ESERCIZI PER IL CORSO DI INTELLIGENZA ARTIFICIALE
I parte del corso - Anno accademico 2003/2004

M. Simi [*]

[*] La dott.ssa Chiara Renso ha contribuito alla stesura di questi esercizi nell'a.a. 1997/98. Il dott. Andrea Bracciali ha contribuito alla soluzione di alcuni esercizi negli anni 1998-2000. Il dott. Fabio Ganovelli ha contribuito alcuni esercizi e soluzioni nell'anno 2001/02.

Introduzione all'IA

Definizioni di IA

Discutere le seguenti definizioni di I.A. e dire se la visione dominante è quella dell'I.A. forte o dell'IA debole:

a. "… una raccolta di algoritmi computazionalmente trattabili, approssimazione adeguata di problemi non trattabili" [Partridge, 91].

IA debole.

b. " … l'impresa di costruire sistemi di simboli fisici che possono passare in maniera affidabile il Test di Turing" [Ginsberg, 93].

IA forte.

c. "… il campo dell'informatica che studia come si possono creare macchine che si comportano in maniera intelligente" [Jackson 86].

Neutra.

d. "… un campo di studio che comprende tecniche computazionali per compiere compiti che apparentemente richiedono intelligenza se compiuti da umani" [Tanimoto 90].

IA debole.

e. "… una investigazione generale sulla natura dell'intelligenza e i principi e meccanismi richiesti per comprenderla e replicarla"[Sharples 89].

IA forte.

f. "… il rendere i calcolatori in grado di fare cose che sembrano intelligenti" [Rowe 88]

Neutra.

Agenti

Definizioni relative agli agenti

Riportare, con parole vostre, le definizioni di

a. agente
b. agente razionale
c. agente autonomo

secondo Russell e Norvig. Un programma tradizionale è un agente secondo Russell e Norvig?
(NOTA: questo esercizio è stato assegnato senza materiale di consultazione)

Formulazione PAGE

Dare una formulazione PEAS (*Performance, Environment, Actions, Sensors*) di agenti che assolvono i seguenti compiti:

a) Sistema didattico per la lingua inglese
b) Bilbiotecario
c) Giocatore di scacchi
d) Giocatore di scacchi con tempo limitato per effettuare una mossa
e) Robot che raccoglie lattine vuote di bibite e le porta in un raccoglitore
f) *Information broker*: un agente che a seguito di un bisogno informativo espresso dal suo committente umano, ricerca in Internet documenti che potrebbero essere rilevanti; l'agente potrebbe utilizzare un feedback dal committente per imparare di più sui suoi gusti e interessi e migliorare le sue prestazioni con l'esperienza; l'agente potrebbe infine anche essere pro-attivo, nel senso di arrivare a suggerire nuovi interessi o documenti su temi non esplicitamente richiesti.

Problema	P	E	A	S
Sistema didattico per la lingua inglese	Massimizzare il punteggio dello studente nei test	Studente	Proponi test mirati, valuta, correggi, dai suggerimenti	Frasi (sequenze di caratteri)
Bibliotecario	Gestire patrimonio librario	Utenti, libri, fornitori	Ricerca libro, consegna libro, registra prestito, sollecito prestito, ordini a fornitori, catalogazione …	Richieste libri, ordinazioni, prenotazioni, nuovi libri, restituzione
Giocatore di scacchi	Vincere	Avversari, scacchiera	Mosse legali	Stato del gioco (scacchiera e pezzi catturati)
Robot che raccoglie lattine	Portare le lattine vuote nei bidoni	Stanze, lattine, bidoni per la raccolta	Riconosce lattine, raccoglie, trasporta, si muove, deposita	Immagini da telecamera
Information broker	Tenere il committente aggiornato; suggerire nuovi interessi	Internet, committente	Ricerca in Internet, selezione, suggerimenti di documenti interessanti, aggiornamento modello utente	Interessi e bisogni informativi del committente, valutazione di rilevanza dei doc proposti
Giocatore di calcio	Fare più goal possibile	Altri giocatori (compagni e avversari), campo di calcio	Dare calci al pallone, correre	Locazione pallone altri giocatori, porte

Tipi di agenti

Che tipo di agente usereste per …

a. … il puzzle dell'otto.

Agente con obiettivo, una funzione di utilità non è necessaria come in tutti i giochi in cui l'essenziale è vincere piuttosto che vincere bene (non c'è un punteggio che misura le configurazioni vincenti)

b. … il guidatore di taxi

Agente con funzione di utilità: è un caso in cui ci sono obiettivi in conflitto tra di loro: la sicurezza di guida, il comfort, la rapidità con cui si raggiunge la destinazione, la necessità di non commettere infrazioni.

4. Con che tipo di ambiente ha a che fare un agente per …

a. … il gioco dell'otto
b. … gli scacchi
c. … gli scacchi con tempo prefissato per le mosse
d. … la briscola
e. … il raccoglitore di lattine

	Osservabile/no	Deterministico/ stocastico	Episodico/no	Statico/ dinamico	Discreto/continuo
Gioco dell'otto	SI	SI	NO	Statico	Discreto
Scacchi	SI	SI (*)	NO	Statico	Discreto
Scacchi con scadenza	SI	SI (*)	NO	Semi-dinamico (**)	Discreto
Briscola	NO	NO	NO	Statico	Discreto
Raccoglitore di lattine	NO	NO	NO	Dinamico	Continuo

(*) *Sono deterministici perché l'effetto dell'azione è certa; il fatto che la mossa dell'avversario non è prevedibile li rende problemi "con contingenza". In queste situazioni l'agente non riesce a pianificare tutta*

una sequenza di azioni ma può solo pianificare la mossa successiva, eseguirla, vedere cosa succede e ripianificare.
(**) *Si dice semi-dinamico un ambiente in cui l'ambiente non cambia col passare del tempo ma ci sono dei vincoli temporali da rispettare e quindi le prestazioni dell'agente dipendono dal tempo che ci mette a rispondere.*

Valutazione prestazioni/utilità

Spiegare la differenza tra la funzione di valutazione di prestazioni di un agente e la funzione di utilità.

La misura delle prestazioni è una misura esterna ed oggettiva del comportamento dell'agente che ci dice in che misura l'agente si è comportato in maniera razionale; questa funzione sta nell'ambiente e serve a misurare la bontà della soluzione (la sequenza di azioni) generata dall'agente.
La funzione di utilità è una funzione che valuta la bontà di uno stato obiettivo in termini di bontà per l'agente e quindi da un punto di vista interno e soggettivo. Serve all'agente a decidere quale obiettivo perseguire.

Valutazione euristica/utilità

Spiegare la differenza tra la funzione di valutazione euristica di uno stato e la funzione di utilità.

La funzione di valutazione euristica si applica agli stati intermedi nel processo di ricerca e ci da una stima della bontà di uno stato, una misura di quanto promettente è rispetto al raggiungimento di un obiettivo. La funzione di utilità si applica gli stati obiettivo e ne valuta la bontà per l'agente. Negli agenti con funzione di utilità la funzione di valutazione euristica è una stima della utilità attesa a partire dallo stato e dovrebbe essere in accordo con la funzione di utilità sugli stati terminali, guidando la ricerca verso gli stati più utili per l'agente.

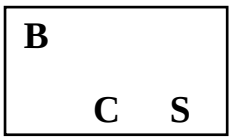
Formulazione di Problemi

La scimmia e la banana

Una scimmia sta in una stanza, in cui si trovano una cassa e un casco di banane appeso al soffitto. La scimmia può raggiungere le banane solo salendo sulla cassa. La scimmia può muoversi, spostare la cassa, salirvi sopra. La scimmia desidera mangiarsi le banane.

a. Formulare il seguente problema come ricerca in uno spazio di stati.
b. Dire che algoritmo di ricerca usereste per trovare la soluzione (quale direzione di ricerca, se controllo sugli stati ripetuti, quale strategia), supponendo che ci interessi una soluzione ottimale.
c. Se non interessasse la ottimalità, quale algoritmo usereste?

a) Lo stato è una rappresentazione degli aspetti rilevanti per la soluzione del problema: B, le banane, C, la cassa, S, la scimmia e loro proprietà tipo la loro locazione (1,2,3), se la scimmia è sulla cassa o no, se la scimmia ha le banane o no.
Lo stato iniziale e lo stato obiettivo:

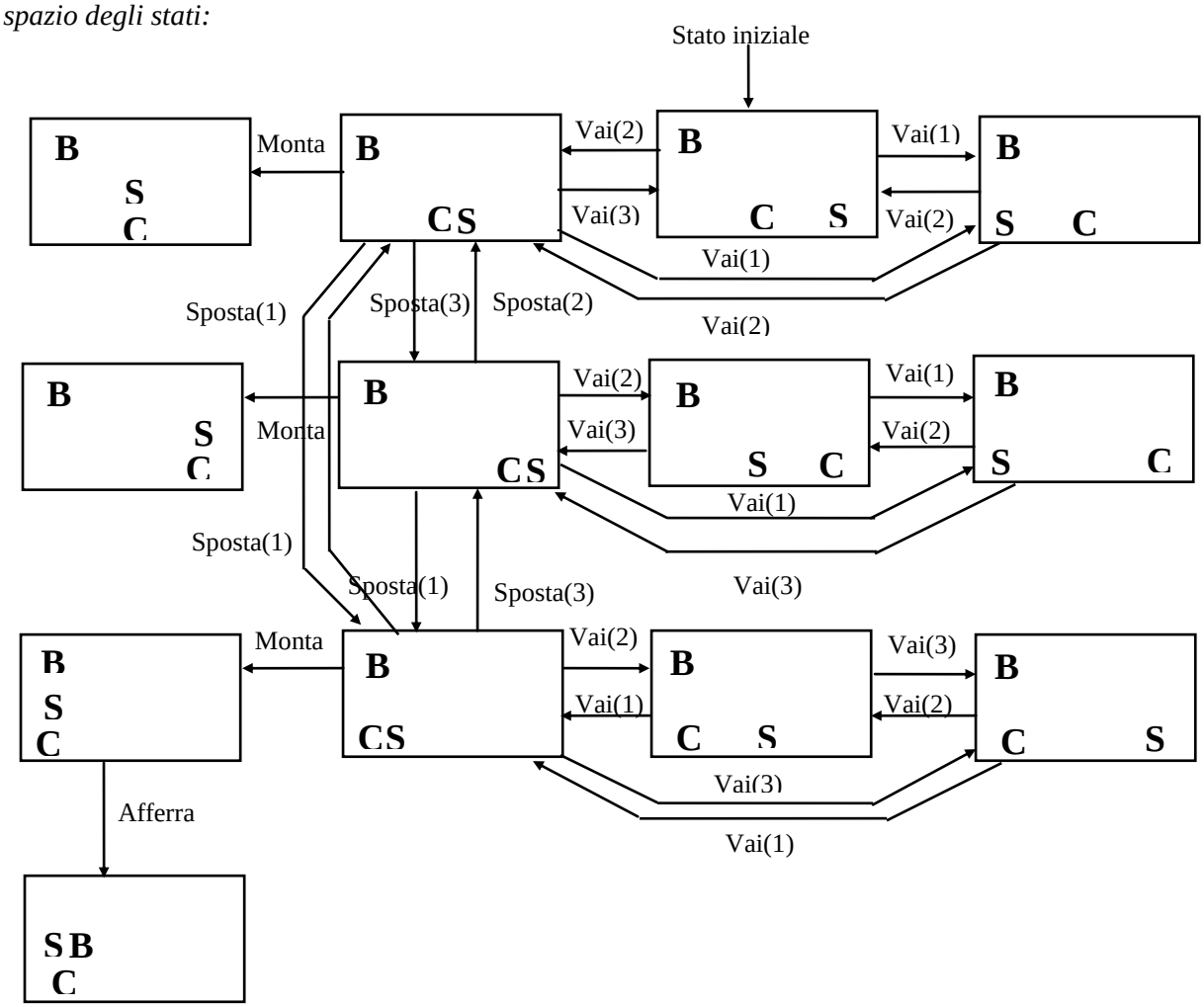


In questo modo esistono 13 possibili stati diversi (assumendo che non ci interessano gli stati dopo che la scimmia ha afferrato la banana).

Gli operatori: Vai, Sposta, Monta, Scendi, Afferra.
Gli operatori hanno delle precondizioni per la loro applicabilità e degli effetti:
Vai(y) - *Precondizioni*: S in x, x≠y - *Effetti*: S in y

Sposta(y) - *Precondizioni*: S in x, C in x, , x≠y - *Effetti*: S in y, C in y
Monta - *Precondizioni*: S in x, C in x, non SuCassa - *Effetti*: SuCassa
Scendi - *Precondizioni*: S in x, C in x, SuCassa - *Effetti*: non SuCassa
Mangia - *Precondizioni*: SuCassa, C in 1 - *Effetti*: HaBanana
Il costo del cammino: 1 per ogni azione della scimmia

Lo spazio degli stati:



b) Procedere all'indietro dal goal o in avanti dallo stato iniziale è indifferente visto che ogni mossa ha la sua mossa inversa e quindi il fattore di diramazione è lo stesso nelle due direzioni. È importante il controllo sugli stati ripetuti che può essere semplicemente quello di non visitare nodi già visitati nel cammino dalla radice dell'albero di ricerca allo stato corrente.
Una strategia di ricerca che trova il cammino ottimale è quella in ampiezza, che trova la soluzione a profondità 4 (Vai(2), Sposta(1), Monta, Afferra); può funzionare anche una ricerca in profondità con limite di profondità dato dal numero di stati possibili (13) ma in questo caso per trovare la soluzione minima sarebbe necessario trovare tutte le soluzioni e scegliere la minima (magari evitando di generare soluzioni che comportino un numero di mosse maggiori della soluzione migliore trovata fino al quel punto).

c) *Se non interessasse l'ottimalità (basta che la scimmia prenda le banane, non importa se non lo fa con il minimo numero di mosse) una visita in profondità con limite di profondità 13, in cui ci si ferma alla prima soluzione è migliore di una visita in ampiezza perché più efficiente nel tempo e nell'occupazione di spazio.*

Avanti, indietro o bidirezionale?

Dire per ciascuno dei seguenti problemi, in che direzione conviene esplorare lo spazio di ricerca (in avanti, all'indietro o ricerca bidirezionale?). Motivare la risposta.

a. Tom è un discendente di Paul Revere?
b. Integrazione simbolica (risoluzione di un integrale con tecniche simboliche)
c. Diagnosi medica.

I missionari e i cannibali

Tre missionari e tre cannibali si trovano sulla riva est di un fiume; c'è una barca che può trasportare fino a due persone. Trovare un modo per far arrivare tutti i personaggi sulla sponda ovest del fiume sani e salvi, tenendo conto del fatto che non è possibile, per la incolumità dei missionari, lasciare in un posto più cannibali che missionari.

***Stati**: sequenza ordinata di 3 numeri che rappresentano il numero di missionari, cannibali. e la posizione della barca sulla riva est (la situazione sulla riva ovest può essere ottenuta per differenza).*

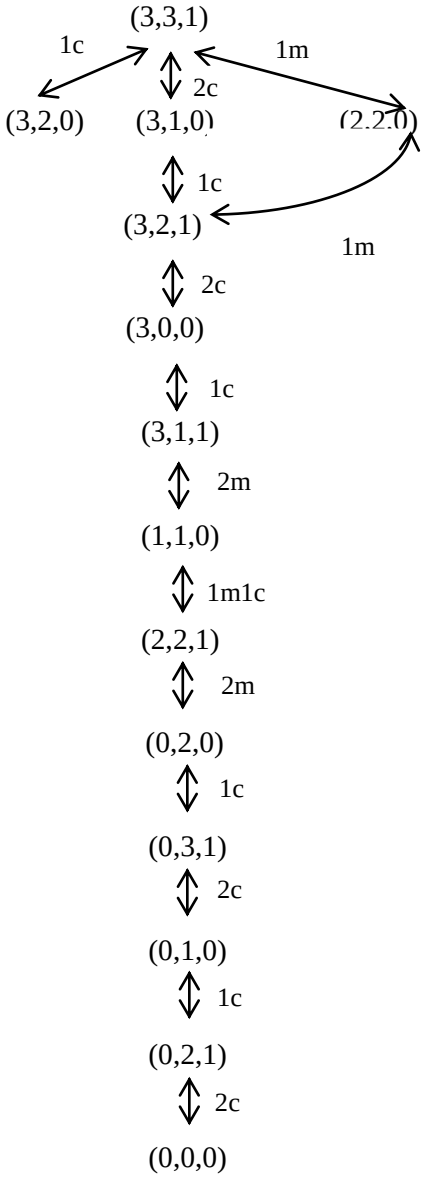
***Stato iniziale**: (3,3,1)*

***Operatori**: prendere o un missionario, un cannibale, due missionari, due cannibali, o uno di entrambi per attraversare il fiume. Avrò 5 operatori:*
a) 1 missionario
b) 1 cannibale
c) 2 missionari
d) 2 cannibali
e) 1 missionario + 1 cannibale
Nel fare la mossa controllerò di evitare gli stati illegali, cioè quelli con più cannibali che missionari.

***Goal test**: stato (0,0,0) cioè non ci sono né missionari, né cannibali sulla sponda di partenza.*

***Path cost**: contando 1 il costo di un attraversamento, il costo di un cammino è pari al numero di attraversamenti.*

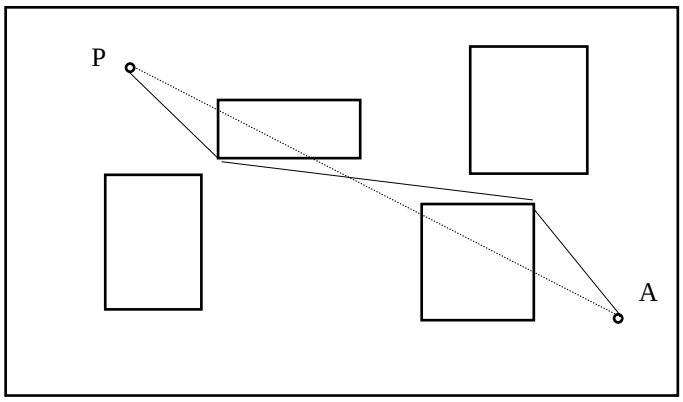
Grafo degli stati:



Algoritmi di ricerca euristica

Navigazione di un robot

Si tratta di pianificare per il robot il percorso minimo tra due punti dati in una stanza popolata di ostacoli a forma di poligoni convessi del tipo di quella mostrata in figura.



Una riduzione drastica dello spazio degli stati si ha considerando solo i punti del piano che corrispondono ai vertici dei poligoni, più i due punti di partenza e di arrivo, e considerando spostamenti in linea retta tra questi punti.

a) spiegare perché questa semplificazione non fa perdere l'ottimalità
b) definire lo spazio degli stati e gli operatori
c) dire quale strategia di ricerca usereste.

Soluzione

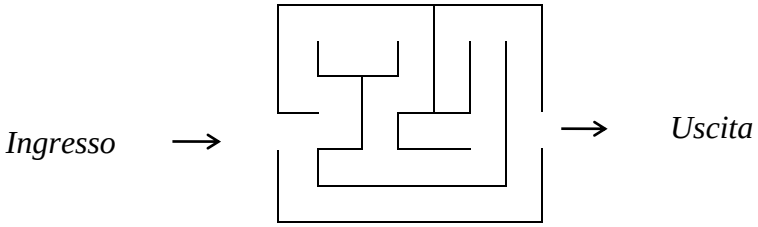
a) Un modo di vedere le cose è che la linea retta tratteggiata da P ad A, distanza in linea d'aria tra i due punti, è in assoluto il percorso minimo, ma tipicamente non percorribile per la presenza di ostacoli. La minima deviazione da questo percorso passa per qualche vertice di poligono. È come se la linea tratteggiata fosse un elastico che allunghiamo il minimo indispensabile per aggirare gli ostacoli (vedi figura, dove la linea solida rappresenta l'elastico allungato).

b) Lo spazio degli stati è dato dai vertici dei poligoni più i due punti di partenza e di arrivo. Gli operatori sono gli spostamenti in linea retta da un punto ai successivi visibili (se sono visibili possono essere raggiunti con spostamenti in linea retta).

c) Il problema è del tutto equivalente a quello del route finding *e può pertanto essere risolto con un algoritmo di tipo A* con funzione di valutazione f(s) =g(s) + h(s)*
dove la g è la somma delle lunghezze dei tratti percorsi per arrivare nel punto del piano che corrisponde allo stato n e la h la distanza in linea d'aria dal punto in cui si è e il punto di arrivo A; questa è chiaramente una sotto-stima per la presenza di ostacoli visto che il robot non vola. Quindi la soluzione trovata dall'algoritmo è una soluzione minimale.

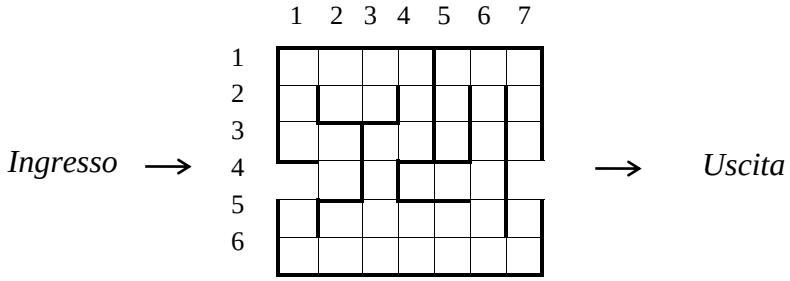
Il labirinto

Dato il seguente labirinto



a. formalizzare il problema di trovare un percorso dall'ingresso all'uscita come un problema di ricerca in uno spazio di stati;
b. descrivere un algoritmo di tipo A* che trova la soluzione ottimale;
c. mostrare come si comporta l'algoritmo.

Soluzione. *Possiamo rappresentare il labirinto su una griglia:*



Indichiamo una posizione nel labirinto con una coppia di coordinate (righe, colonne).

Stati*: una coppia di coordinate (x,y) appartenenti al labirinto*

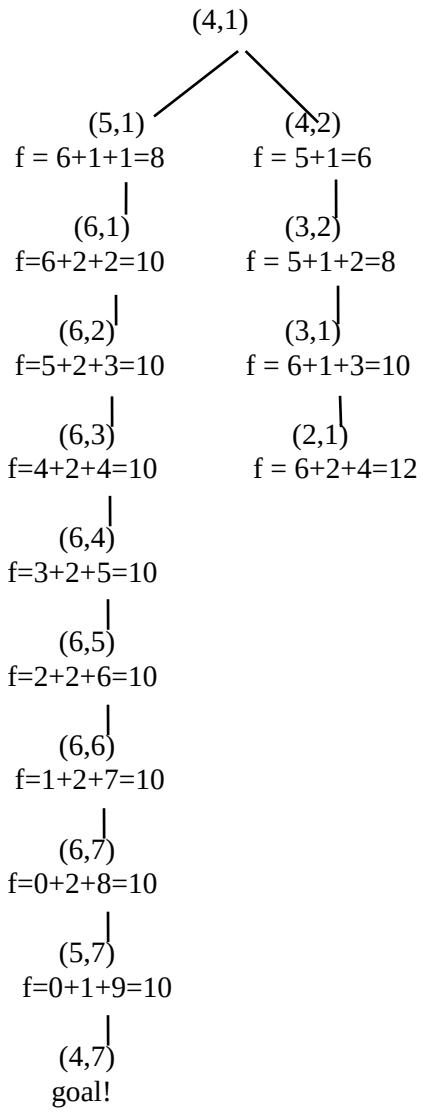
***Stato iniziale**: Ingresso in (4,1)*

***Goal State**: Uscita in (4,7)*

***Operatori**: posso andare su, giù a destra o a sinistra dove non c'è il muro.*
Ad esempio, dallo stato (4,1) possiamo andare agli stati (5,1), (4,2)

Algoritmo A: è un algoritmo Best First con euristica h ammissibile e cioè h(n) è una sottostima rispetto alla valutazione h*(n).*
f(n) = h(n) + g(n)
dove h(n) stima del costo per raggiungere l'uscita, g(n) costo del cammino percorso. Possiamo prendere come h(n) la Manhattan distance tra lo stato n e lo stato goal e come g(n) il numero di caselle che l'agente ha percorso.Potremmo considerare come h(n) anche la distanza in linea d'aria tra lo stato n e l'uscita, ma la Manhattan distance domina *la distanza in linea d'aria pur rimanendo una sottostima. Preferiamo quindi scegliere questa come stima. La Manhattan distance tra due punti (x1,y1), (x2,y2) è data dalla formula*
$M = |y2 - y1| + |x2 - x1|$

Sull'esempio con goal state (4,7) l'algoritmo A genera il seguente albero di ricerca*

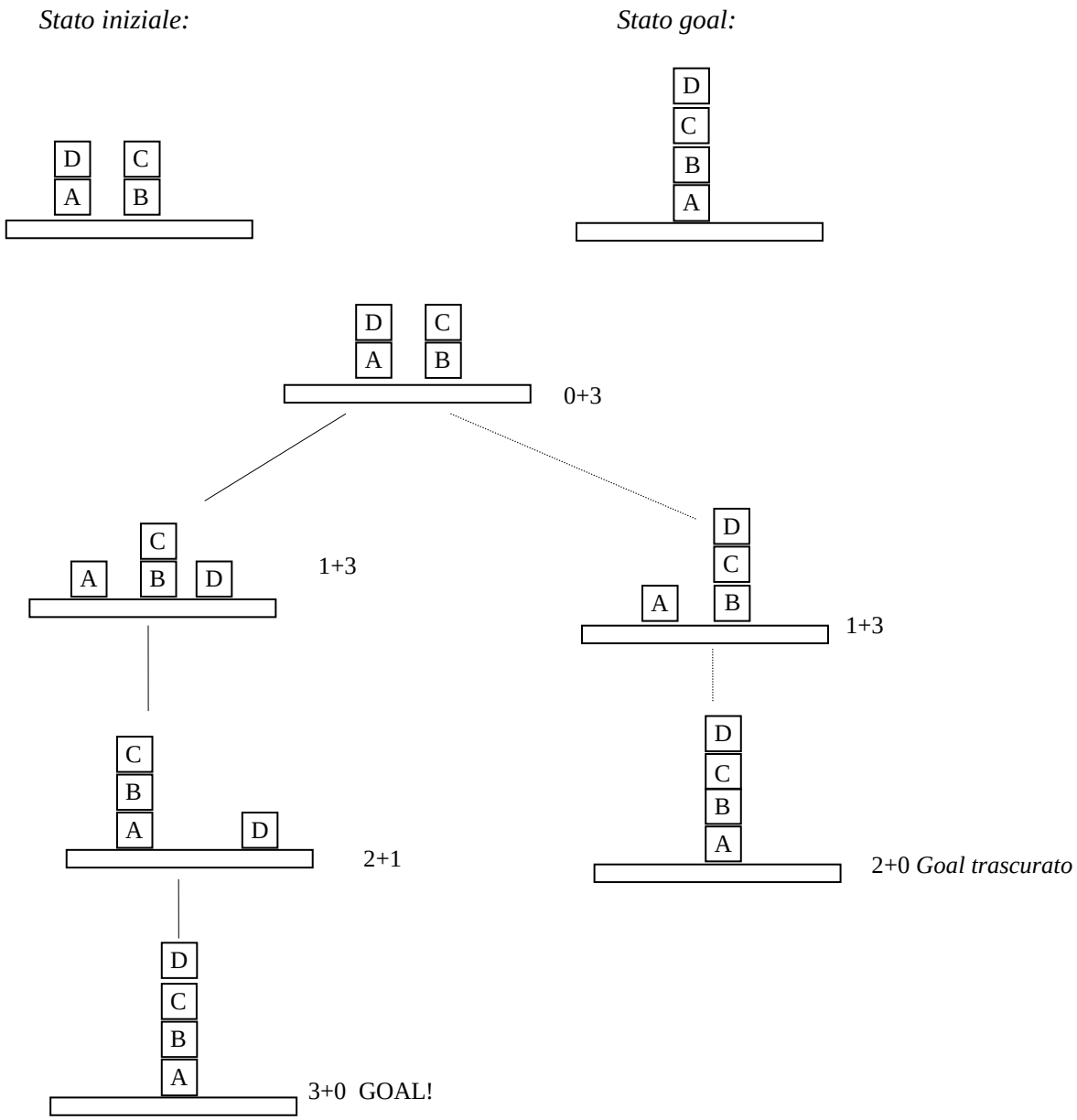


La soluzione trovata è ottimale perché è il percorso minimo tra l'ingresso e l'uscita del labirinto.

Il Mondo dei Blocchi con sovrastima

Inventarsi una euristica *h* per il mondo dei blocchi che qualche volta sovrastima la distanza dello stato goal e mostrare con un esempio che questa euristica, usata con un algoritmo di ricerca A*, può portare ad una soluzione non ottimale.

Per risolvere questo esercizio conviene pensare ad una formulazione leggermente diversa del problema del mondo dei blocchi in cui gli operatori sono tali che consentono di raggiungere la configurazione finale con cammini disgiunti (senza stati in comune). Per esempio potremmo consentire anche lo spostamento di torri di blocchi. In questa ipotesi, e con una semplice funzione di valutazione che misura il numero di blocchi fuori posto (un blocco è fuori posto a meno che tutta la struttura sottostante non sia corretta) , un controesempio è il seguente:



L'algoritmo A trova la soluzione a profondità 3 e non quella a profondità 2.

Il gioco dell'otto con sovrastima

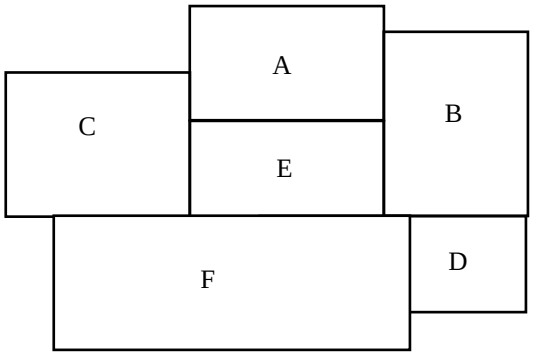
Dare un'euristica per il gioco dell'otto che sovrastima in qualche caso la distanza della soluzione. Dire se l'algoritmo A con la vostra euristica

a. trova sempre una soluzione;
b. trova sempre una soluzione ottimale.

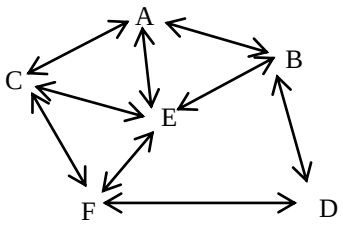
Colorazione di una mappa come CSP

Data una mappa del tipo qui raffigurato colorare la mappa con tre colori (rosso, verde, blu) evitando di colorare due paesi adiacenti con lo stesso colore.

Esempio di mappa:



Grafo dei vincoli



Gli stati: *mappe parzialmente colorate ((A blu)(B verde) …*)
Stato iniziale: ( )
Goal_test: *mappa colorata senza violazione di vincoli*
Path_cost*: non importa*
Operatori: *colora nuovo paese*

L'ampiezza dello spazio degli stati è : $4\times4\times \ldots\times4 = 4^{\#paesi}$
Conviene vedere il problema come un problema di soddisfacimento di vincoli ed utilizzare le euristiche e tecniche consigliate per questo tipo di problemi.
Le variabili sono: A,B,C,D,E
Il dominio delle variabili è {rosso,verde,blu} per tutte
I vincoli sono di tipo binario e sono del tipo A≠B.

In questo caso conviene usare una euristiche del tipo

- *Assegnare prima la variabile con meno valori residui (MRV)*
- *A parità dei domini assegnare prima la variabile più vincolante, coinvolta in più vincoli con le altre variabili (euristica del grado)*
- *Usare forward checking per propagare i vincoli*

A : {rosso, verde, blu} B : {rosso, verde, blu} C : {rosso, verde, blu} D : {rosso, verde, blu} E : {rosso, verde, blu} F : {rosso, verde, blu}	A : {verde, blu} B : {verde, blu} C : {verde, blu} D : {rosso, verde, blu} **E : rosso** F : {verde, blu}	A : {verde, blu} B : {verde, blu} C : {blu} D : {rosso, blu} **E : rosso** **F : verde**
A : {verde} B : {verde, blu} **C : blu** D : {rosso, blu} **E : rosso** **F : verde**	**A : verde** B : {blu} **C : blu** D : {rosso, blu} **E : rosso** **F : verde**	**A : verde** **B : blu** **C : blu** D : {rosso} **E : rosso** **F : verde**

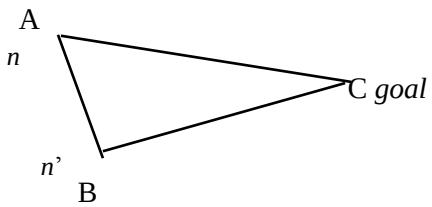
Sull'esempio si sceglie E all'inizio perchè questa variabile è coinvolta in più vincoli con le altre variabili (grado 4) e assegniamo ad E Rosso. Propagando i vincoli escludiamo Rosso per A,B,C,F.
A questo puntoné MRV né il grado ci aiutano, scegliamo una delle rimanenti variabili a caso, esempio F, e assegniamo ad essa uno dei restanti colori, esempio Verde; questa scelta esclude il Verde per C, che deve essere necessariamente Blu. Nota che con Forward Checking (FC) non ci si accorge che A deve essere necessariamente Verde. Questo perchè FC interessa solo le variabili collegate direttamente alla variabile assegnata.
A questo punto si sceglie un valore per C, Blu, in base all'euristica MRV. Di conseguenza FC escludeBlu per A. Al prossimo passo viene scelto Verde per A, in base a MRV, ed escluso Verde per B.
Successivamente Blu per B, ed escluso Blu per D, che quindi è Rosso.

Dire che cosa succede se invece delle euristiche precedenti si usa in controllo di consistenza degli archi.

Diseguaglianza triangolare

Si dice che la componente *h* di una funzione euristica soddisfa la diseguaglianza trangolare quando dati tre punti A B e C la stima del costo da A a C non supera il costo effettivo da A a B più la stima del costo da B a C.

a. mostrare un esempio in cui la *h* ha questa caratteristica (può essere uno di quelli visti a lezione)
b. dimostrare che una funzione euristica con una h di questo tipo è monotona.



a) distanza in linea d'aria, come nel problema del route finding;
b) Consideriamo il seguente caso: sia A il nodo n, B lo stato successore n' e sia C lo stato goal.
Allora, per la diseguaglianza triangolare, abbiamo che g(n') - g(n) + h (n') - h(goal) >= h(n) - h(goal); poichè h(goal)=0 abbiamo: g(n') - g(n) + h(n') >= h(n); e cioè g(n') - g(n) >= h(n) - h(n') che è la condizione di monotonia.

Un'euristica più informata è più efficiente

Sia A_1 un algoritmo A* con funzione di valutazione euristica $f_1=g+h_1$. Sia A_2 un algoritmo A* con funzione di valutazione euristica $f_2=g+h_2$ dove h_2 è un'euristica più informata rispetto ad h_1.

a) Si giustifichi il più formalmente possibile il fatto che A_2 visita non più nodi di A_1.
b) Questo implica che A_2 è in genere più efficiente di A_1? Spiegare perché.

Piastrelle bianche e nere

È un gioco dell'otto semplificato. Sia data una scacchiera con 5 caselle, di cui due occupate da piastrelle nere (N), due occupate da piastrelle bianche (B) e una vuota (0). Le mosse consentite sono:

- una piastrella può spostarsi in una casella vuota adiacente; il costo di questa mossa è 1.
- una piastrella può saltare sopra al più due caselle in una casella vuota; il costo di questa mossa è pari al numero di caselle saltate.

Data la seguente configurazione iniziale:

N	N	B	B	0

l'obiettivo è quello di raggiungere una configurazione in cui tutte le piastrelle bianche sono a sinistra delle nere, indipendentemente dalla posizione della casella vuota.

a. Definire una funzione euristica *h* per questo problema e mostrare il comportamento dell'algoritmo A con questa euristica.

b. Dire se *h* è un'euristica ammissibile e se soddisfa la restrizione di monotonia, motivando la risposta.

La scacchiera di due colori

Colorare le celle di una griglia 3x3 con due colori (R e B) in modo da minimizzare il numero di celle adiacenti dello stesso colore.

a. A partire da una configurazione iniziale, con i colori disposti casualmente, e avendo a diposizione un operatore che inverte il colore di una cella, si trovi un'euristica ammissibile da utilizzare in connessione con un algoritmo di tipo A*.
b. Data una funzione di valutazione che conta il numero di coppie di caselle adiacenti dello stesso colore, si tracci l'andamento dell'algoritmo di *hill-climbing* (o meglio *hill-descending*) a partire dalla seguente configurazione iniziale:

R R R
B R R
B B B

c. L'algoritmo trova sempre la soluzione, qualunque sia la configurazione iniziale ?
d. L'algoritmo trova sempre la soluzione di costo minimo (ottenibile nel minor numero di mosse)?

a. *Si nota che esistono due sole configurazioni obiettivo:*

C1 = *R B R* / *B R B* / *R B R* *C2 =* *B R B* / *R B R* / *B R B*

Cioè le due possibili scacchiere (ottenibili una dall'altra invertendo i colori). Sia Δ(Ci,Cj) il numero di caselle differenti fra le configurazioni Ci e Cj. A partire da una qualsiasi configurazione Ci, è possibile ottenere C1 in Δ(Ci,C1) e C2 in Δ(Ci,C2) passi. La funzione

$$h(Ci) = Min\{\Delta(Ci,C1), \Delta(Ci,C2)\}$$

è sempre positiva e nulla in una configurazione goal, quindi è una stima della distanza dalla soluzione. Inoltre, essendo una stima esatta è un oracolo (chiaramente ammissibile).

La funzione di valutazione del punto b. invece, non poteva essere utilizzata come euristica ammissibile, per esempio in

B ***R*** *B*
R R R
B ***R*** *B*

vale 4 (numero delle coppie evidenziate), mentre la soluzione dista 1 (basta effettuare la mossa che inverte la casella centrale).

Altre soluzioni consistono nel considerare ogni mossa di costo 4, oppure di dividere la funzione del punto b per 8, 5, o 4; si ottengono così delle euristiche ammissibili ma talvolta molto poco informate.

b. *L'esercizio richiede di tracciare il comportamento dell'algoritmo a partire dalla configurazione data, in modo da poter giustificare ad ogni passo la scelta dello stato successivo.*

R R R
B R R
B B B

B R R R B R R R B R R R R R R R R R R R R R R R R R R
B R R B R R B R R R R R B B R B R B B R R B R R B R R

B B B B B B B B B B B B B B B B B B R B B B R B B B R

*8 **5** 6 9 8 7 6 7 8*

Lo stato successivo scelto corrisponde alla configurazione con valutazione migliore: la seconda da sinistra con valutazione 5. Iterando per altre 3 volte la generazione dei possibili stati successivi e la scelta dello stato migliore, si ottiene una soluzione ottima passando per stati con valutazione 3, 2 e 0.

***C.** L'algoritmo è completo.*
(Questo punto richiede una dimostrazione un po'articolata. Anche spiegazioni intuitive sono state valutate positivamente).

La prima osservazione consiste nel notare che in generale ad ogni passo la funzione di valutazione decresce. Quindi ci si potrebbe aspettare che l'algoritmo converga in ogni caso ad una soluzione ottima. Purtroppo, per alcune configurazioni si nota che il migliore stato successivo ha la stessa valutazione del padre, come ad esempio:

Z=
R B R
B R B
B R B
3

che ha due migliori stati successivi di costo 3:

A=
R B R
*B **R B***
R R B

B=
R B R
***B R** B*
B R R

Si potrebbe pensare che l'algoritmo possa incappare in un pianoro, ma questo non blocca la ricerca se solo si ammette di poter passare ad uno stato con una valutazione uguale a quella del predecessore. Allora si potrebbe pensare che l'algoritmo possa ciclare, ma affinché così sia anche il miglior successore di A o B dovrebbe continuare ad avere una valutazione uguale a 3: in tal caso Z, il predecessore di A (o B), che è anche un suo successore (invertendo la stessa casella che da Z porta ad A), potrebbe essere nuovamente selezionato (ciclo). Intuitivamente – per tentativi – ci si rende conto che tutti i successori di A (o B) migliorano la funzione di valutazione. Alternativamente, ciò può essere dimostrato ragionando sul numero di celle che l'inversione di una casella può influenzare a seconda della sua posizione sulla scacchiera. Si nota che solo l'inversione di celle con un numero pari di adiacenti può produrre stati successori con la stessa valutazione (A e B nel nostro caso). Ragionando per casi, si riesce a dimostrare che necessariamente tali stati hanno almeno un successore che migliora la funzione di valutazione.

***e.** L'algoritmo non è ottimale. Per esempio partendo dalla configurazione*

*R **R** R*
R R R
*R **R** R*

12

la mossa che migliora maggiormente la funzione di valutazione è l'inversione del colore della casella centrale, che elimina 4 coppie di caselle adiacenti con lo stesso colore, ottenendo lo stato

R R R
R B R
R R R

8

Da questa configurazione l'algoritmo procede verso la soluzione C2 raggiungendola in 5 passi complessivi (invertendo cioè anche le 4 caselle in angolo), mentre la soluzione C1 poteva essere raggiunta in soli 4 passi.

Euristica di Gasdching

Si pensi ad un gioco dell'otto in cui A può essere spostata in B, se B è vuota. Sia Hg una stima esatta della distanza della soluzione minima per questo gioco "meno vincolato".

a. Hg (detta l'euristica di Gasching) è ammissibile per il gioco dell'otto tradizionale?
b. Hg è un'euristica migliore (più informata) rispetto ad H1 (l'euristica che conta il numero di caselle fuori posto) per il gioco dell'otto tradizionale?
c. Esistono casi in cui Hg è più accurata di H2 (l'euristica che somma le distanze Manhattan delle caselle dalla loro collocazione finale)?

a. Hg è ammissibile per il gioco dell'otto tradizionale in quanto tutte le mosse possibili nel gioco dell'otto tradizionale sono anche possibili nel gioco dell'otto meno vincolato. Quindi tipicamente ci vogliono più mosse e nel caso migliore ce ne vuole lo stesso numero.

b. Per ogni n H1(n) ≤ Hg(n). Infatti le caselle fuori posto possono essere messe a posto con una o due mosse: una se la casella di destinazione è vuota, due se va svuotata prima di metterla a posto (non sempre la mossa di svuotamento può servire a mettere a posto un'altra casella).

c. Si, esistono casi in cui Hg è più accurata di H2. Si consideri il seguente esempio:

2	1	3
4	5	6
7	8	

1	2	3
4	5	6
7	8	

H1=2 le due caselle fuori posto sono la 1 e la 2
Hg=3 per mettere le cose a posto nel gioco meno vincolato servono 3 mosse

Il cavallo

Supponiamo una scacchiera infinita ed un cavallo nella posizione iniziale (0,0); sia data inoltre una posizione (*m*, *n*) di arrivo per il cavallo. Il problema consiste nello spostare il cavallo dalla posizione iniziale alla posizione di arrivo con il numero minimo di mosse legali (la figura 1 mostra le mosse legali per il cavallo in un caso particolare).

a. si formuli il problema come un problema di ricerca in uno spazio di stati;
b. si trovi una euristica ammissibile, il più possibile informata, per il problema;
c. si trovi una soluzione tracciando l'andamento dell'algoritmo A* sul problema della figura 2 (obiettivo in (5, 4)

0	1	2	3	4	5	6	7	8	9
1									
2			×		×				
3		×				×			
4				C					
5		×				×			
6			×		×				
7									
8									
9									

0	1	2	3	4	5	6	7	8	9
1	C								
2									
3									
4									
5				G					
6									
7									
8									
9									

Il puzzle di Snape

Uno dei puzzle logici più famosi nella letteratura recente è quello escogitato dal prof, Severus Snape per proteggere la pietra filosofale. Questo puzzle, risolto da Hermione Granger (amica di Harry Potter), consiste nell'individuare due pozioni (una che consente di avanzare tra le fiamme e una per tornare indietro sani e salvi) tra una serie di sette bottiglie dal contenuto sconosciuto allineate sul tavolo. Il prof. Snape ha fornito una serie di indizi sotto forma di un poema. Per aiutarvi riportiamo sulla destra gli indizi fondamentali e la collocazione delle bottiglie così come si presume siano apparse ad Hermione.

Danger lies before you, while safety lies behind,
Two of us will help you, whichever you would find,
One among us seven will let you move ahead,
Another will transport the drinker back instead,
Two among our number hold only nettle-wine,
Three of us are killers, waiting hidden in line
Choose, unless you wish to stay here forevermore
To help you in your choice, we give you these clues four:
First, however slyly the poison tries to hide
You will always find some on nettle wine's left side
Second, different are those who stand at either end
But if you would move onward, neither is your friend;
Third as you see clearly, all are different size
Neither dwarf nor giant hold death in their insides;
Fourth, the second left and the second on the right
Are twins once you taste them, though different at first sight.

Una pozione serve per avanzare, l'altra per tornare indietro.

Due bottiglie contengono vino d'ortica, tre contengono veleno.

Gli indizi sono quattro:

1. C'è sempre del veleno a sinistra del vino d'ortica.
2. Le bottiglie alle estremità hanno contenuti diversi, ma nessuna di queste serve per andare avanti.
3. Né la bottiglietta più piccola, né quella gigante contengono veleno.
4. La seconda da sinistra e la seconda da destra hanno lo stesso contenuto.

Nell'interpretare i vincoli potete far riferimento alla figura a lato.

Formulare come problema di soddisfacimento di vincoli e risolvere con una delle euristiche viste a lezione dichiarando quale state usando e illustrando il procedimento passo per passo.

Suppongo di numerare le bottiglie con 1, 2 … 7 da sinistra a destra
Variabili: V1, V2 … V7
Dom(Vi) = {A, I, O, V} dove A=avanti, I=indietro, O=vino d'ortica, V=veleno
Vincoli:

1. *Vi = O => Vi-1 = V (i=2, … 7)*
2. *V1≠V7; V1≠A; V7≠A;*

3. *V4≠V; V6≠V*
4. *V2=V6*

Uso l'euristica MRV e Forward checking

V1={I, V} V2={A, I, O, V} V3={A, I, O, V} V4={A, I, O} V5={A, I, O, V} V6={A, I, O} V7= {I, O, V}	V1=V V2={A, I, O, V} V3={A, I, O, V} V4={A, I, O} V5={A, I, O, V} V6={A, I, O} V7= {I, O}	V1=V V2={A, O, V} V3={A, O, V} V4={A, O} V5={A, O, V} V6={A, O} V7= I	V1=V V2={ } V3={O, V} V4={O} V5={O, V} V6=A V7=I BACK
V1=V V2={O} V3={A, O, V} V4={A, O} V5={V} V6=O V7= I	V1=V V2=O V3={A, O, V} V4={A, O} V5=V V6=O V7= I	V1=V V2=O V3=V V4=A V5=V V6=O V7= I	

Horizon search

Supponiamo che l'agente non sia in grado, a causa di limitazioni nelle risorse di calcolo, o che non sia opportuno, perché l'ambiente è dinamico e può cambiare, pianificare l'intera sequenza di azioni per raggiungere uno stato obiettivo.
Supponiamo inoltre che l'agente disponga di una buona funzione di valutazione euristica monotona crescente *f* da applicare agli stati. Si potrebbe espandere l'albero di ricerca fino ad una profondità prefissata *d*, applicare agli stati a profondità *d* la funzione di valutazione e vedere tra questi quale è lo stato più promettente, cioè quello con valore di *f* minore.

a. Prendendo spunto dagli algorimi visti a lezione, scrivere un algoritmo di ricerca in profondità che restituisca il nodo più promettente a profondità *d* (nel senso detto sopra).
b. Questa ricerca potrebbe essere ottimizzata con una tecnica di potatura: modificare l'algoritmo introducendo questa ottimizzazione.

Versione semplice

```
function Horizon-Search (problem, limit) returns a solution node or failure
  return RHS(Make-Node(Initial-State[problem]), problem, limit)

function RHS(node, problem, limit)
  if (Goal-Test[problem](State[node]) or Depth[node]=limit) then return node
    else
      successors ← Expand(node, problem);
      if successors is empty then return failure
      else
         v ← ∞;
         for each s  in successors do
             n ← RHS(s, problem, limit)
             if n != failure and  f(n) ≤ v then v ← f(n); bestNode ← n;
         end
      return bestNode
```

Versione ottimizzata

```
function Horizon-Search (problem, limit) returns a solution node or failure
  return RHS(Make-Node(Initial-State[problem]), problem, limit, ∞)

function RHS(node, problem, limit, best)
  if (Goal-Test[problem](State[node]) or Depth[node]=limit) then return node
    else
     if f(node) > best then return failure
     else
     successors ← Expand(node, problem);
     if successors is empty then return failure
     else
        v ← best;
         for each s  in successors do
             n ← RHS(s, problem, limit, v)
             if n != failure and  f(n) < v then v ← f(n); bestNode ← n;
         end
      return bestNode
```

Giochi con avversario

Alfa-beta applicato al filetto

Nel gioco del filetto (tic-tac-toe) supponiamo di usare la seguente funzione di valutazione euristica:

$$f(n) = 3X_2 + X_1 - (3O_2 + O_1)$$

dove Xn = numero di righe, colonne, diagonali con n X e nessuna O
On = numero di righe, colonne, diagonali con n O e nessuna X

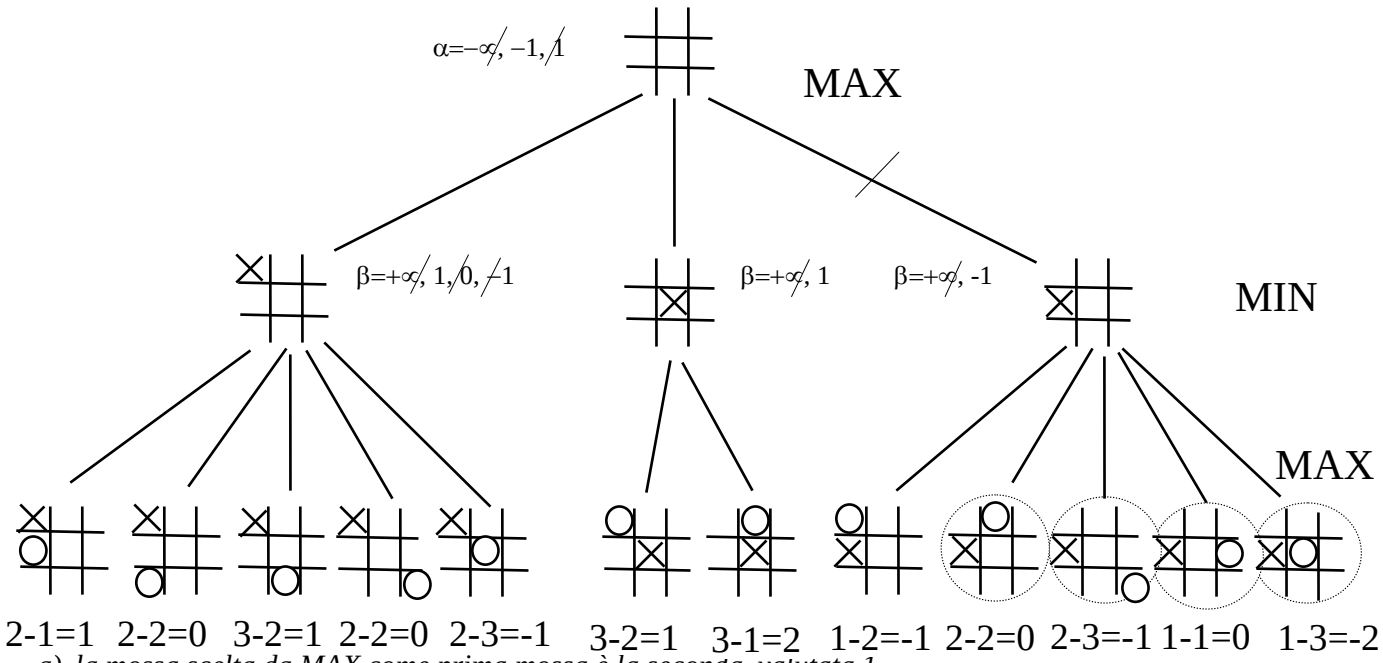
a) Espandere 2 livelli a partire dallo stato iniziale, valutare gli stati risultanti, propagare all'indietro con la regola del MIN e MAX e scegliere la mossa migliore.

b) evidenziare cerchiandoli i nodi a livello 2 che non sarebbero valutati se si usasse la tecnica di potatura alfa-beta
c) Lo stesso del punto b) dopo avere riordinato i successori in modo che siano nell'ordine ottimale per la potatura alfa-beta.

NOTA: la funzione di utilità potrebbe essere:

+1 *se* $X_3=1$
- 1 *se* $O_3=1$

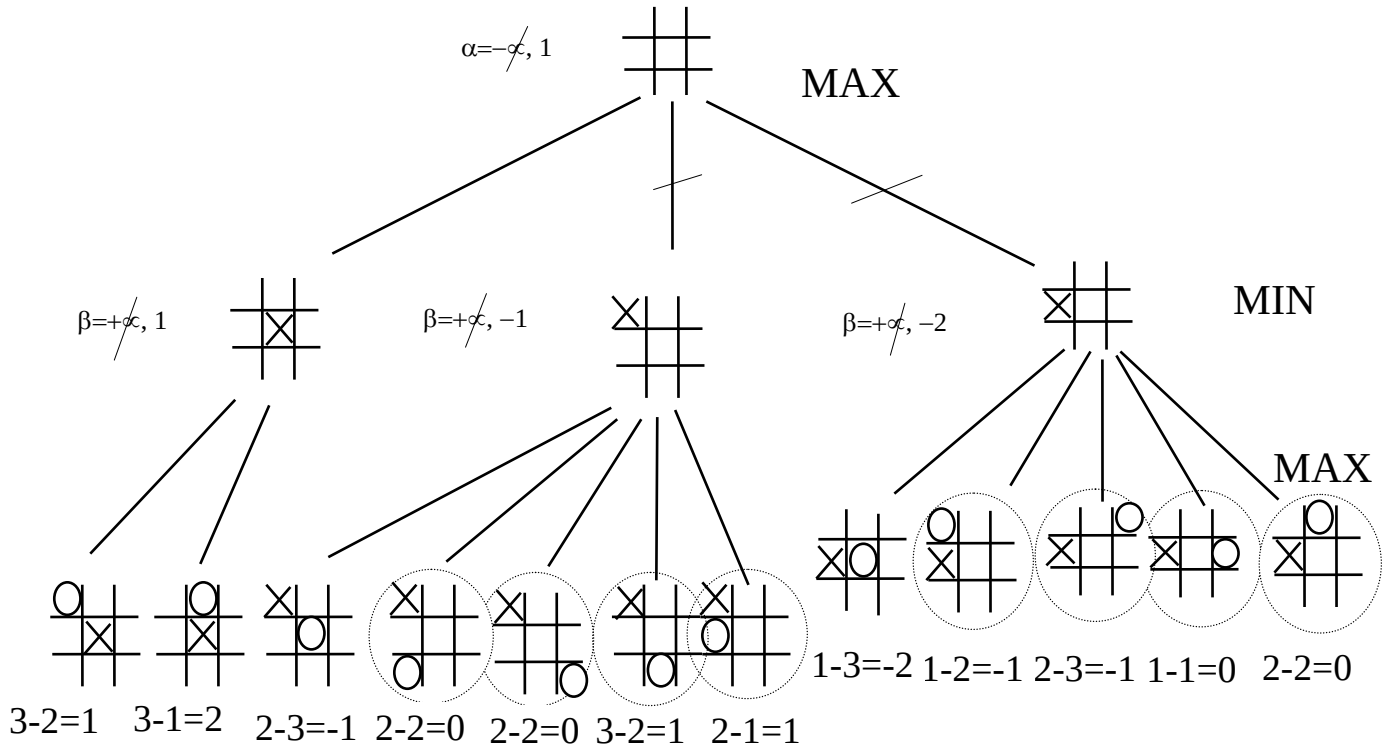
e quindi in accordo con la funzione di valutazione in quanto per vincere si devono prima mettere due simboli X in fila e questi stati sono valutati meglio degli altri.



a) la mossa scelta da MAX come prima mossa è la seconda, valutata 1.

b) gli stati non visitati con la potatura alfa-beta sono quelli cerchiati.

c) la seguente è la situazione dopo un opportuno riordinamento dei nodi in modo da massimizzare l'effetto della potatura alfa-beta.

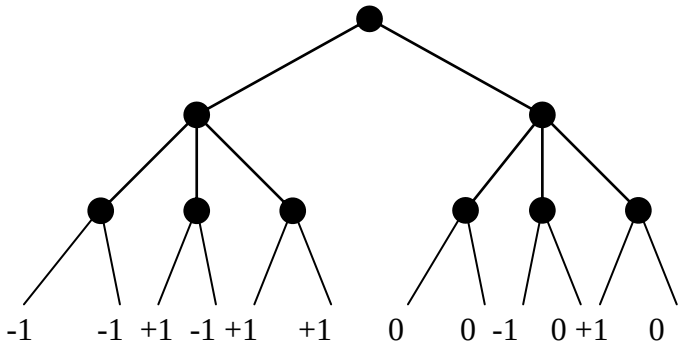


NEG-MAX

Un'elegante semplificazione della procedura MIN-MAX è stata proposta da D. Knuth e R.W. Moore nel 1975. La procedura, della NEG-MAX, usa a tutti i livelli la stessa funzione di valutazione così definita:

F(*j*)=*f*(*j*), dove *f* è una funzione di valutazione euristica dello stato per i nodi terminali, dal punto di vista dell'avversario di chi muove negli stati terminali. Se è A a muovere, la valutazione dovrà essere dal punto di vista dell'avversario B, in particolare dare un punteggio alto a situazioni perdenti per A e basso a situazioni vincenti per A.

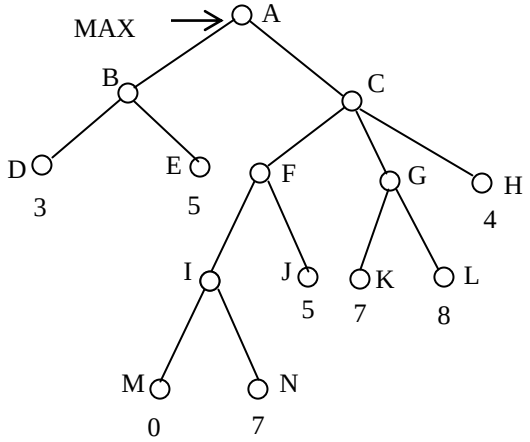
F(*j*)=MAX{– F(i_1), – F(i_1), …, – F(i_1)}, con i_1, i_2, …, i_n successori di j, per i nodi non terminali.



a. Sul seguente esempio, dati i valori della *f* sugli stati terminali, assegnare un valore ai nodi non terminali con la regola del NEG-MAX.
b. Dire se, o sotto quale ipotesi, l'euristica che asd ogni passo sceglie il massimo dei valori negati dei successori coincide con quella usata da MIN-MAX, motivando la risposta.

MIN MAX con ordini diversi di visita

Sia dato il seguente esempio di gioco con avversario in cui si assume una funzione di valutazione degli stati terminali che valuta le foglie come indicato in figura 3.



a. Tracciare l'esecuzione dell'algoritmo MIN-MAX, evidenziare le valutazioni intermedie dei nodi e dire quale mossa farebbe MAX.
b. Tracciare l'esecuzione dell'algoritmo con potatura α-β con visita dell'albero da sinistra a destra (evidenziando le potature). Tracciare l'esecuzione di α-β con visita dell'albero da destra a sinistra. La mossa scelta è la stessa nei due casi? I nodi non visitati sono gli stessi nei due casi? Perché?