti i giochi in cui c'è un dado).

Altre classificazioni tengono conto della composizione degli ambienti, ad esempio se questi sono discreti, statici, episodici, ed altro.

Ricerca con Avversari

Teoria dei Giochi: Ambienti

I giochi più comuni, e quelli che privilegeremo in questo corso, sono quelli che vengono definiti a **somma zero** e con **informazione perfetta**.

Questi sono i giochi in cui due giocatori, a turno, effettuano azioni che influenzano l'ambiente così come il comportamento dell'altro giocatore.

Nella nostra terminologia, parliamo di ambienti multi-agente, deterministici e completamente osservabili, in cui due agenti agiscono alternandosi e in cui i valori di utilità, alla fine della partita, **sono sempre uguali ma di segno opposto**.

Ad esempio, consideriamo il caso degli scacchi. Se un giocatore vince, l'altro deve necessariamente perdere. Questa opposizione ci fa parlare di *"avversari"*.

I giochi hanno da sempre catturato l'interesse degli studiosi, anche e soprattutto perché, a differenza dei problemi giocattolo che abbiamo visto (ad esempio, le N-regine) sono *facilmente rappresentabili ma estremamente difficili da risolvere*.

Un tipico esempio è quello degli scacchi. In questo caso, il fattore di ramificazione è di circa 35, con partite che spesso arrivano a 50 mosse a giocatore. L'albero di ricerca avrà quindi **circa 35^{100} nodi** —> questa è una delle ragioni per cui gli scacchi hanno interessato da sempre il mondo dell'intelligenza artificiale.

Come vedremo, i giochi richiedono l'abilità di prendere una decisione quando il calcolo di una soluzione ottima non è realizzabile o comunque ne penalizzerebbe l'efficienza.

Ricerca con Avversari

Teoria dei Giochi: Definizione

Più formalmente, possiamo definire un gioco come un problema di ricerca con i seguenti componenti:

s_0: lo **stato iniziale**, che specifico come è configurato il gioco in partenza;

GIOCATORE(*s*): definisce il **giocatore** a cui tocca fare una mossa nello stato *s*;

AZIONI(*s*): restituisce l'insieme delle **mosse lecite** in uno stato *s*;

RISULTATO(*s, a*): il **modello di transizione**, che definisce il risultato di una mossa;

TEST-TERMINAZIONE(*s*): un **test di terminazione**, che restituisce 'vero' se la partita è finita e 'falso' altrimenti. Gli stati che fanno terminare una partita sono detti *"stati terminali"*.

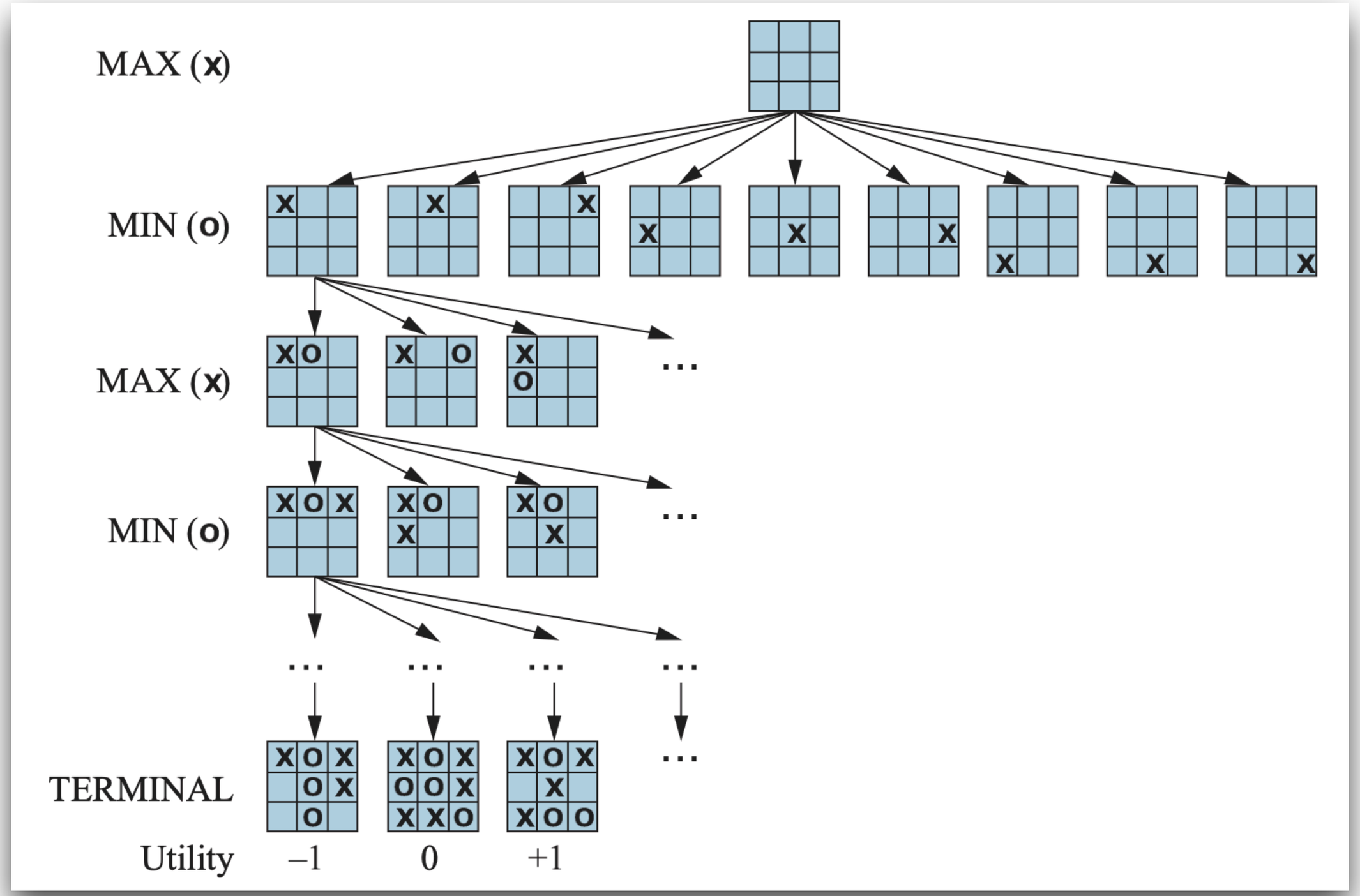
UTILITÀ(*s, p*): una **funzione utilità**, chiamata anche **funzione obiettivo** o **funzione di payoff**, che definisce il valore numerico finale per un gioco che termina nello stato terminale s per un giocatore *p*. In altri termini, la funzione da il punteggio finale da assegnare ai giocatori.

Negli scacchi, i possibili risultati sono 'vittoria', 'sconfitta' o 'pareggio', ai quali vengono assegnati i corrispondenti valori di +1, 0 o 1/2.

Lo stato iniziale, la funzione AZIONI e la funzione RISULTATO definiscono **l'albero di gioco**, un albero in cui i nodi sono stati del gioco e gli archi le mosse.

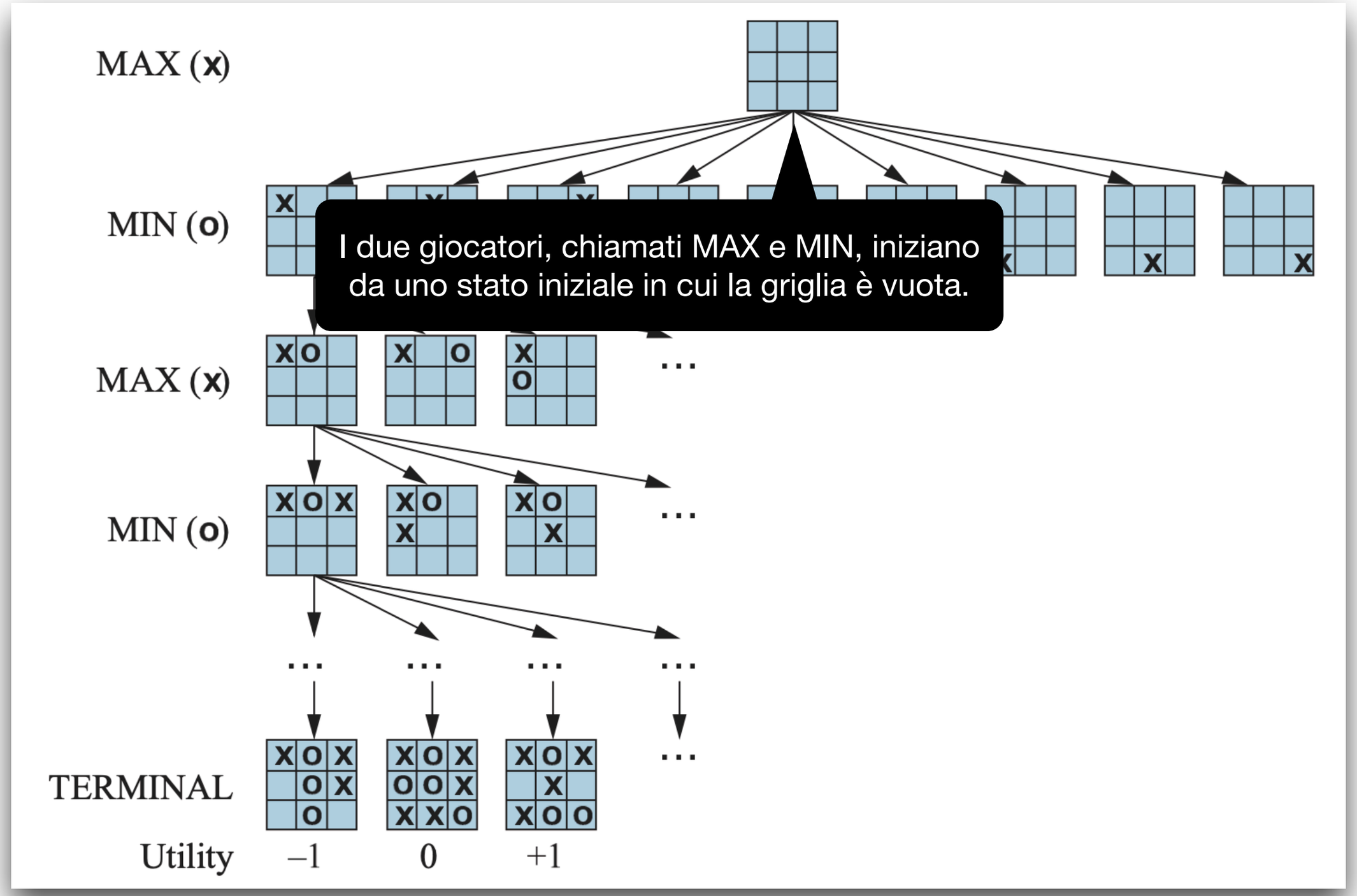
Ricerca con Avversari

Teoria dei Giochi: L'esempio del Tris



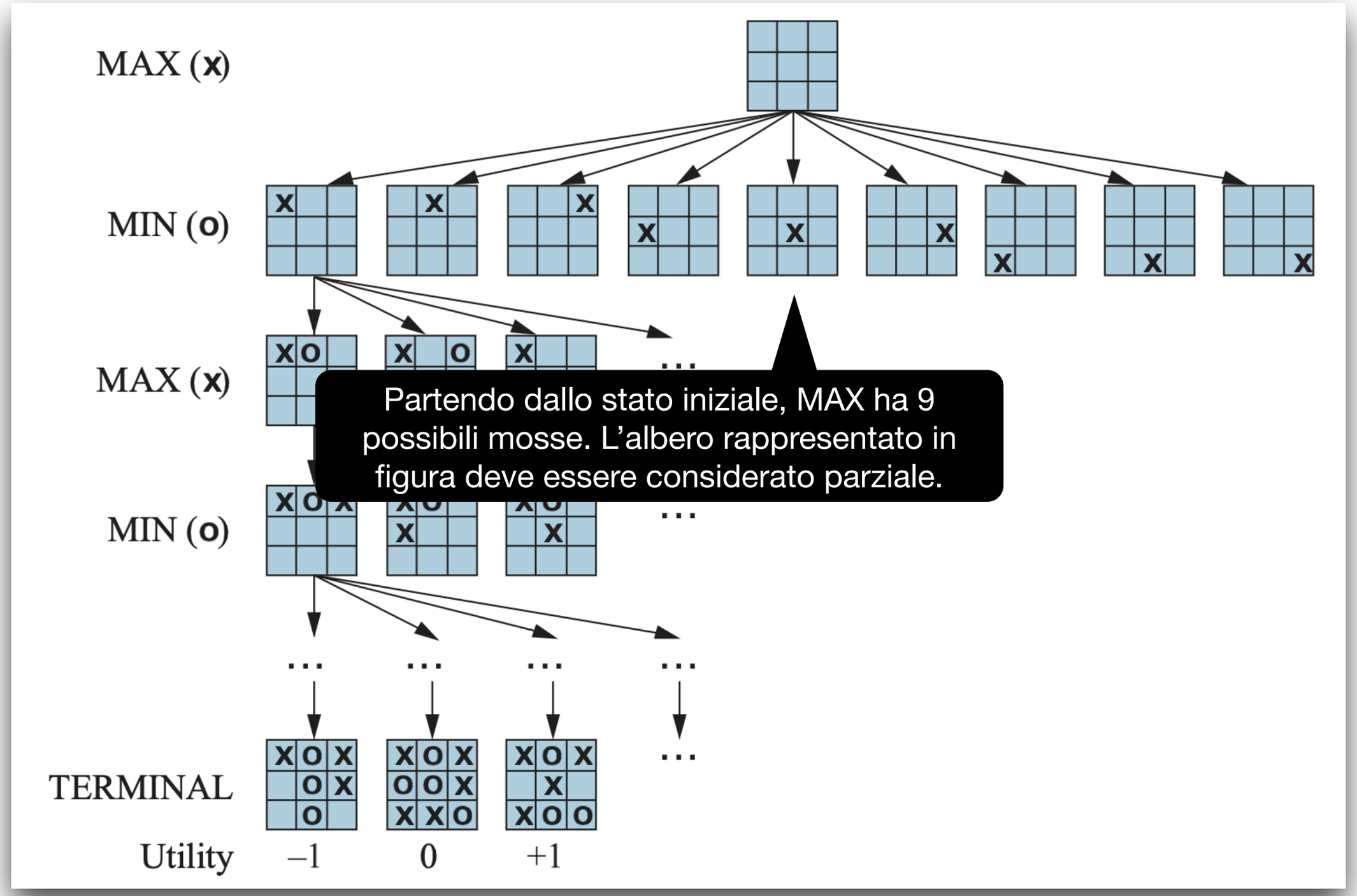
Ricerca con Avversari

Teoria dei Giochi: L'esempio del Tris



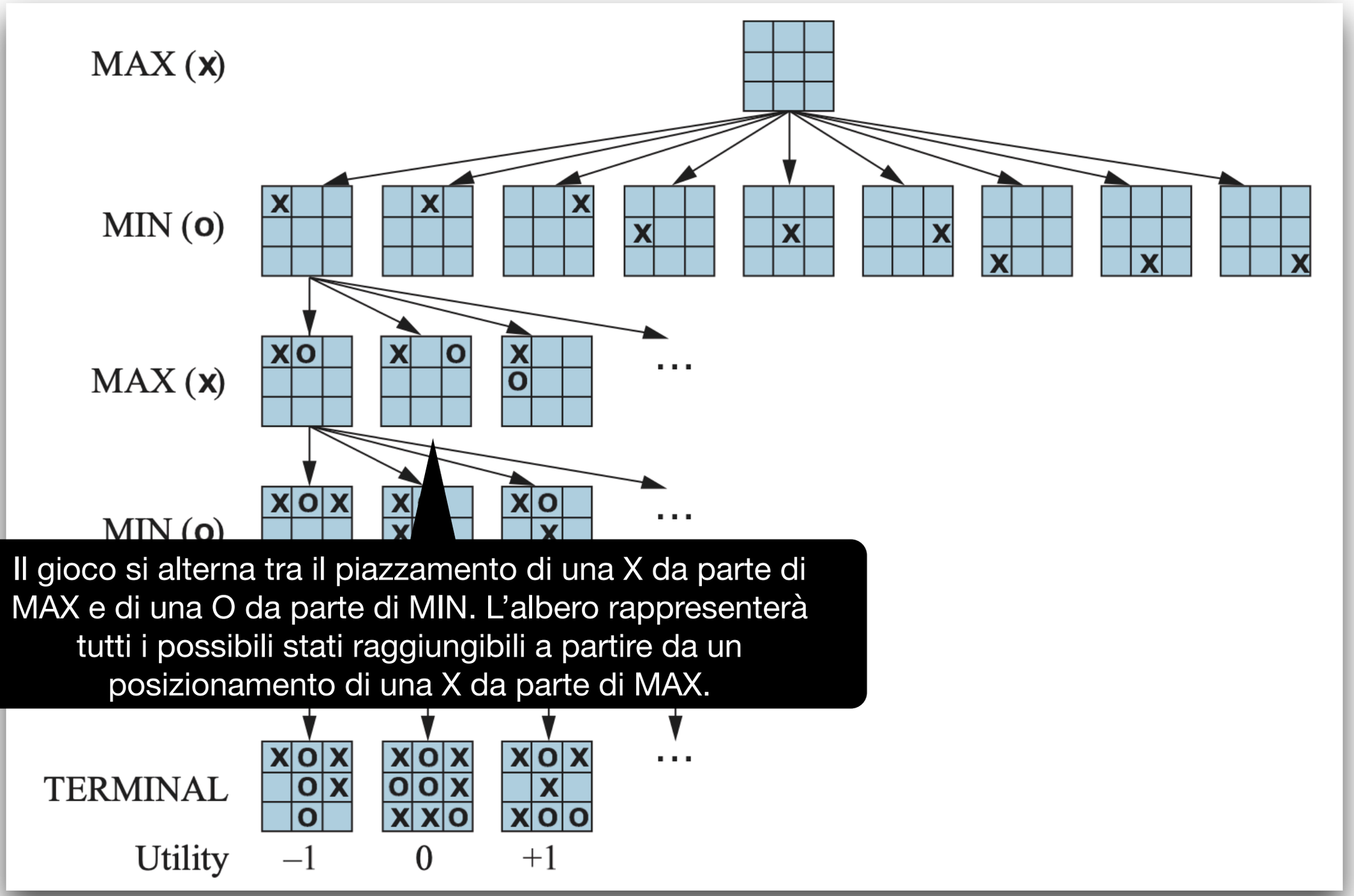
Ricerca con Avversari

Teoria dei Giochi: L'esempio del Tris



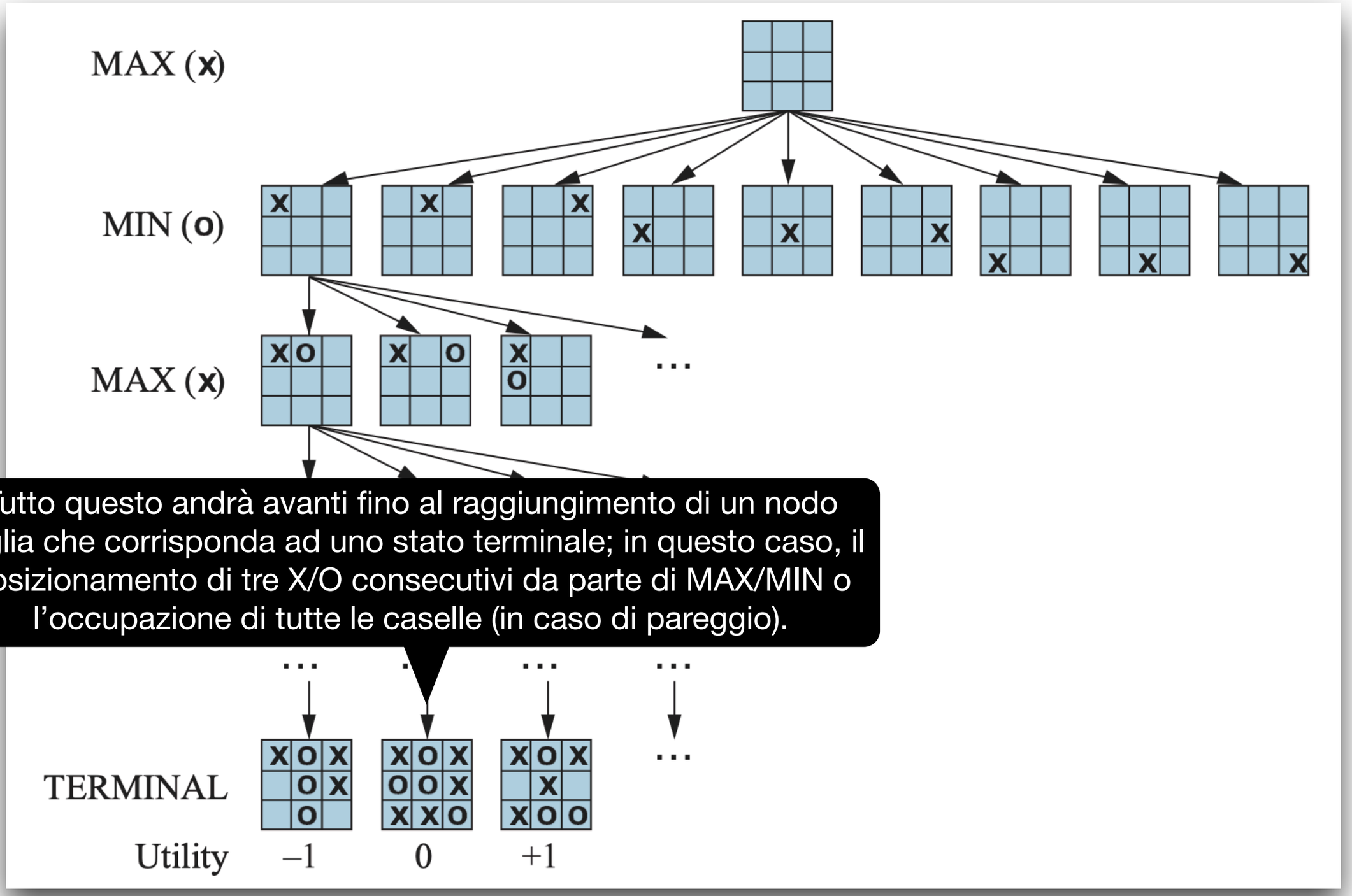
Ricerca con Avversari

Teoria dei Giochi: L'esempio del Tris



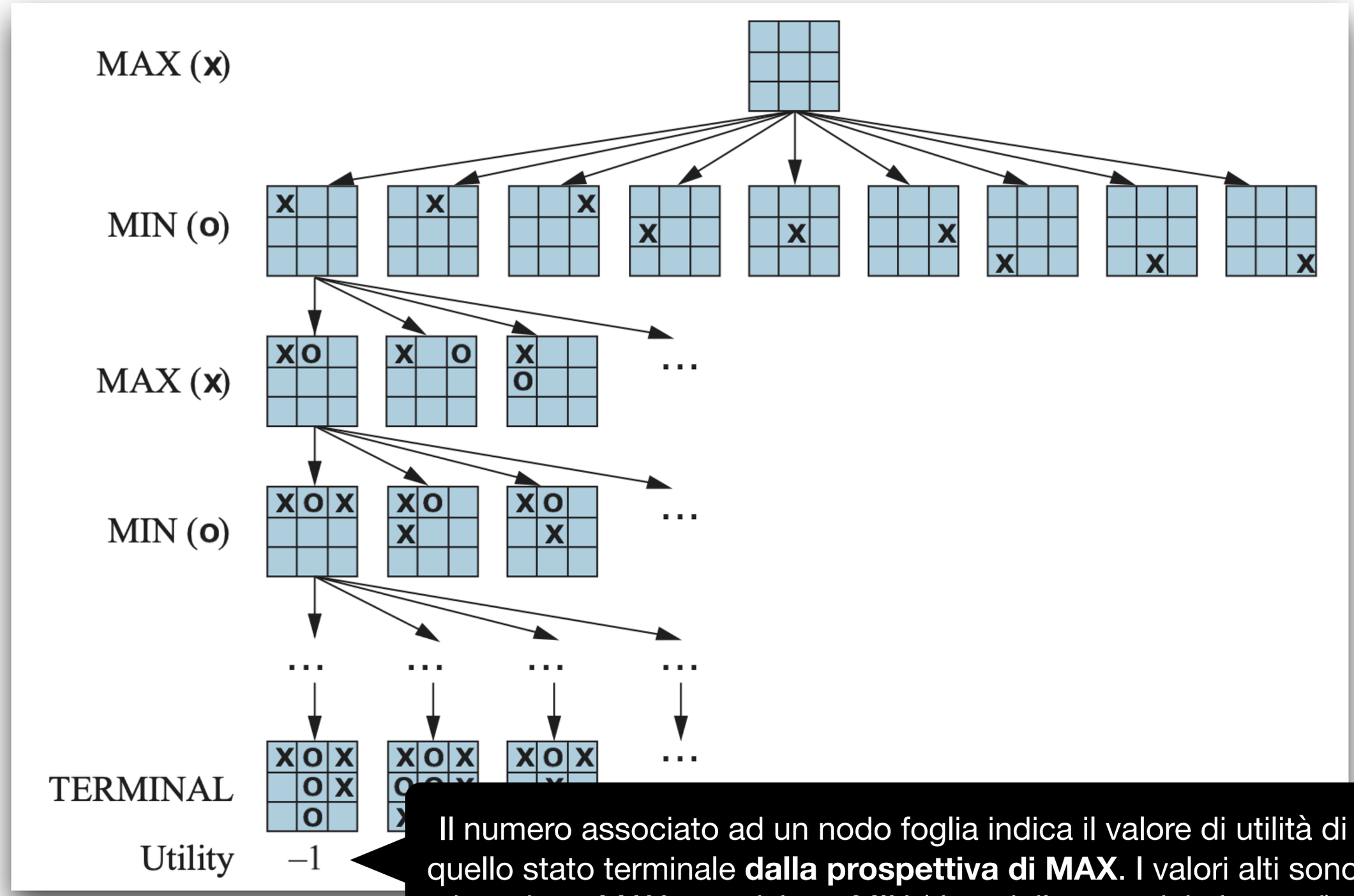
Ricerca con Avversari

Teoria dei Giochi: L'esempio del Tris



Ricerca con Avversari

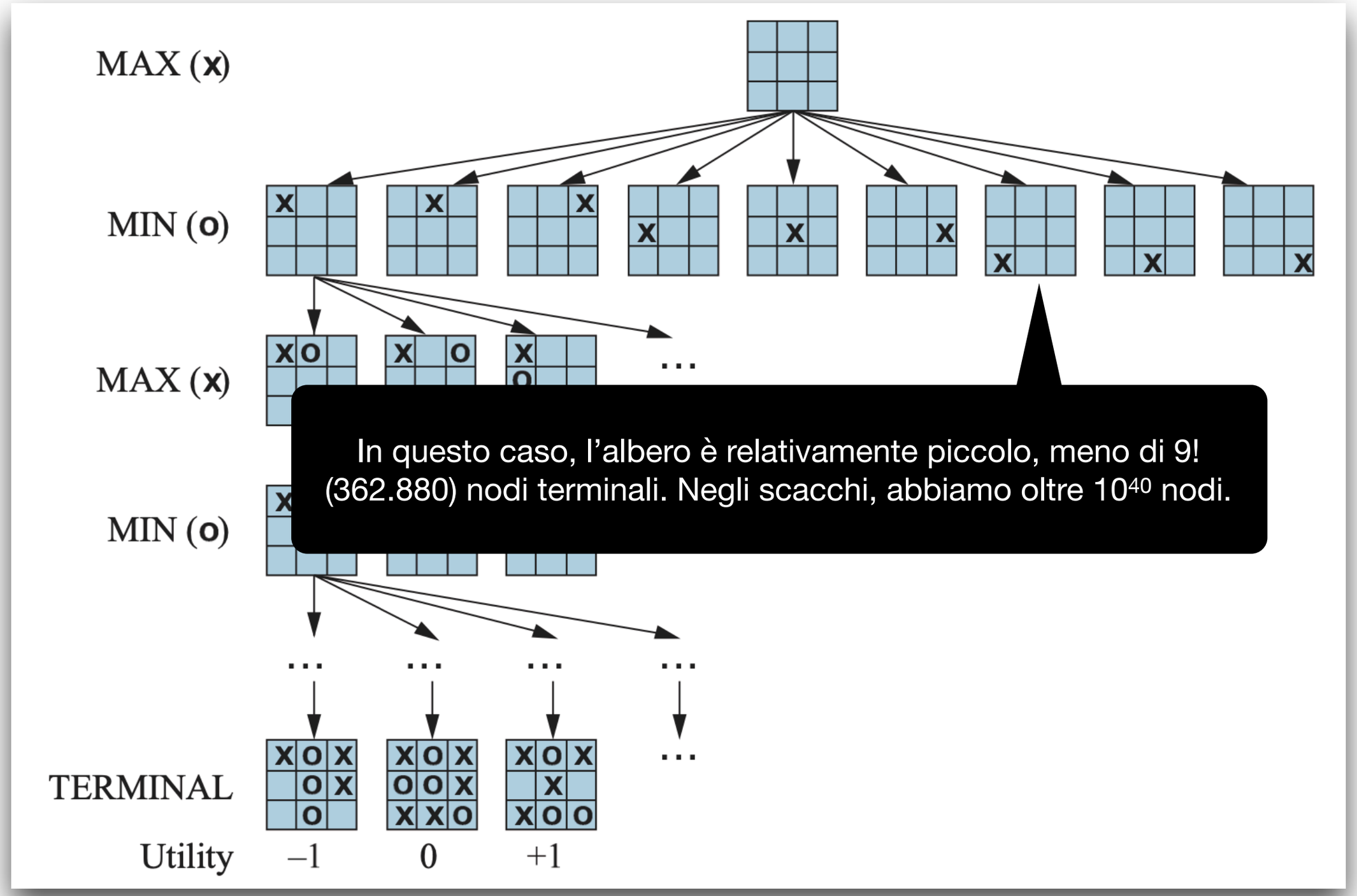
Teoria dei Giochi: L'esempio del Tris



Il numero associato ad un nodo foglia indica il valore di utilità di quello stato terminale **dalla prospettiva di MAX**. I valori alti sono buoni per MAX e cattivi per MIN (da qui, il nome dei giocatori)

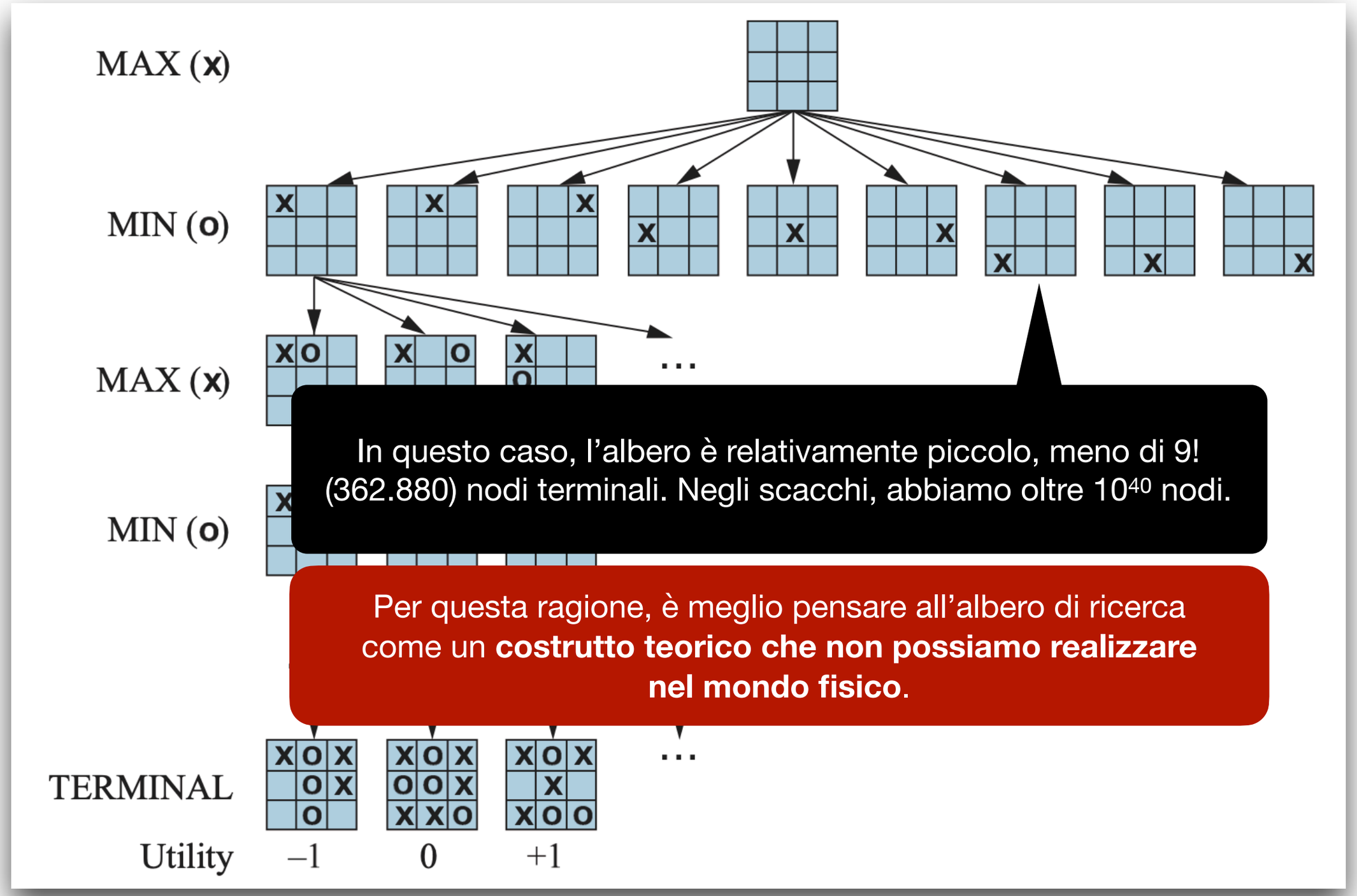
Ricerca con Avversari

Teoria dei Giochi: L'esempio del Tris



Ricerca con Avversari

Teoria dei Giochi: L'esempio del Tris



Ricerca con Avversari

Decisioni Ottime

In un normale problema di ricerca, la soluzione ottima è costituita da una sequenza di mosse che portano ad uno stato obiettivo.

In una ricerca con avversari, MIN può *“dire la sua”*. MAX dovrà quindi trovare una **strategia** contingente che specifichi la sua mossa nello stato iniziale, così come le mosse in tutti gli stati possibili risultanti dalla prima mossa di MIN, e così via.

Una **strategia ottima** porta ad un risultato che è almeno pari a quello di qualsiasi altra strategia, *assumendo che si stia giocando contro un giocatore infallibile*.

A proposito di giochi, strategie e decisioni ottime, vale la pena discutere del celeberrimo *dilemma del prigioniero* - reso ai più noto da *John Forbes Nash*, matematico americano insignito del premio Nobel per l’Economia nel 1994 e uno dei padri della teoria dei giochi, insieme a - tra gli altri - John Von Neumann.

Dilemma del Prigioniero. Due criminali vengono accusati di aver commesso di porto abusivo di armi e, per questo, gli investigatori li arrestano e li chiudono in due celle diverse, *impedendo loro di comunicare*. Ad ognuno di loro vengono date due scelte: collaborare, oppure non collaborare.

Gli investigatori spiegano loro le varie opportunità che hanno a disposizione. Queste possono essere riassunte come segue.

Ricerca con Avversari

Il Dilemma del Prigioniero

Dilemma del Prigioniero. Due criminali vengono accusati di aver commesso di porto abusivo di armi e, per questo, gli investigatori li arrestano e li chiudono in due celle diverse, *impedendo loro di comunicare*. Ad ognuno di loro vengono date due scelte: collaborare, oppure non collaborare.

Ai due criminali viene spiegato che:

(1) Se solo uno dei due collabora accusando l'altro, chi ha collaborato **evita la pena**; l'altro viene però condannato a **7** anni di carcere;

(2) Se entrambi accusano l'altro, vengono entrambi condannati a **6** anni.

(3) Se nessuno dei due collabora, entrambi vengono condannati a **1** anno, perché comunque già colpevoli di porto abusivo di armi.

Il problema può essere descritto tramite una matrice 2x2:

	Collabora	Non Collabora
Collabora	(6, 6)	(0, 7)
Non Collabora	(7, 0)	(1, 1)

Ricerca con Avversari

Il Dilemma del Prigioniero

Dilemma del Prigioniero. Due criminali vengono accusati di aver commesso di porto abusivo di armi e, per questo, gli investigatori li arrestano e li chiudono in due celle diverse, *impedendo loro di comunicare*. Ad ognuno di loro vengono date due scelte: collaborare, oppure non collaborare.

	Collabora	Non Collabora
Collabora	(6, 6)	(0, 7)
Non Collabora	(7, 0)	(1, 1)

La miglior strategia di questo gioco **non cooperativo** è (*collabora, collabora*) perché un prigioniero non sa cosa sceglierà di fare l'altro. La scelta di collaborare è, quindi, la migliore possibile per uno dei due, in assenza di informazioni su cosa farà l'altro.

Perché? Per ognuno dei due lo scopo è quello di **minimizzare** la propria condanna. Quindi, ogni prigioniero:

Collaborando	Rischia da 0 a 6 anni;
Non Collaborando	Rischia da 1 a 7 anni.

Ricerca con Avversari

Il Dilemma del Prigioniero

Dilemma del Prigioniero. Due criminali vengono accusati di aver commesso di porto abusivo di armi e, per questo, gli investigatori li arrestano e li chiudono in due celle diverse, *impedendo loro di comunicare*. Ad ognuno di loro vengono date due scelte: collaborare, oppure non collaborare.

	Collabora	Non Collabora
Collabora	(6, 6)	(0, 7)
Non Collabora	(7, 0)	(1, 1)

Supponiamo che i due *si siano promessi di non collaborare in caso di arresto.* Sono ora rinchiusi in due celle diverse e si chiederanno se la promessa sarà mantenuta dall'altro; se un prigioniero non rispetta la promessa e l'altro sì, il primo è allora liberato. C'è dunque un **dilemma**: collaborare o non collaborare. *La teoria dei giochi ci dice che c'è un solo equilibrio (collabora, collabora).*

Equilibrio di Nash. Questo equilibrio, ovvero quello *(collabora, collabora)* è chiamato equilibrio di Nash. La definizione di equilibrio di Nash ha avuto negli anni implicazioni non solo matematiche/economiche, ma persino filosofiche! Cerchiamo di spiegare:

Ricerca con Avversari

Il Dilemma del Prigioniero - Implicazioni

A Beautiful Mind: https://www.youtube.com/watch?v=StuEQi0MZsc. Questa è solo una scena di un film Premio Oscar e Golden Globe nel 2002.

Ricerca con Avversari

Il Dilemma del Prigioniero - Implicazioni

Potremmo trattare il problema con una matrice 2x2 uguale alla precedente:

	Non provarci	Provarci
Non provarci	(6, 6)	(0, 7)
Provarci	(7, 0)	(1, 1)

Ricerca con Avversari

Il Dilemma del Prigioniero - Implicazioni

Potremmo trattare il problema con

Se tutti i giocatori ci provano con la bionda (o con il biondo), ci sarà il rischio che nessuno riesca a vincere - questo minimizza le chance di successo in questo caso.

Non provarci	(6, 6)	(0, 7)
Provarci	(7, 0)	(1, 1)

Ricerca con Avversari

Il Dilemma del Prigioniero - Implicazioni

Potremmo trattare il problema con una matric

Se solo uno dei giocatori ci prova, arrecherà un danno agli altri.

	Non prov ci	Pro arci
Non provarci	(6, 6)	(0, 7)
Provarci	(7, 0)	(1, 1)

Ricerca con Avversari

Il Dilemma del Prigioniero - Implicazioni

Potremmo trattare alla precedente:

Provarci con le amiche (gli amici) porterebbe ad un ottimo “globale”, ovvero un caso in cui tutti siano (più) soddisfatti.

	Non provarci	Provarci
Non provarci	(6, 6)	(0, 7)
Provarci	(7, 0)	(1, 1)

Il Dilemma del Prigioniero - Implicazioni a più ampio spettro

La scelta migliore per la *'società'* composta dai due prigionieri sarebbe di non parlare, ma questa scelta **non è** un equilibrio di Nash: infatti uno qualsiasi dei due, se fosse il solo a parlare, potrebbe migliorare la sua situazione. Paradossalmente, l'unico equilibrio di Nash di questo esempio è la situazione peggiore per la *'società'*, quella in cui entrambi parlano prendendo 6 anni a testa.

Dal punto di vista del **singolo** prigioniero, infatti:

- Se il suo compagno non ha parlato, parlando è libero invece di essere condannato ad un anno;
- Se il suo compagno ha parlato, parlando è condannato a 6 anni invece che a 7;

A ciascuno dei due prigionieri **conviene parlare**; dal punto di vista dell'equilibrio di Nash è irrilevante il fatto che se un prigioniero parla peggiora la situazione dell'altro.

Secondo la teoria economica, detta 'classica', di *Adam Smith* (1723-1790), se ogni componente di un gruppo persegue *il proprio interesse personale* e vi sono condizioni di concorrenza perfetta, nell'equilibrio che ne esce **ogni azione individuale accresce la ricchezza complessiva del gruppo**.

La teoria dei giochi dimostra che **non è così!** Se ogni componente del gruppo fa ciò che è meglio per sé, il risultato cui si giunge è, in generale, un equilibrio di Nash che però non rappresenta la soluzione migliore per la *'società'* e può inoltre rappresentare un'allocazione inefficiente delle risorse.

Il Dilemma del Prigioniero - Implicazioni a più ampio spettro (a livello macro-economico e oltre)

In altri termini, la questione diventa persino *filosofica*: è meglio acquisire l'ottimo individuale o l'ottimo societario? Questo è, in qualche modo, alla base della distinzione tra capitalismo e socialismo… Nash ha vinto un Nobel per l'Economia per questo!

Ottimo Paretiano. Wilfredo Pareto (1848-1923) fu un famoso economista, noto per alcuni fondamentali concetti come quello di **ottimo paretiano** (vedi algoritmi genetici). Si ha un *ottimo paretiano* quando si è raggiunta una situazione in cui **non è possibile migliorare la condizione di una persona senza peggiorare la condizione di un'altra**. L'ottimo paretiano si raggiunge *se si collabora*, a differenza che nell'equilibrio di Nash, *anche se nessuno accetta di peggiorare neppure di poco la propria situazione*.

Nel caso del dilemma del prigioniero si hanno *tre ottimi paretiani*: il caso in cui nessuno dei due prigionieri accusa l'altro (1,1) ed i due casi in cui solo A o solo B accusano l'altro (0,7 e 7,0). Se ci si trova nella situazione in cui entrambi accusano l'altro, infatti, è possibile ottenere un vantaggio per entrambi se tutti e due ritirano l'accusa.

Nelle due situazioni intermedie, invece, se l'unico prigioniero che ha accusato l'altro ritirasse l'accusa avrebbe un danno, passando dalla libertà ad un anno di carcere.

Questi sono 'ottimi di Pareto' perché non è possibile migliorare la situazione dell'uno senza peggiorare quella dell'altro e rappresentano una allocazione ottimale delle risorse. Un 'ottimo di Pareto' non è però necessariamente la situazione migliore per la 'società'.

Il Dilemma del Prigioniero - Implicazioni a più ampio spettro (a livello macro-economico e oltre)

Le strategie *egoistiche* **non portano ad una soluzione ottimale per la società** né se non ci si cura di quanto succede agli altri (come nelll'equilibrio di Nash) ma neppure se si è disposti a collaborare a patto di non avere svantaggi (come nella ricerca dell'ottimo di Pareto). E' possibile che per ottenere la situazione migliore si debba **imporre a qualcuno di rinunciare a qualche cosa**: in una società evoluta questo è il *ruolo della legge* (ad esempio, questo sarebbe un buon motivo per il lockdown!).

Dinamiche Dominanti. Parliamo di dinamiche dominanti quando un giocatore fa il meglio che può indipendentemente da ciò che farà l'altro.

Il **dilemma dei beni pubblici** rappresenta una generalizzazione del dilemma del prigioniero ed include N giocatori. Se ciascun giocatore rispetta, nel consumo individuale, la capacità di auto-rigenerazione del bene pubblico, esso perdurerà nel tempo a beneficio di tutti. Tuttavia, il singolo giocatore è spinto a comportarsi in maniera opportunistica (ad esempio, consideriamo l'inquinamento).

Da un punto di vista pratico, molte situazioni del mondo reale sono paragonabili o convertibili in giochi e sono utilizzati per meglio comprendere le possibili dinamiche di cooperazione tra economie in modo da agire preventivamente su di esse tramite l'introduzione di strumenti di controllo.

Ricerca con Avversari

Il Dilemma del Prigioniero - Alcune curiosità

Il Dilemma del Prigioniero è stato utilizzato per studiare il comportamento degli Stati Uniti e dell'URSS all'epoca della guerra fredda ed, in particolare, per quanto riguarda la corsa agli armamenti. I due giocatori hanno le proprie strategie: non costruire un'arma nucleare (che corrisponde a 'confessare') o costruirla ('non confessare'). Qualunque cosa faccia l'altro, conviene costruire una nuova arma nucleare, ma in questo modo ci si trova nella situazione peggiore: di fatti, non si è mai riusciti ad ottenere la supremazia militare sull'avversario pur avendo speso somme ingenti e reso il mondo meno sicuro.

Altre applicazioni consistono nella modellazione strategica del marketing, la guida di un auto, distribuzione di risorse, e molte altre ancora.

Navigando un minimo su Google, troverete centinaia di problemi che richiedono una modellazione simile. D'altronde, nel 1968, M. Minsky - matematico e scienziato americano, co-fondatore dell'IA Project al MIT - ebbe a dire che:

> *"I giochi non vengono scelti perché sono chiari e semplici, ma perché ci danno la massima complessità con le minime strutture iniziali"*

Ricerca con Avversari

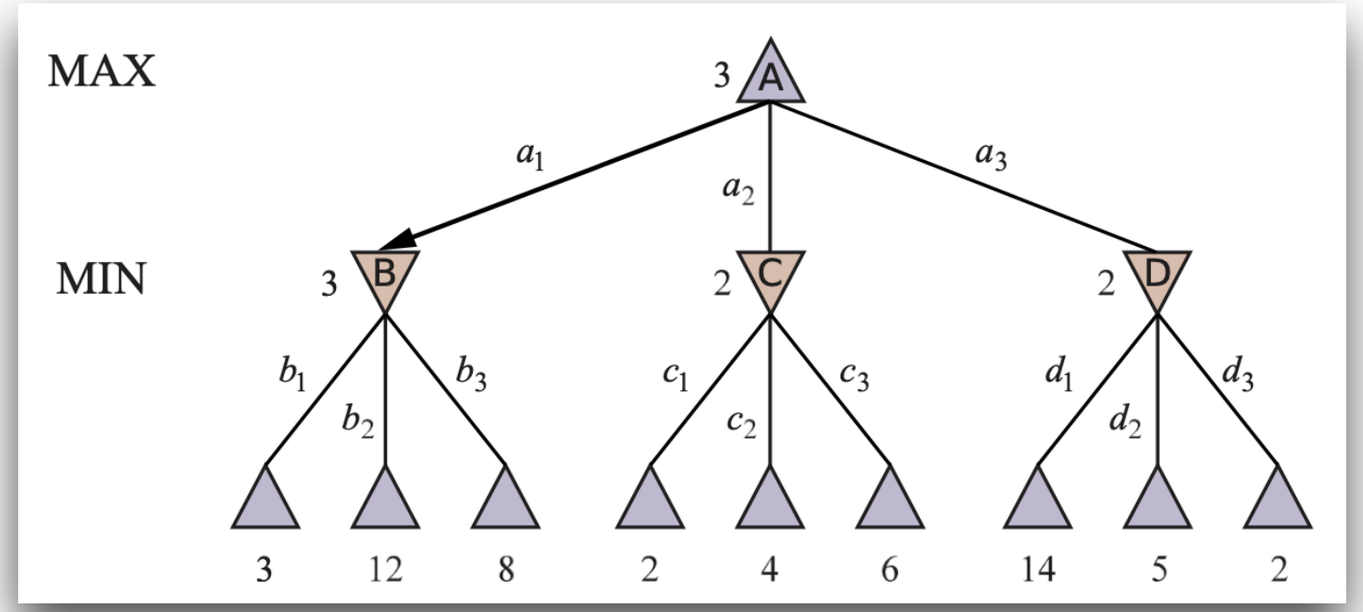
Decisioni ottime, valori e decisioni minimax

Torniamo alle decisioni ottime nei giochi. Abbiamo detto che, a differenza di un normale problema di ricerca, l'agente dovrà necessariamente trovare una strategia contingente che tenga in considerazione le possibile azioni dell'avversario.

Abbiamo anche detto che una strategia ottima è quella che porta ad un risultato almeno pari a quello di qualsiasi altra strategia.

Il problema è, semmai, quello di come poter trovare una strategia ottima. La questione è abbastanza complessa poiché anche semplici giochi (come il tris) sono troppo complessi per poter disegnare un intero albero di gioco.

Per questa ragione, discuteremo un esempio più semplice.



Ricerca con Avversari

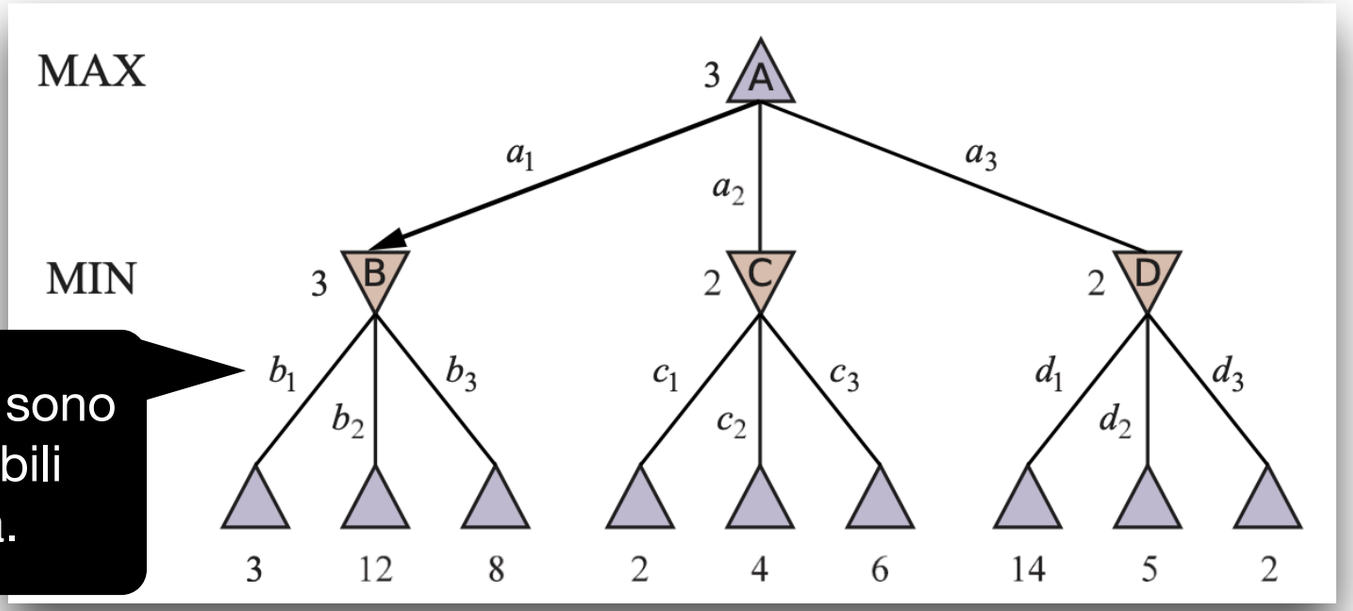
Decisioni ottime, valori e decisioni minimax

Torniamo alle decisioni ottime nei giochi. Abbiamo detto che, a differenza di un normale problema di ricerca, l'agente dovrà necessariamente trovare una strategia contingente che tenga in considerazione le possibile azioni dell'avversario.

Abbiamo anche detto che una strategia ottima è quella che porta ad un risultato almeno pari a quello di qualsiasi altra strategia.

Il problema è, semmai, quello di come poter trovare una strategia ottima. La questione è abbastanza complessa poiché anche semplici giochi (come il tris) sono troppo complessi per poter disegnare un intero albero di gioco.

Per questa ragione, discuteremo un esempio più semplice.



Le mosse possibili per MAX nel nodo radice sono etichettate con a_1, a_2 e a_3, mentre le possibili risposte di MIN con b_1, b_2 e b_3 e così via.

Ricerca con Avversari

Decisioni ottime, valori e decisioni minimax

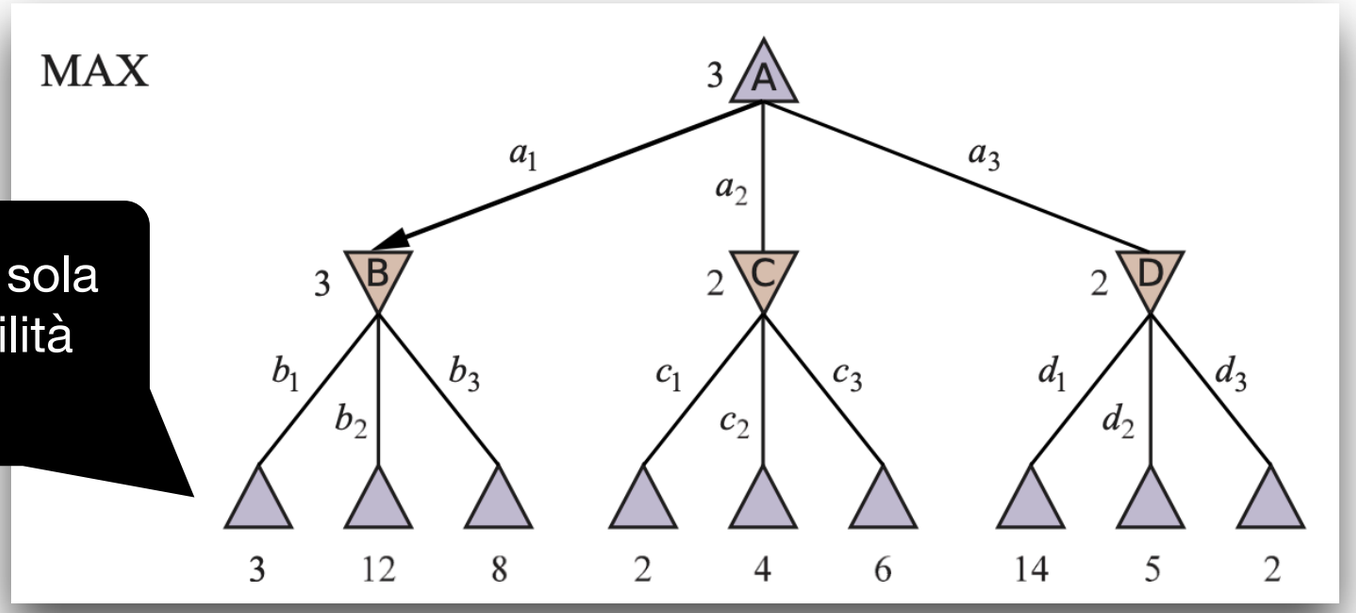
Torniamo alle decisioni ottime nei giochi. Abbiamo detto che, a differenza di un normale problema di ricerca, l'agente dovrà necessariamente trovare una strategia contingente che tenga in considerazione le possibile azioni dell'avversario.

Abbiamo anche detto che una strategia ottima è quella che porta ad un risultato almeno pari a quello di qualsiasi altra strategia.

Il problema è, semmai, quello di come poter trovare una strategia ottima. La questione è abbastanza complessa poiché anche semplici giochi (come il tris) sono troppo complessi per poter disegnare un intero albero di gioco.

Per questa ragione, discuteremo un esempio più semplice.

Questo particolare gioco termina dopo una sola mossa di MAX ed una di MIN. I valori di utilità degli stati terminali vanno da 2 a 14.



Ricerca con Avversari

Decisioni ottime, valori e decisioni minimax

Dato un albero di gioco, la strategia ottima può essere determinata analizzando il **valore minimax di ogni nodo**, che scriveremo MINIMAX(*n*).

Valore Minimax. Il valore minimax di un nodo corrisponde all'utilità di trovarsi nello stato corrispondente, *assumendo che entrambi gli agenti giochino in modo ottimo da lì alla fine della partita*.

Il valore minimax di uno stato terminale si riduce alla sua utilità. Inoltre, avendone la possibilità, MAX preferirà muoversi in uno stato di valore minimax massimo, MIN in uno di valore minimax minimo.

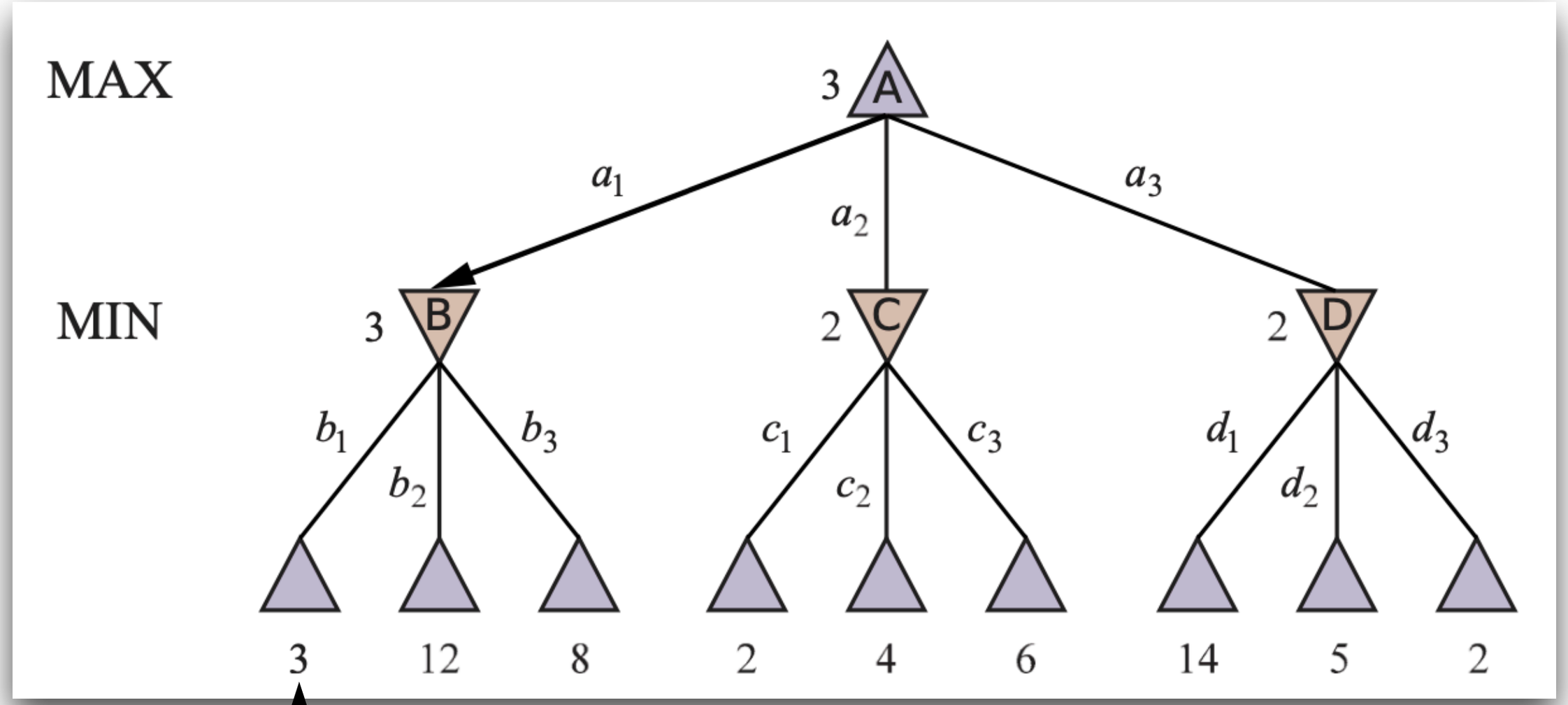
Possiamo formalizzare quanto detto come segue:

$$\text{MINIMAX}(n) = \begin{cases} \text{UTILITÀ}(s, \text{MAX}) & \text{se TEST-TERMINALE}(s); \\ \max_{a \in \text{AZIONI}(s)} \text{VALORE-MINIMAX-RISULTATO}(s, a) & \text{se GIOCATORE}(s) = \text{MAX}; \\ \min_{a \in \text{AZIONI}(s)} \text{VALORE-MINIMAX-RISULTATO}(s, a) & \text{se GIOCATORE}(s) = \text{MIN}; \end{cases}$$

Applichiamo queste definizioni sull'albero di gioco preso come esempio.

Ricerca con Avversari

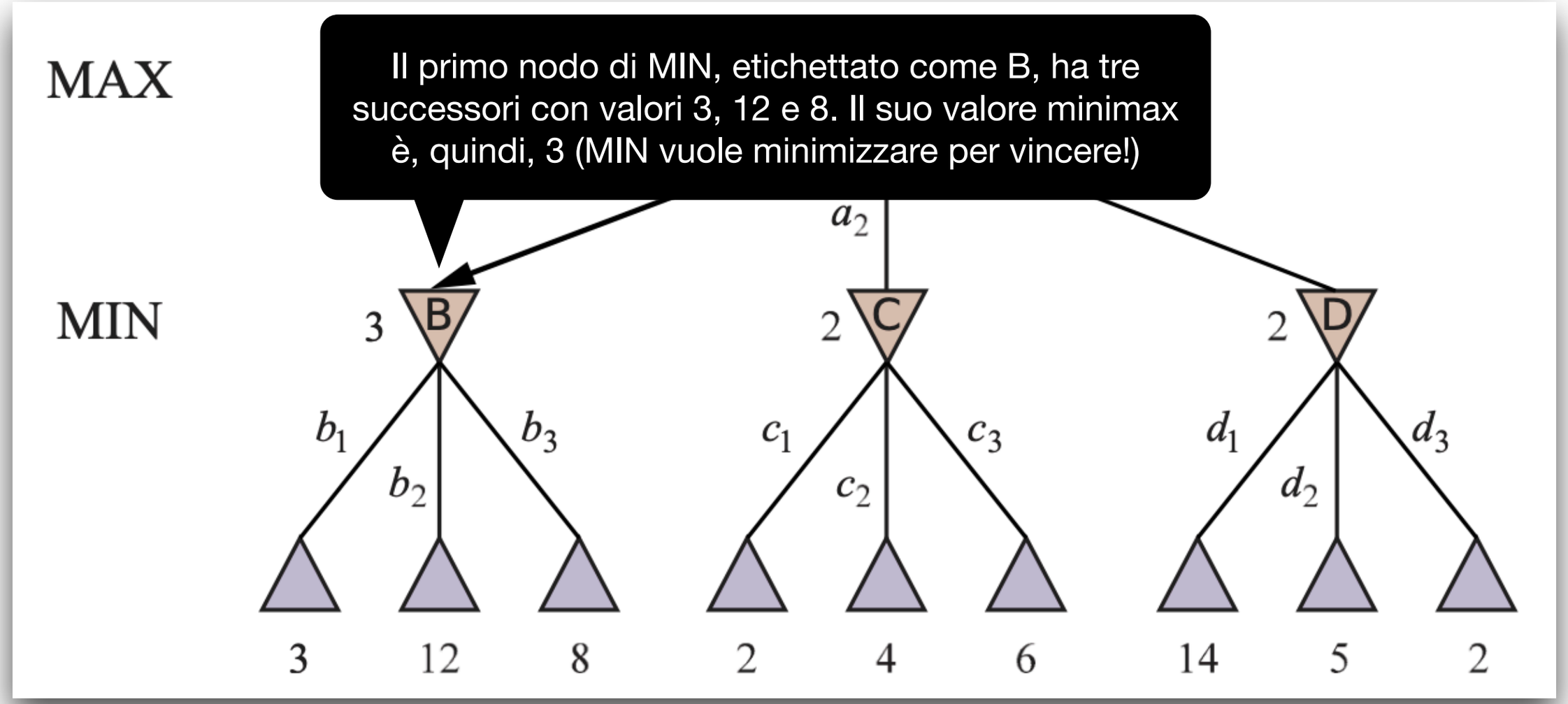
Decisioni ottime, valori e decisioni minimax



I nodi terminali al livello più basso ottengono i loro valori dalla funzione UTILITÀ del gioco.

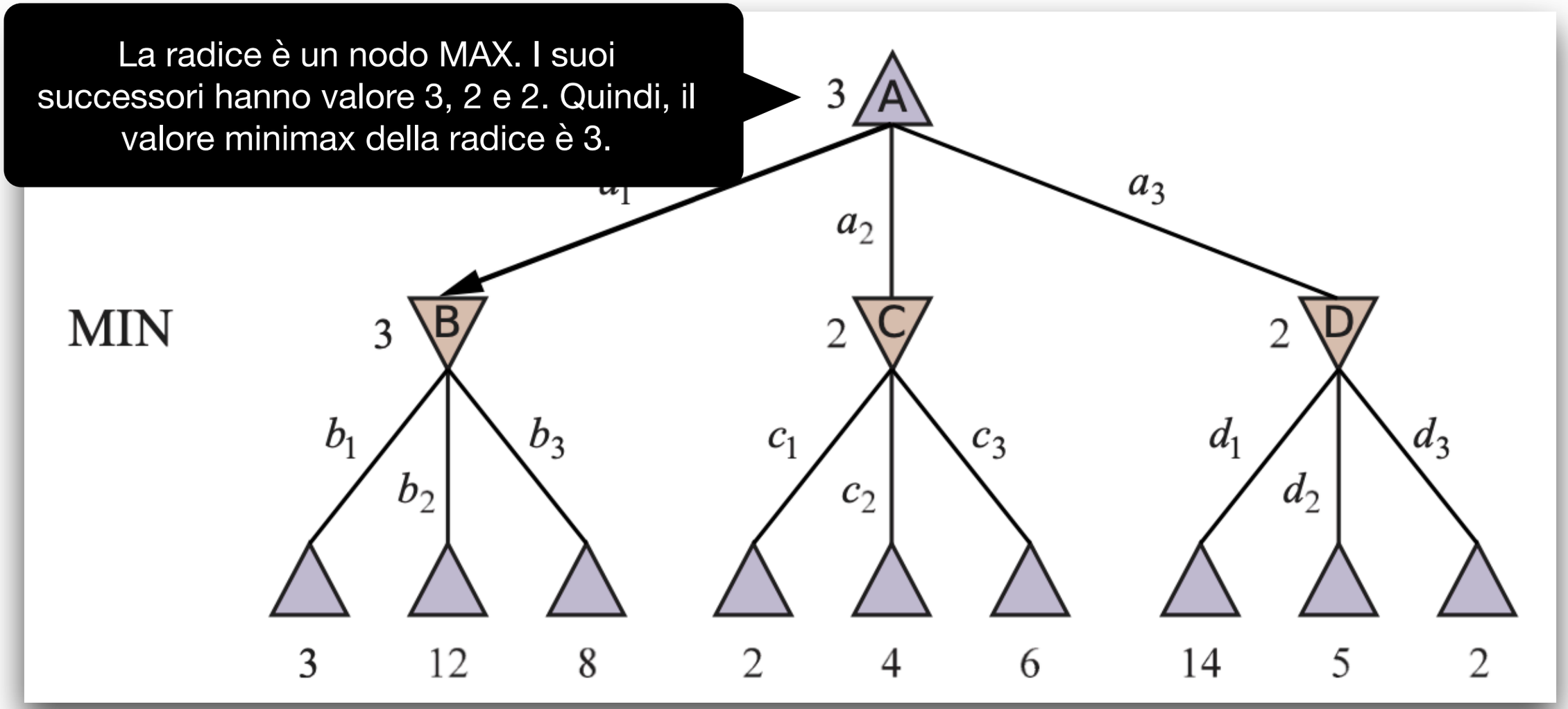
Ricerca con Avversari

Decisioni ottime, valori e decisioni minimax



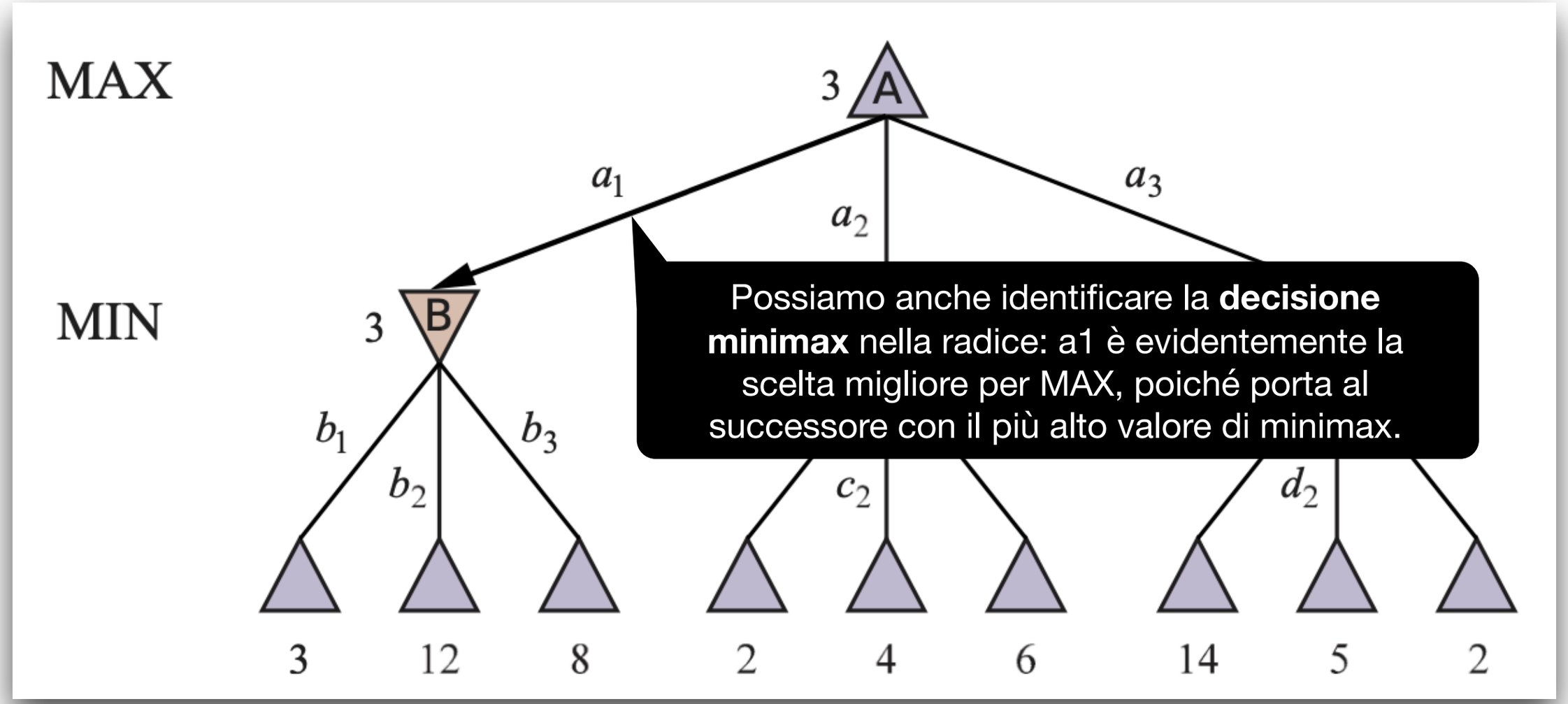
Ricerca con Avversari

Decisioni ottime, valori e decisioni minimax



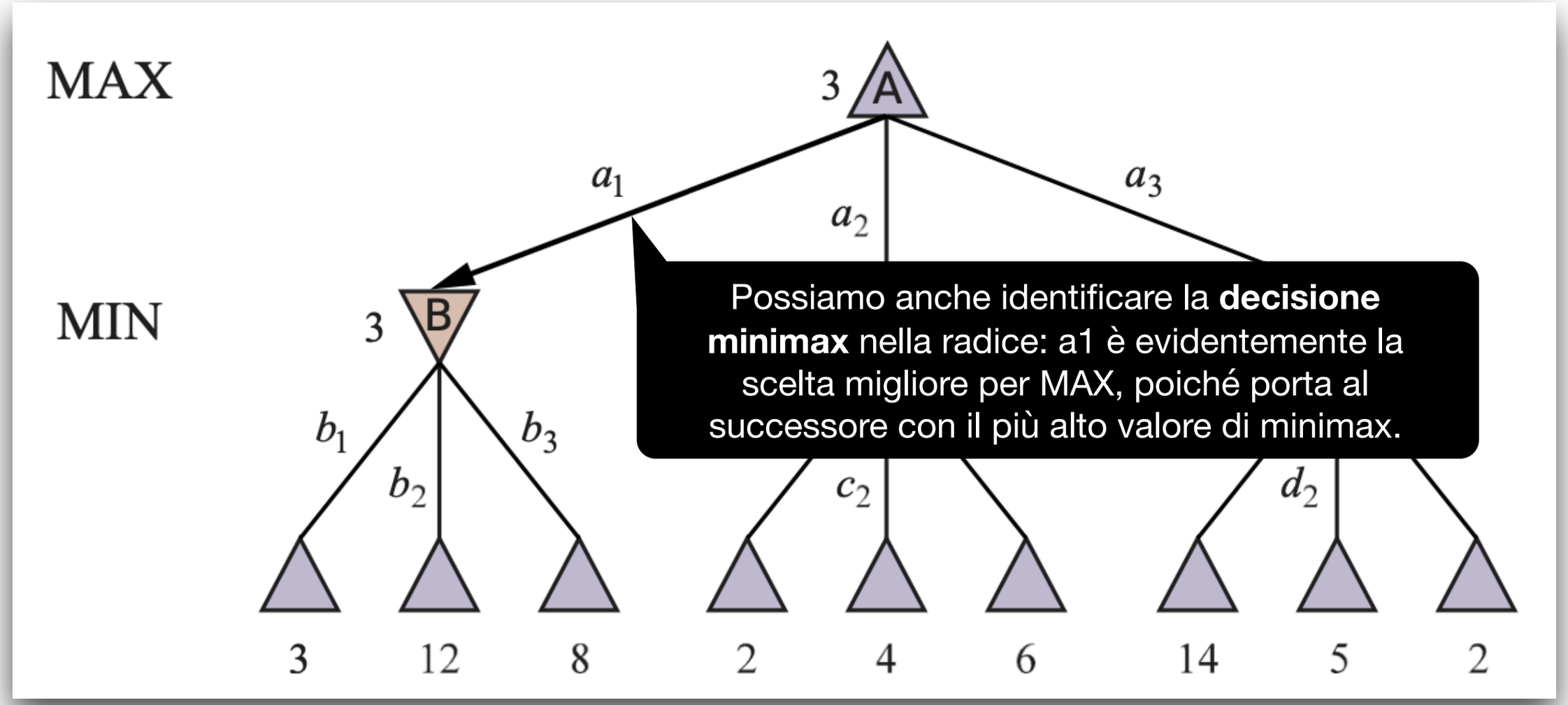
Ricerca con Avversari

Decisioni ottime, valori e decisioni minimax



Ricerca con Avversari

Decisioni ottime, valori e decisioni minimax



Come detto, questa strategia ottima per MAX presume che anche MIN giochi in modo ottimo: in altri termini, la strategia massimizza il risultato di MAX *nel caso pessimo*.

Se MIN non gioca ottimamente, allora MAX farà sicuramente meglio, massimizzando ancora di più i suoi valori minimax.

Ricerca con Avversari

Algoritmo Minimax

```
function DECISIONE-MINIMAX(s) returns un'azione
   return argmax_{a∈AZIONI(s)} VALORE-MIN(RISULTATO(s, a))
```

```
function VALORE-MAX(s) returns un valore di utilità
   if TEST-TERMINAZIONE(s) then return UTILITÀ(s)
   v <- -∞

   for each a in AZIONI(s) do
      v <- MAX(v, VALORE-MIN(RISULTATO(s, a)))

   return v
```

```
function VALORE-MIN(s) returns un valore di utilità
   if TEST-TERMINAZIONE(s) then return UTILITÀ(s)
   v <- +∞

   for each a in AZIONI(s) do
      v <- MIN(v, VALORE-MAX(RISULTATO(s, a)))

   return v
```

Ricerca con Avversari

Algoritmo Minimax

```
function DECISIONE-MINIMAX(s) returns un'azione
   return argmax_{a∈AZIONI(s)} VALORE-MIN(RISULTATO(s, a))
```

```
function VALORE-MAX(s) returns un valore di utilità
   if TEST-TERMINAZIONE(s) then return UTILITÀ(s)
   v <- -∞

   for each a in AZIONI(s) do
      v <- MAX(v, VALORE-MIN(RISULTATO(s, a)))

   return v
```

/* Le funzioni VALORE-MIN e VALORE-MAX
non fanno altro che attraversare l'intero
albero di gioco fino alle foglie per determinare
il valore da "far risalire". */

```
function VALORE-MIN(s) returns un valore di utilità
   if TEST-TERMINAZIONE(s) then return UTILITÀ(s)
   v <- +∞

   for each a in AZIONI(s) do
      v <- MIN(v, VALORE-MAX(RISULTATO(s, a)))

   return v
```

Ricerca con Avversari

Algoritmo Minimax

```
function DECISIONE-MINIMAX(s) returns un'azione
   return argmax_{a∈AZIONI(s)} VALORE-MIN(RISULTATO(s, a))
```

```
function VALORE-MAX(s) returns un valore di utilità
   if TEST-TERMINAZIONE(s) then return UTILITÀ(s)
   v <- -∞

   for each
      v <- M

   return v
```

/* Se troviamo un nodo foglia, restituiamo proprio il valore utilità della foglia (e l'unico che possiamo avere dopotutto). */

```
function VALORE-MIN(s) returns un valore di utilità
   if TEST-TERMINAZIONE(s) then return UTILITÀ(s)
   v <- +∞

   for each a in AZIONI(s) do
      v <- MIN(v, VALORE-MAX(RISULTATO(s, a)))

   return v
```

Ricerca con Avversari

Algoritmo Minimax

```
function DECISIONE-MINIMAX(s) returns un'azione
   return argmax_{a∈AZIONI(s)} VALORE-MIN(RISULTATO(s, a))
```

```
function VALORE-MAX(s) returns un valore di utilità
   if TEST-TERMINAZIONE(s) then return UTILITÀ(s)
   v <- -∞

   for each a in AZIONI(s) do
      v <- MAX(v, VALORE-MIN(RISULTATO(s, a)))

   return v
```

/* Altrimenti, iteriamo sulle azioni e, per ognuna di esse, cerchiamo il valore minimax. Per MAX, questo sarà dato dal valore massimo contenuto nel sotto-albero di livello inferiore - il quale è costruito tramite la funzione VALORE-MIN. */

```
function VALORE-M
   if TEST-TERMIN
   v <- +∞

   for each a in AZIONI(s) do
      v <- MIN(v, VALORE-MAX(RISULTATO(s, a)))

   return v
```

Ricerca con Avversari

Algoritmo Minimax

```
function DECISIONE-MINIMAX(s) returns un'azione
   return argmax_{a∈AZIONI(s)} VALORE-MIN(RISULTATO(s, a))
```

```
function VALORE-MAX(s) returns un valore di utilità
   if TEST-TERMINAZIONE(s) then return UTILITÀ(s)
   v <- -∞

   for each a in AZIONI(s) do
      v <- MAX(v, VALORE-MIN(RISULTATO(s, a)))

   return v
```

```
function VALORE-MI
   if TEST-TERMINA
   v <- +∞

   for each a in AZIONI(s) do
      v <- MIN(v, VALORE-MAX(RISULTATO(s, a)))

   return v
```

/* Per MIN, il discorso è simile. Cercheremo il valore minimo del sotto-albero costruito con la funzione `VALORE-MAX`. */

Ricerca con Avversari

Algoritmo Minimax

```
function DECISIONE-MINIMAX(s) returns un'azione
   return argmax_{a∈AZIONI(s)} VALORE-MIN(RISULTATO(s, a))
```

```
function VALORE-MA
   if TEST-TERMINA
   v <- -∞

   for each a in AZIONI(s) do
      v <- MAX(v, VALORE-MIN(RISULTATO(s, a)))

   return v
```

/* La notazione `argmax` calcola semplicemente il valore massimo del sotto-albero di livello inferiore. */

```
function VALORE-MIN(s) returns un valore di utilità
   if TEST-TERMINAZIONE(s) then return UTILITÀ(s)
   v <- +∞

   for each a in AZIONI(s) do
      v <- MIN(v, VALORE-MAX(RISULTATO(s, a)))

   return v
```

Ricerca con Avversari

Algoritmo Minimax, performance

La ricerca minimax non è altro che una ricerca in profondità e, pertanto, eredita molti degli aspetti già visti nelle precedenti lezioni:

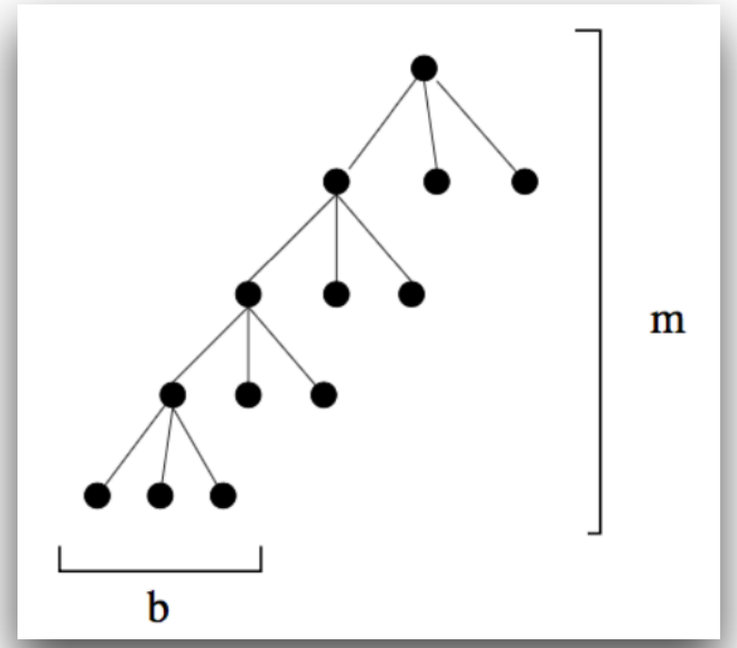
Completezza: L'algoritmo è completo se l'albero è finito - molti giochi, inclusi gli scacchi, hanno regole specifiche per garantire il termine del gioco.

Ottimalità: Come già detto, l'algoritmo è ottimo nel caso in cui l'avversario è ottimo.

Complessità temporale: La complessità è governata dalla dimensione dell'albero di gioco. Sia *m* la profondità massima, nel caso pessimo la ricerca genererà $O(b^m)$ nodi.

Complessità spaziale: La ricerca deve memorizzare un solo cammino dalla radice ad un nodo foglia, insieme ai rimanenti nodi fratelli non espansi per ciascun nodo sul cammino. Una volta che un nodo è stato espanso, può essere rimosso dalla memoria non appena tutti i suoi discendenti sono stati esplorati completamente.

Per un albero di gioco con fattore di ramificazione *b* e profondità *m*, la complessità sarà di O(bm).



Ricerca con Avversari

Algoritmo Minimax, performance

La ricerca minimax non è altro che una ricerca in profondità e, pertanto, eredita molti degli aspetti già visti nelle precedenti lezioni:

Completezza: L'algoritmo è completo se l'albero è finito - molti giochi, inclusi gli scacchi, hanno regole specifiche per garantire il termine del gioco.

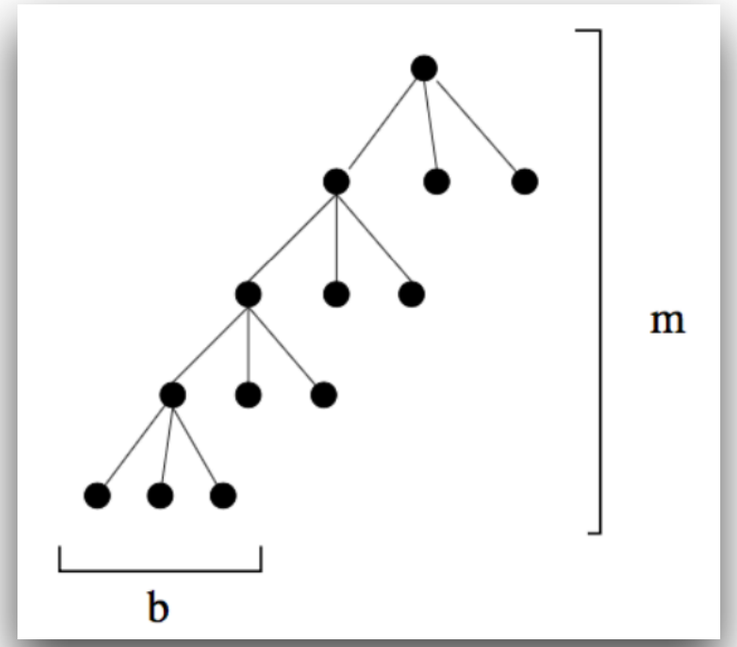
Ottimalità: Come già detto, l'algoritmo è ottimo nel caso in cui l'avversario è ottimo.

Complessità temporale: La complessità è governata dalla dimensione dell'albero di gioco. Sia *m* la profondità massima, nel caso pessimo la ricerca genererà $O(b^m)$ nodi.

Per i giochi reali, questa complessità temporale è improponibile! Nella prossima lezione vedremo come migliorare l'usabilità di questo algoritmo.

cammino. Una volta che un nodo è stato espanso, può essere rimosso dalla memoria non appena tutti i suoi discendenti sono stati esplorati completamente.

Per un albero di gioco con fattore di ramificazione *b* e profondità *m*, la complessità sarà di O(bm).

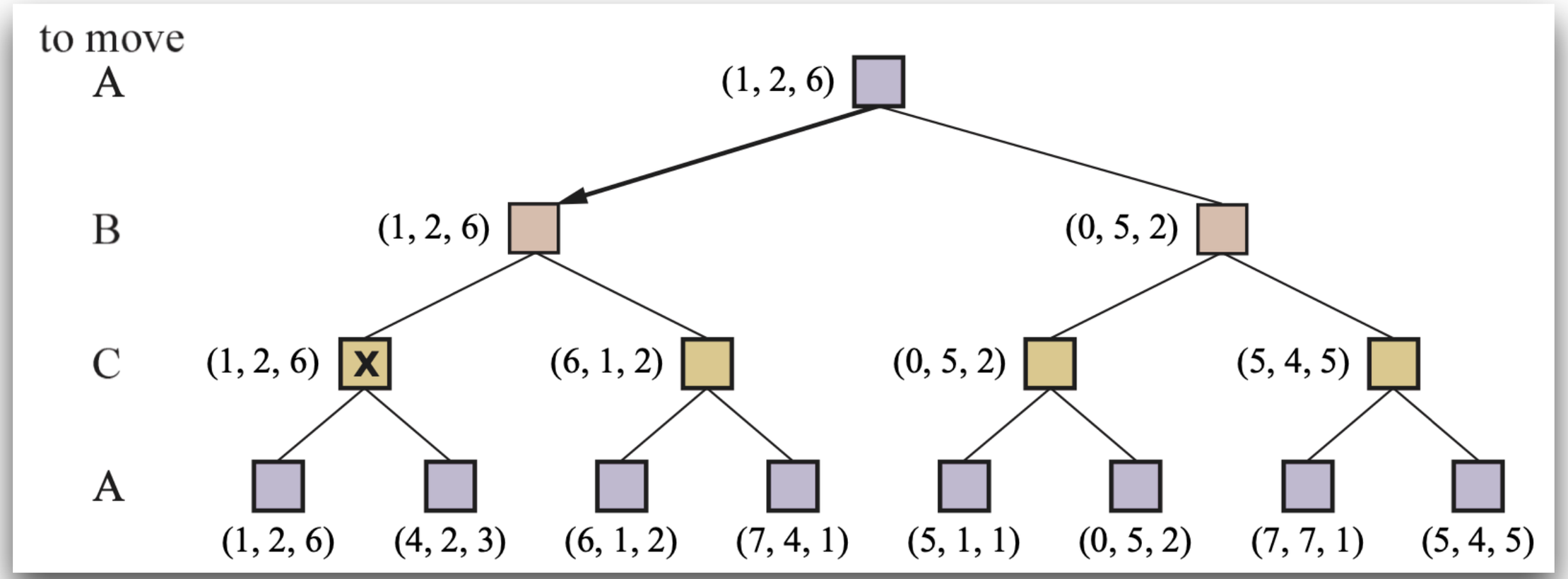


Ricerca con Avversari

Algoritmo Minimax in giochi multi-player

Finora, abbiamo considerato lo scenario in cui due giocatori si alternano, effettuando mosse fino al termine della partita.

Chiaramente, molti giochi prevedono l'esistenza di più giocatori. L'algoritmo minimax può essere adattato e utilizzato anche in questi contesti. Tuttavia, ci sono da fare alcune considerazioni. Prendiamo, ad esempio, un gioco con tre giocatori: A, B e C.

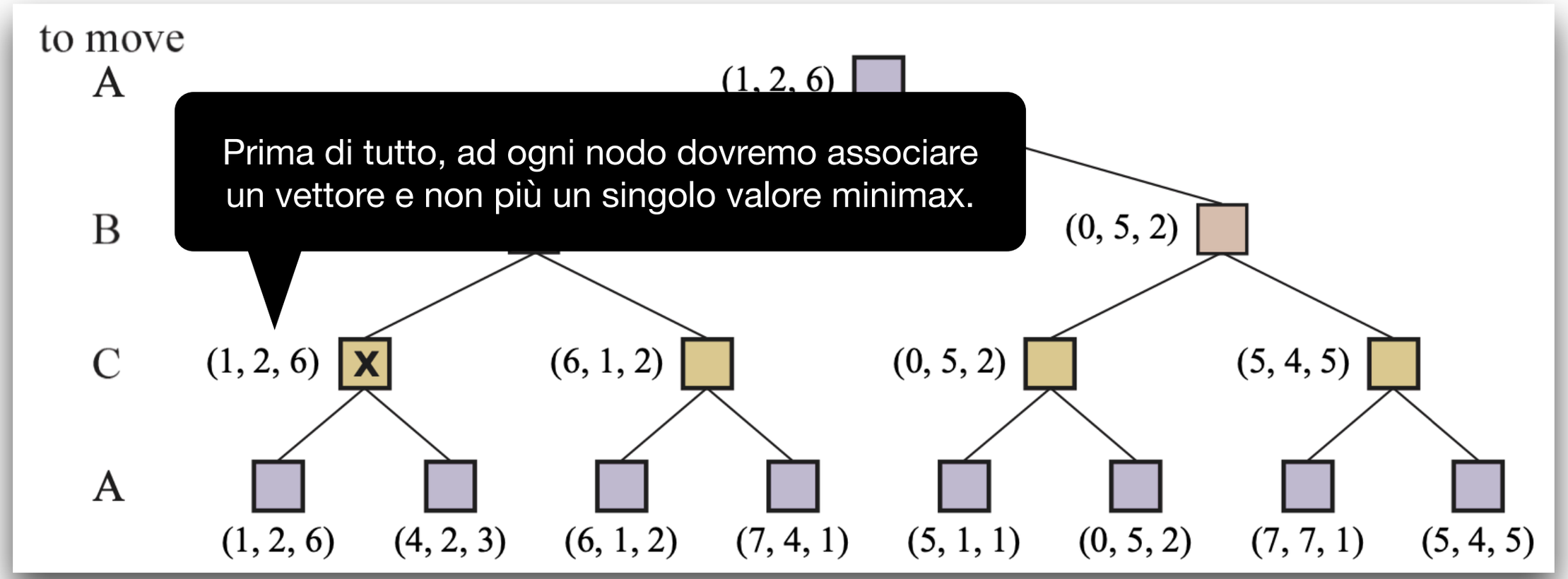


Ricerca con Avversari

Algoritmo Minimax in giochi multi-player

Finora, abbiamo considerato lo scenario in cui due giocatori si alternano, effettuando mosse fino al termine della partita.

Chiaramente, molti giochi prevedono l'esistenza di più giocatori. L'algoritmo minimax può essere adattato e utilizzato anche in questi contesti. Tuttavia, ci sono da fare alcune considerazioni. Prendiamo, ad esempio, un gioco con tre giocatori: A, B e C.

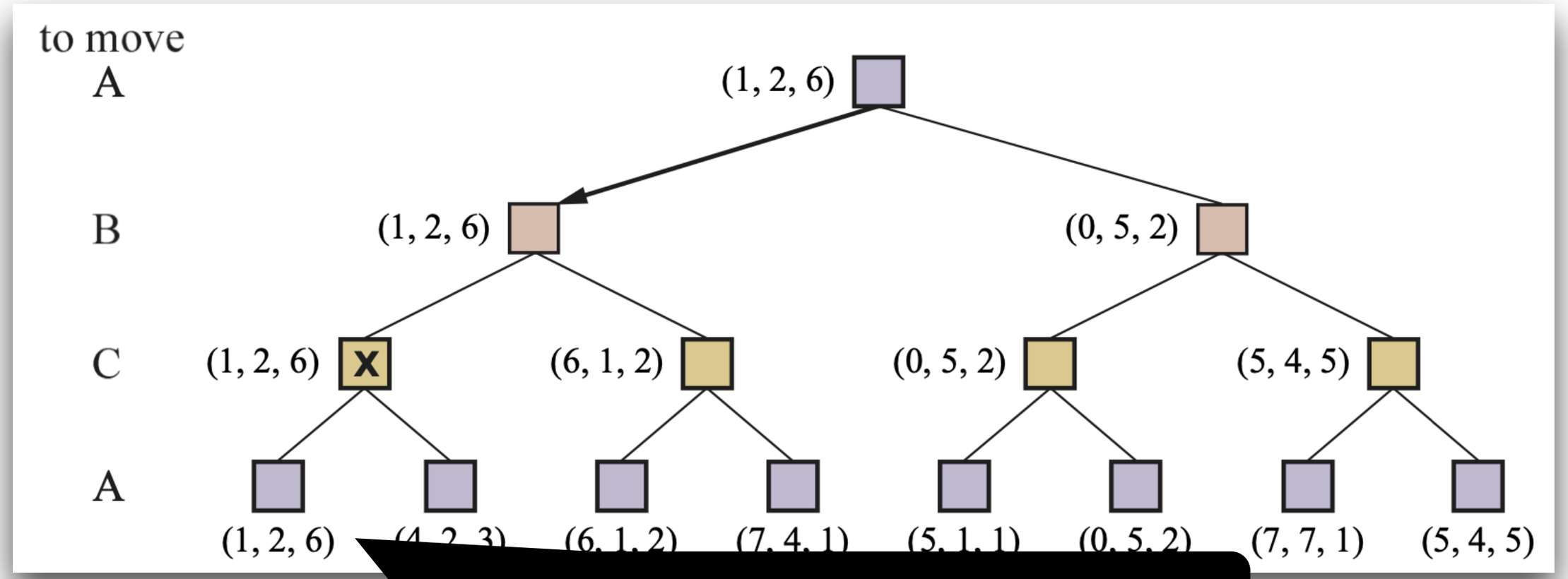


Ricerca con Avversari

Algoritmo Minimax in giochi multi-player

Finora, abbiamo considerato lo scenario in cui due giocatori si alternano, effettuando mosse fino al termine della partita.

Chiaramente, molti giochi prevedono l'esistenza di più giocatori. L'algoritmo minimax può essere adattato e utilizzato anche in questi contesti. Tuttavia, ci sono da fare alcune considerazioni. Prendiamo, ad esempio, un gioco con tre giocatori: A, B e C.



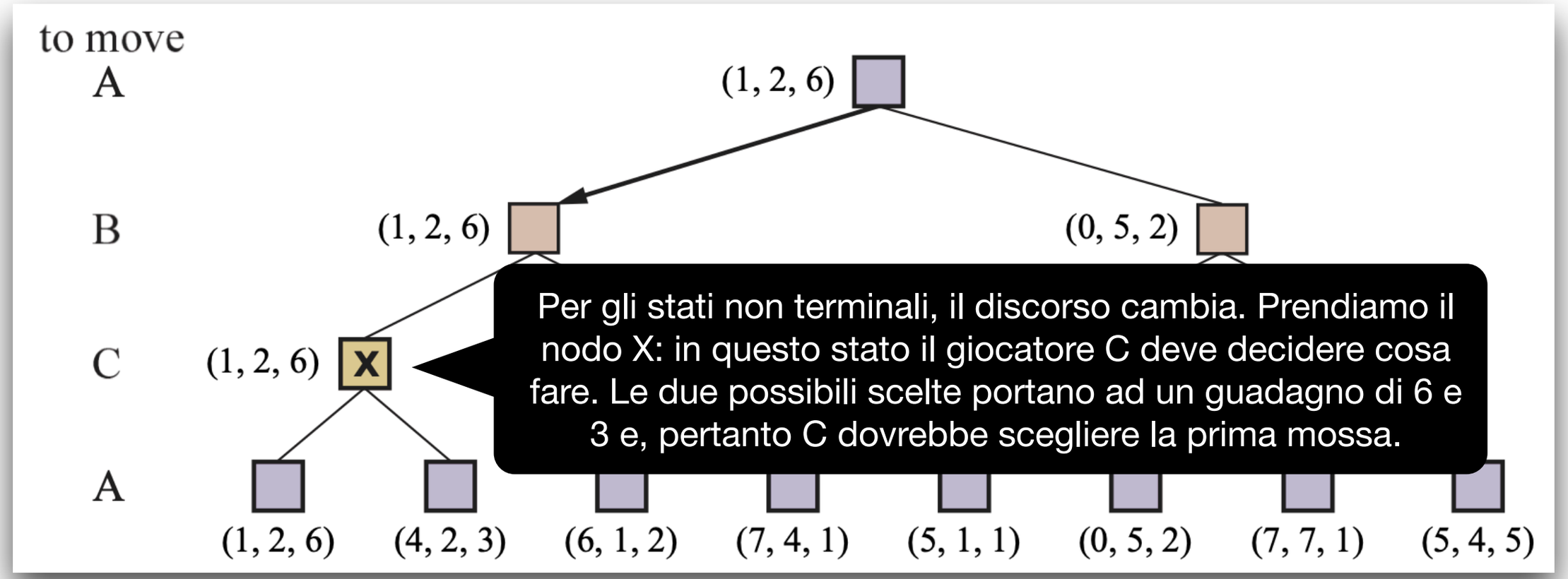
Per gli stati terminali, il vettore fornirà l'utilità dello stato dal punto di vista di ogni giocatore.

Ricerca con Avversari

Algoritmo Minimax in giochi multi-player

Finora, abbiamo considerato lo scenario in cui due giocatori si alternano, effettuando mosse fino al termine della partita.

Chiaramente, molti giochi prevedono l'esistenza di più giocatori. L'algoritmo minimax può essere adattato e utilizzato anche in questi contesti. Tuttavia, ci sono da fare alcune considerazioni. Prendiamo, ad esempio, un gioco con tre giocatori: A, B e C.

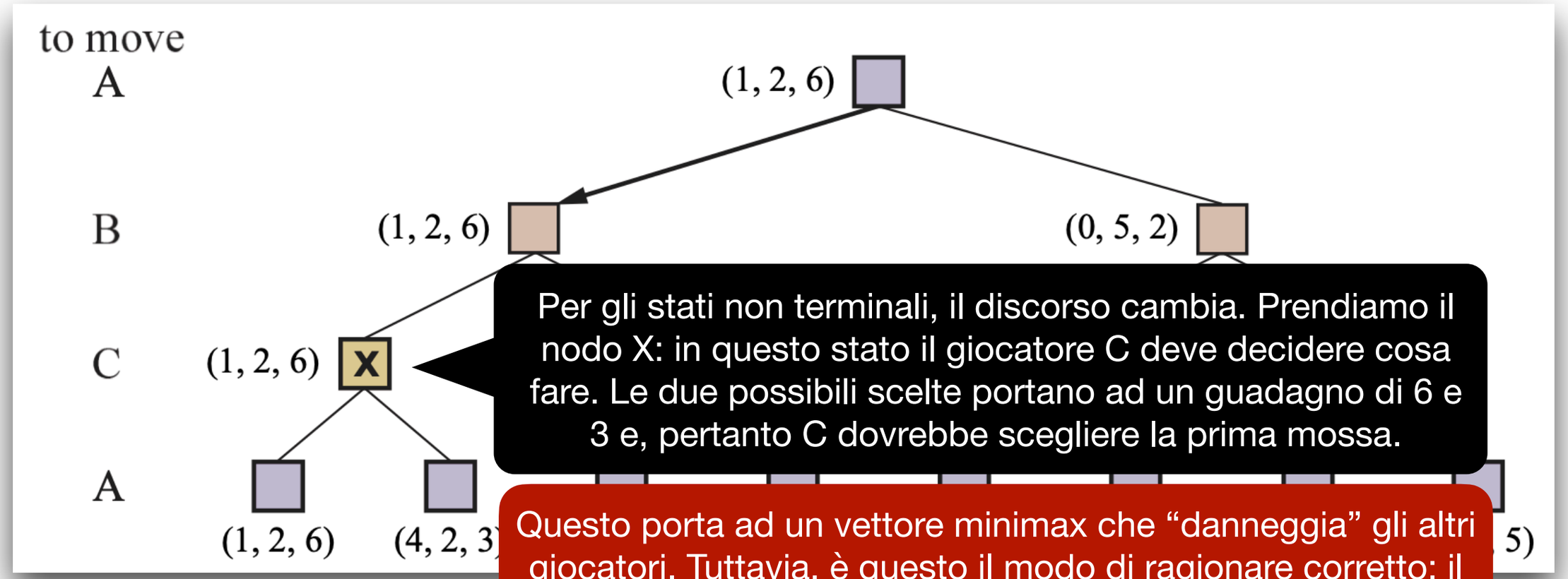


Ricerca con Avversari

Algoritmo Minimax in giochi multi-player

Finora, abbiamo considerato lo scenario in cui due giocatori si alternano, effettuando mosse fino al termine della partita.

Chiaramente, molti giochi prevedono l'esistenza di più giocatori. L'algoritmo minimax può essere adattato e utilizzato anche in questi contesti. Tuttavia, ci sono da fare alcune considerazioni. Prendiamo, ad esempio, un gioco con tre giocatori: A, B e C.



Per gli stati non terminali, il discorso cambia. Prendiamo il nodo X: in questo stato il giocatore C deve decidere cosa fare. Le due possibili scelte portano ad un guadagno di 6 e 3 e, pertanto C dovrebbe scegliere la prima mossa.

Questo porta ad un vettore minimax che "danneggia" gli altri giocatori. Tuttavia, è questo il modo di ragionare corretto: il valore passato verso l'alto è **il vettore di utilità dello stato successore più favorevole a chi sta scegliendo la mossa**.

Ricerca con Avversari

Algoritmo Minimax in giochi multi-player

Finora, abbiamo considerato lo scenario in cui due giocatori si alternano, effettuando mosse fino al termine della partita.

Chiaramente, molti giochi prevedono l'esistenza di più giocatori. L'algoritmo minimax può essere adattato e utilizzato anche in questi contesti. Tuttavia, ci sono da fare alcune considerazioni. Prendiamo, ad esempio, un gioco con tre giocatori: A, B e C.

Altre complicazioni, a livello di design, consistono nell'esistenza di *alleanze* tra giocatori. Ad esempio, A e B potrebbero voler decidere di sfavorire C nel caso in cui questo si trovi in una situazione di forza in un determinato momento del gioco.

Sono le alleanze parte della strategia ottima di ogni giocatore? Possono le alleanze variare dinamicamente durante lo svolgimento del gioco?

Apparentemente, i risultati raggiunti in ambito di ricerca ci dicono che sì, le alleanze fanno parte della strategia ottima e dovrebbero quindi essere modellate all'interno dell'albero di gioco.

Più in generale, però, possiamo dire che ancora una volta la questione principale sta nel design del problema, nella comprensione del contesto e di come i giocatori possono interagire tra loro, in maniera cooperativa o non cooperativa.

UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Laurea triennale in Informatica

Fondamenti di Intelligenza Artificiale

Lezione 10 - Ricerca con avversari