II esercitazione

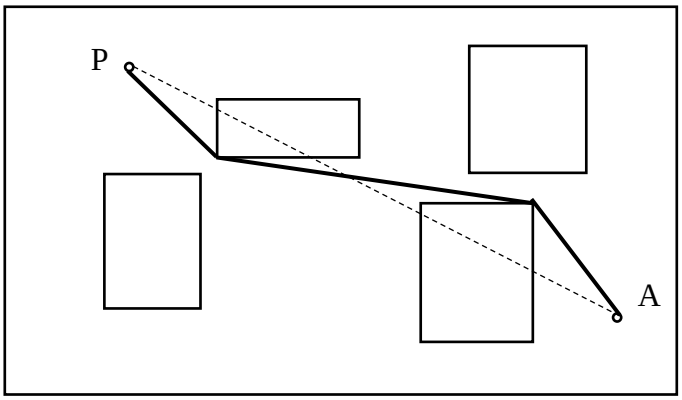
Nell'esercitazione ci concentreremo sugli esercizi con asterisco, ma lasciamo gli altri come esercizi.

Formulazione di Problemi

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. Il robot è dotato di un normale apparato percettivo e riesce a vedere gli ostacoli ma non gli oggetti che rispetto al suo punto di vista sono nascosti dagli ostacoli.

a. Si dia una formulazione del problema come problema di ricerca in uno spazio di stati prestando attenzione alle dimensioni dello spazio di ricerca.
b. Si definisca una euristica ammissibile da usare con un algoritmo A*.



Soluzione

Si potrebbe sovrapporre una griglia più o meno fitta sopra la stanza e fare muovere il robot per piccoli passi nel mondo a griglia.
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 P e di arrivo A, e considerando spostamenti in linea retta tra questi punti.

Questa semplificazione non fa perdere l'ottimalità.

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 percorribile 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).

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 con una azione).

Il problema quindi 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 $g(s)$ è la somma delle lunghezze dei tratti percorsi per arrivare nel punto del piano che corrisponde allo stato s e $h(s)$ la distanza in linea d'aria da s al punto di arrivo A.

Questa è chiaramente una sottostima per la presenza di ostacoli, che il robot deve aggirare, visto che non vola e non può penetrare gli ostacoli. Essendo una euristica *ammissibile* (e anche *consistente*) la soluzione trovata dall'algoritmo è una soluzione ottimale.

Scala di parole

Una scala di parole è una sequenza di parole significative (appartenenti al vocabolario italiano nel nostro caso) tale che ogni parola differisce dalla precedente esattamente per una lettera. Sia dato il problema di trovare una scala di parole, più breve possibile, che porta da una parola data ad un'altra, anch'essa data. Ad esempio, una scala da **auto** a **vita** potrebbe essere:

auto > muto > mito > dito > dita > vita

a. Si formuli il problema come un problema di ricerca in uno spazio di stati.
b. Quale è la dimensione massima dello spazio di ricerca per parole di lunghezza *k?* Dare una stima, anche approssimata, del fattore di diramazione relativamente all'esempio.
c. Quale algoritmo di ricerca non informata usereste per ricercare una soluzione nello spazio degli stati? Discutere completezza, ottimalità, complessità della soluzione proposta.
e. La ricerca bidirezionale sarebbe appropriata per questo problema?
d. Si proponga un'euristica da usare con un algoritmo di ricerca informata. Si dica se l'euristica è ammissibile e se ne discuta la completezza e l'ottimalità.

Soluzione

a)
Stati: parole
Stato iniziale: parola da cui si parte (es. auto)
Stato finale: parola che si vuole ottenere (es. vita)
Operatori: i successori di una parola p di lunghezza *k* sono tutte le parole che appartengono a Diz(*k*)—le parole del dizionario di lunghezza *k*--- e che hanno una sola lettera diversa da p.
Costo azione: 1

b)
Numero delle parole in Diz(*k*) è un limite superiore. Una stima al fattore di diramazione è *k* x *n*, dove *n* è il numero di parole significative ottenute sostituendo a una delle *k* lettere un'alternativa, tipicamente molto minore delle 25 possibili (come sarebbe considerando un alfabeto di 26 lettere).

c)
Ricerca in ampiezza. Il fattore di diramazione *b* è comunque piuttosto alto e quindi l'occupazione di memoria per tenere in memoria tutta la frontiera fino ad arrivare alla soluzione che si trova ad una profondità variabile *d*, tipicamente maggiore di *k*. Occupazione memoria O(b^d). Tempo O(b^d). Completo e ottimale.
Ricerca in profondità. Esiste un limite alla profondità se si spezzano i cicli. Il limite è dato dal numero di parole di lunghezza *k*. L'ottimalità non è garantita. Si potrebbe allora usare l'approfondimento iterativo, che garantisce completezza e ottimalità. Occupazione memoria O(*b*.*d*).
Ricerca bidirezionale. È possibile utilizzare la ricerca bidirezionale e in tal modo ridurre tempo e occupazione di memoria rispetto alla ricerca in ampiezza in una solo direzione.

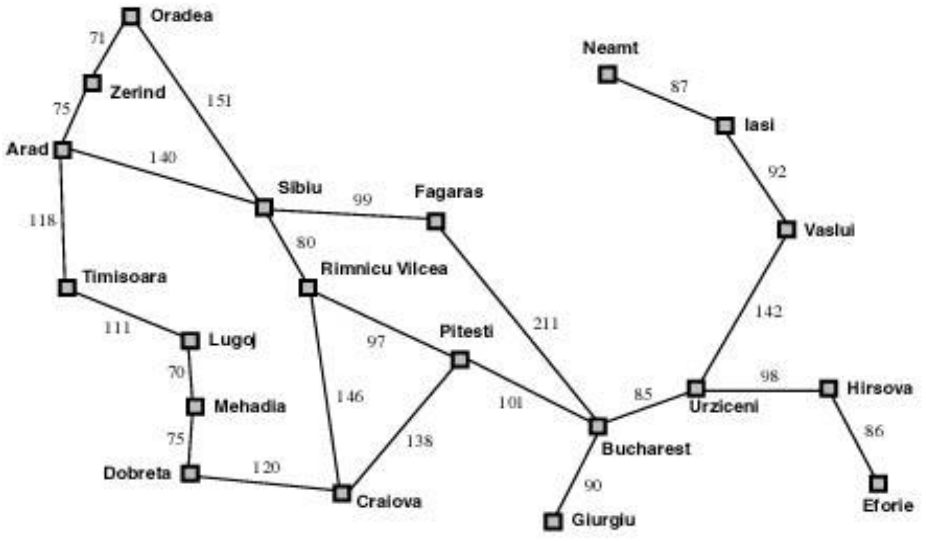
d)
Un'euristica che stima la distanza dalla soluzione potrebbe essere "contare il numero di lettere diverse rispetto alle corrispondenti nella parola obiettivo". È un'euristica ammissibile: infatti se non avessi il vincolo della significatività della parola (se tutte le combinazioni di lettere avessero significato) sarebbe un oracolo. Se usata con A, garantisce completezza e una soluzione ottimale. Si noti che se l'euristica fosse usata con Hill-climbing la completezza non sarebbe garantita perché in certe situazioni può succedere che ci si debba "allontanare" dalla parola obiettivo per trovare una catena.

Esempio: nel problema di trovare una catena da PIGRO a PEGNO.
PIGRO dista 2 da PEGNO; ma non esistono parole di senso compiuto ottenibili cambiando la I in E o la R in N (almeno io non le ho trovate; se ci fossero basta restringere il vocabolario). Questo non significa che la catena non esiste, ma che ci si deve temporaneamente allontanare. Una catena potrebbe essere
PIGRO [2] > MIGRO [3] > MAGRO [3] > MAGNO [3] > RAGNO [2] > REGNO [1] > PEGNO [0]
In questo caso Hill-climbing si arrenderebbe.

La mappa della Romania rivisitata

Due amici vivono in città diverse su una mappa, come quella della Romania mostrata in figura. L'obiettivo dei due amici è incontrarsi il più presto possibile. Ad ogni passo ciascuno si sposta su una città collegata nella mappa. Il tempo necessario per spostarsi è pari alla lunghezza della strada, ma il primo che arriva a destinazione deve aspettare che anche l'altro arrivi (e lo chiami sul cellulare) prima di procedere verso la destinazione successiva.

a. Si formuli il problema come un problema di ricerca: si definisca lo spazio di ricerca, la funzione successore, il goal e il costo delle azioni.



Straightline distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

b. Se DLA(*i*, *j*) è la distanza in linea d'aria tra due città *i* e *j* quali tra le seguenti euristiche sono ammissibili per uno stato (*i. j*)?

h_1. DLA(i, j)
h_2. 2 × DLA(i, j)
h_3. DLA(i, j)/2

c. Ci sono mappe completamente connesse per cui non esiste soluzione?
Soluzione

a. Formulazione del problema
Stati: Lo stato deve tener conto della posizione dei due amici. Possiamo rappresentarli come una coppia (x, y) dove x è la città in cui si trova il primo amico e y la città in cui si trova il secondo.
Stato iniziale: Lo stato corrispondente alle città in cui vivono.
Goal-Test: un qualunque stato (x, x)
Funzione successore: Se lo stato è (x, y) gli stati successori sono tutti gli stati (x', y') dove x' è uno stato direttamente raggiungibile da x e y' è uno stato direttamente raggiungibile da y.
Costo delle azioni: *max*(costo(x, x'), costo(y, y')) dove costo(i, j) è il costo della strada tra i e j.

b. Solo la h_3 è ammissibile. Infatti nel caso più ottimistico i due amici si trovano in città collegate da una o più strade in linea retta, con un numero dispari di città intermedie, e le strade da percorrere sono tutte della stessa lunghezza (nessuno dovrà perdere tempo ad aspettare l'altro). In tale situazione il tempo minimo impiegato coincide con DLA(i, j)/2 in quanto ciascuno dovrà percorrere esattamente la metà della strada che li separa, che in questo caso coincide con la distanza in linea d'aria. Tipicamente il tempo impiegato sarà maggiore e quindi, per ogni stato (i, j) $h_3((i, j)) =$ DLA(i, j)/2 ≤ h*((i, j)). Le altre due euristiche invece possono sovrastimare.

Esempio.
A ---------- B --------- C ---------- D --------- E stato iniziale (A, E)

c. Basta considerare una mappa con solo due città, quelle in cui vivono i due amici, direttamente collegate da una strada.

A ---------- B

Il cavallo

Supponiamo una scacchiera infinita ed un cavallo nella posizione iniziale (1,1). 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).

1. Si formuli il problema come un problema di ricerca in uno spazio di stati.
2. Si trovi una euristica ammissibile, il più possibile informata, per il problema.
3. Si trovi una soluzione tracciando l'andamento dell'algoritmo A* sull'istanza del problema in figura 2 (obiettivo in (5, 4)).

Fig 1

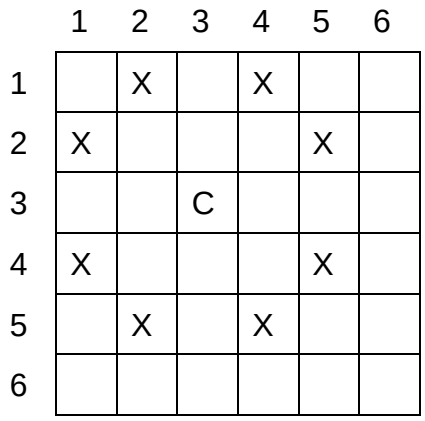
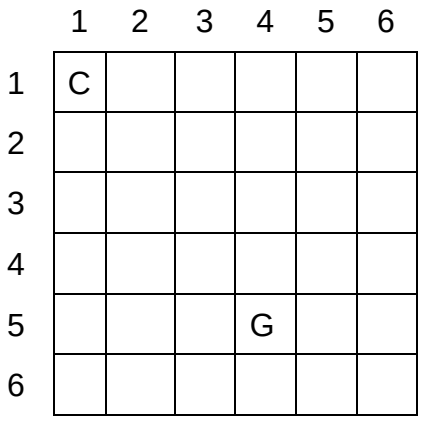


Fig 2



Soluzione

1. Stati: coordinate (x, y) del cavallo.
   Stato iniziale: (1, 1)
   Stato goal: (5, 4)
   Funzione successore: (x, y) -> {(x+1, y-2), (x+2, y-1), (x+2, y+1), (x+1, y+2), (x-1, y+2), (x-2, y+1), (x-2, y-1), (x-1, y-2)}
   Costo: 1 per ogni mossa

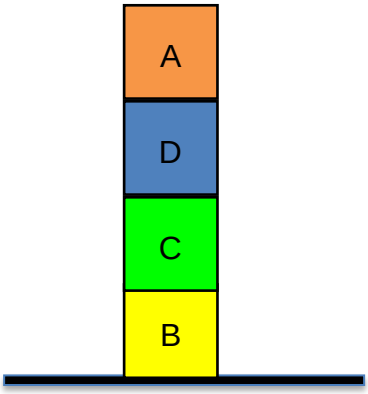
2. Una buona euristica ammissibile per il problema è h = $\lceil MD/3 \rceil$ arrotondato per eccesso. Ad esempio, la stima dallo stato iniziale allo stato goal sarebbe $\lceil 7/3 \rceil = 3$. In questo caso il goal è direttamente raggiungibile in tre mosse (meglio non si può fare) ma tipicamente ne serviranno di più, quindi l'euristica è una sottostima.

Confronto di euristiche ammissibili

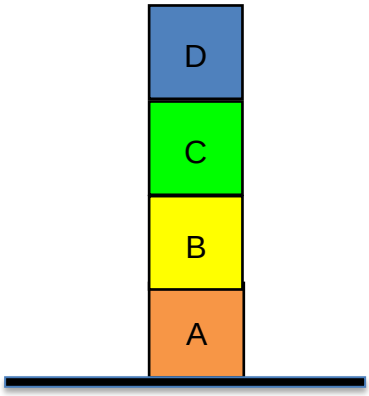
Il mondo dei blocchi

Il mondo dei blocchi è un micro-mondo classico per la ricerca in pianificazione.
Data una certa configurazione iniziale dei blocchi su un tavolo, in cui i blocchi possono essere impilati l'uno sull'altro, si tratta di raggiungere una determinata configurazione obiettivo.
Azioni lecite: spostare un blocco alla volta dal tavolo sulla cima di un altro blocco, da un blocco a un altro blocco o da un blocco al tavolo. Per spostare un blocco è necessario che la superficie del blocco sia libera. Su ogni blocco c'è solo posto per un altro blocco, quindi anche il blocco di destinazione deve essere libero. Spostare sul tavolo è sempre possibile. Ogni mossa costa 1 e ci interessa trovare il piano con meno passi.

Stato iniziale



Stato goal



Euristiche ammissibili possibili? Discutiamo e confrontiamo quelle proposte.

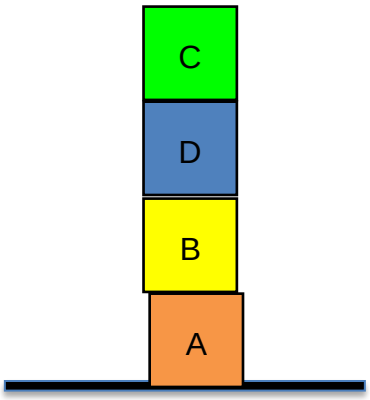
H1 = numero dei blocchi appoggiati su blocco *sbagliato* (o stanno sul tavolo e alla fine no)
H2 = numero dei blocchi con torre di supporto (i blocchi sotto) *sbagliata*

Nota: per "sbagliato" si intende che non è nella posizione che dovrebbe avere nello stato finale. Ci dà una misura della distanza dalla soluzione visto che dovranno essere compiute delle mosse per metterli a posto.

Nell'esempio precedente:

H1(stato iniziale) = 2 A, B sono sbagliati: A sta su B ma dovrebbe stare sul tavolo; B sta sul tavolo ma dovrebbe stare su A. D e C sono corretti. H1(goal)=0	H2(stato iniziale) = 4 Tutti sbagliati considerando la torre di supporto, perché nessuno ha tutta la torre sotto corretta. H2(goal)=0

Altro esempio (sempre rispetto allo stato goal)



H1 = 2

H2 = 2

Formalizzando le regole del gioco e rilassando uno o più vincoli si possono trovare altre euristiche ammissibili. La mossa è descritta da un insieme di vincoli:

1. Si può spostare **un blocco alla volta** (non torri di blocchi).
2. … se la superficie superiore del blocco da spostare è **libera**.
3. … se il blocco di destinazione è **libero** oppure la destinazione è il tavolo

Altre euristiche che potrebbero venire in mente, si discuta la loro ammissibilità. Ad esempio:

1. Numero di blocchi la cui altezza differisce dall'altezza finale.
2. Somma delle differenze in altezza.

Soluzione

Entrambe le euristiche sono ammissibili.
NOTA: *appoggiati su blocco sbagliato* implica *con supporto sbagliato*

H2(s) >= H1(s), H2 domina H1
H2 più accurata di H1

Formalizzando le regole del gioco e rilasciando uno o più vincoli si possono trovare altre euristiche ammissibili. La mossa è descritta da un insieme di vincoli:

1. Si può spostare **un blocco alla volta** (non torri di blocchi).
2. … se la superficie superiore del blocco da spostare è **libera**.
3. … se il blocco di destinazione è **libero** oppure la destinazione è il tavolo

In tutti i casi l'euristica è definita come la distanza esatta dal goal nel problema meno vincolato, quindi è necessariamente ammissibile per il problema classico in quanto le azioni nel gioco più vincolato sono anche possibili nel gioco meno vincolato.

1. Consentiamo di spostare torri di blocchi con costo 1 rilasciando il primo vincolo (solo il blocco in cima alla torre deve essere libero).
   In questa versione semplificata si arriva alla soluzione in due mosse: MOVE(A, Table), MOVE (B-C-D, A). Intuitivamente questa euristica è una sottostima per ogni stato iniziale e finale perché i blocchi con blocco sotto corretto non devono essere spostati uno alla volta ma sono

mantenuti al posto corretto come parte della torre che si sposta. La distanza stimata della soluzione coincide con H1.

2. Si può spostare un blocco alla volta ma anche blocchi non liberi.
   Nell'ipotesi che i blocchi NON cadano quando si sposta il blocco sottostante, i blocchi che non hanno la torre sotto corretta devono essere tutti messi a posto ma se non dobbiamo fare mosse per liberarli per ciascuno basta una mossa.
   Nell'esempio servono 4 mosse: Move(A, Table), Move (B, A), Move(C, B), Move(D, C).
   Potrebbe coincidere con H2.
   Nell' ipotesi che i blocchi cadano sul blocco sottostante la distanza potrebbe anche essere inferiore di quanto stimato con H2.
   Nell'esempio il goal sarebbe raggiungibile in 3 mosse: Move (B, A), Move(C, B), Move(D, C).
   Comunque rimane ammissibile.

3. Si può spostare un blocco alla volta ma anche su blocchi occupati (inserendosi tra un blocco e l'altro) o sul tavolo.
   In questa versione del gioco lo stato finale è raggiungibile in una mossa. È comunque una euristica ammissibile. Corrisponde a un'euristica H0 ancora meno informata di H1. H0 <= H1.

Altre euristiche che potrebbero venire in mente:

3. Numero di blocchi la cui altezza differisce dall'altezza finale. Ammissibile ma poco informata. Esempio:

A	D	A	D	
C	B	B	C	H = 0 mentre la distanza è 4.
---		---		
Stato iniziale		Goal		

4. Somma delle differenze in altezza: non ammissibile.

Confronto di euristiche ammissibili basato su proprietà matematiche

Siano date due euristiche ammissibili: h_1 e h_2. Quali delle seguenti euristiche sono ammissibili? Motivare le risposte.

a. $h(s) = h_1(s) + h_2(s)$
b. $h(s) = |h_1(s) - h_2(s)|$, dove | . | indica il valore assoluto
c. $h(s) = max(h_1(s), h_2(s)) - 2$
d. $h(s) = 2h_1(s) + h_2(s)/2$
e. $h(s) = (h_1(s) + h_2(s))/2$

Soluzione

a. Anche se $h_1(s) \leq h^*(s)$ e $h_2(s) \leq h^*(s)$ non è detto che la loro somma (o il loro prodotto) lo sia e quindi in generale la somma (o il prodotto) di due euristiche ammissibili non è ammissibile.
b. $|h_1(s) - h_2(s)| \leq h_1(s) \leq h^*(s)$ e quindi ammissibile nel caso $h_1(s) \geq h_2(s)$
   $|h_1(s) - h_2(s)| \leq h_2(s) \leq h^*(s)$ e quindi ammissibile nel caso $h_1(s) \leq h_2(s)$

c. Non è una buona stima di distanza perché sul goal non vale 0 ma -2.
d. Non è detto che sia ammissibile. Ad esempio se fosse $h_1(s) = h_2(s)$,
$h(s) = 2h_2(s) + h_2(s)/2 = 2.5\ h_2(s)$
e quindi potrebbe sovrastimare. Potrebbe succedere in un caso specifico che trova la soluzione ottimale ma non abbiamo alcun conforto dal risultato teorico.
f. La media delle due euristiche è sicuramente inferiore della più grande e quindi ancora ammissibile, ma comunque meno informata della maggiore.

Combinazione lineare di euristiche ammissibili

Questo caso merita un discorso a parte.
Sia h_1 l'euristica della somma delle distanze Manhattan di ogni numero dalla sua destinazione per il gioco dell'otto. Si consideri l'euristica $h_2=2*h_1+3$

a. h_2 è una euristica ammissibile?
b. Possiamo garantire che troverà una soluzione ottimale se usata con un algoritmo A (con costo azioni > e > 0)?
c. Se si è risposto SI alle prime due domande, la ricerca sarà più efficiente che con h_1?

Motivare le risposte.

Soluzione

1	2	3
4	5	6
7		8

a. L'euristica h_2 non è ammissibile.
Di fatto non è nemmeno una stima di distanza essendo diversa da zero sul goal. Non è difficile nemmeno far vedere un caso in cui sovrastima la distanza dalla soluzione. Per esempio, nel gioco dell'8, se h_1 è il numero di caselle fuori posto e $h_2=2*h_1+3$, nella seguente configurazione la soluzione dista 1 mossa ma h_2 vale 5.

Ma potrebbe venirci il dubbio che le due euristiche ordinino i nodi sulla frontiera nello stesso modo e producano esattamente le stesse scelte. Siano n e n' due nodi sulla frontiera e confrontiamo i due ordinamenti secondo

$$f_1(n) = g(n) + h_1(n) \quad \text{e} \quad f_2(n) = g(n) + h_2(n) = g(n) + 2*h_1(n) + 3$$

Supponiamo

$$g(n) + h_1(n) > g(n') + h_1(n')$$

è anche anche sempre vero che

$$g(n) + 2h_1(n) + 3 > g(n') + 2h_1(n') + 3\ ?$$

Chiaramente non è detto: se moltiplichiamo per 2 uno solo degli addendi il risultato può cambiare e quindi la nuova euristica potrebbe indurre scelte diverse.

Ma esistono euristiche non ammissibili che ci danno la garanzia di ottimalità?
Possibile: il risultato teorico ci dice Ammissibile => Ottimale. Ma un'euristica potrebbe essere dimostrata ottimale per altra via.

Algoritmo del percorso euristico

L'algoritmo del **percorso euristico** è un algoritmo *best first* in cui la funzione di valutazione è $f(n) = (2 - w)\, g(n) + w\, h(n)$

a. Quale ricerca implementa l'algoritmo quando $w = 0$, $w = 1$ e $w = 2$?
b. Per quali valori di w l'algoritmo è completo, assumendo che h sia una stima di distanza?
c. Per quali valori di w l'algoritmo è ottimale, assumendo che h sia ammissibile?

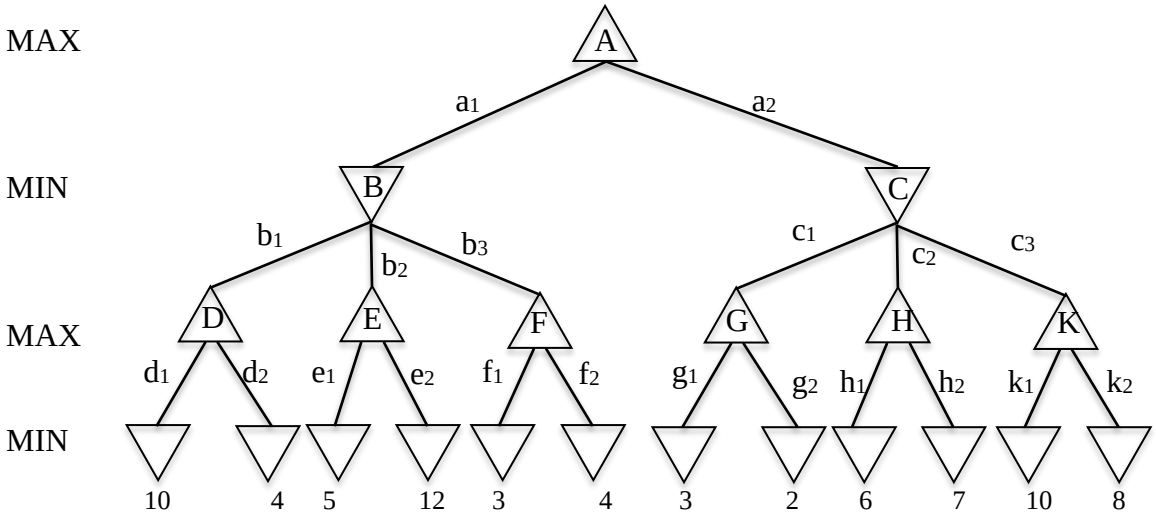
Soluzione

a. Con $w = 0$, $f(n) = 2g(n)$. Un algoritmo *best-first* con questa euristica si comporta esattamente come la ricerca di costo uniforme — il fattore due non fa differenza nell'ordinamento dei nodi.
   Con $w = 1$ abbiamo la ricerca A∗.
   Con $w = 2$ abbiamo $f(n) = 2h(n)$, cioè la ricerca *greedy best-first*.
b. Possiamo riscrivere la definizione come: $f(n) = (2 - w)\, [g(n) + w/(2-w)\, h(n)]$ mettendo in evidenza $(2 - w)$. Questo ci dice che l'algoritmo si comporta come A* con una componente euristica $w/(2-w)\, h(n)$. Pertanto l'algoritmo è completo quando $0 <= w/(2-w)\, h(n) <= 1$, cioè quando $0 \leq w < 2$. Per valori di w inferiori a 0 o superiori a 2 l'euristica sarebbe negativa, e quindi non una buona stima di distanza. Per $w = 2$, sarebbe indefinita.
c. Se $w \leq 1$, l'euristica è inferiore ad $h(n)$ e quindi ammissibile, visto che $h(n)$ è ammissibile per ipotesi.

Giochi con avversario

Esempio di Min-Max e Alfa-Beta

Sia dato il seguente albero di gioco con il valore della funzione di valutazione euristica riportata sulle foglie.

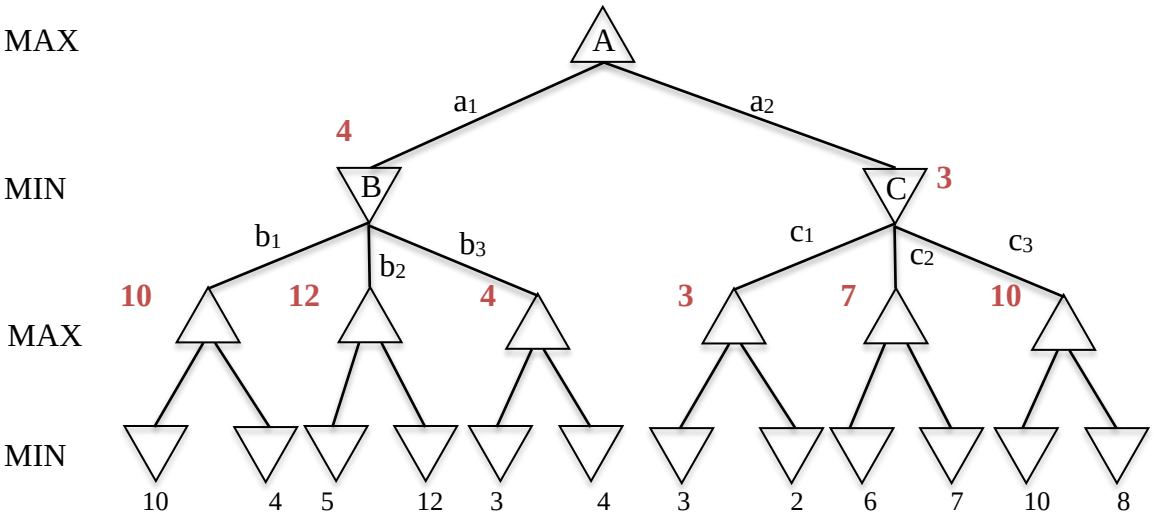


1. Qual è la valutazione minimax dei nodi e quale azione sceglierebbe di fare MAX secondo l'algoritmo Min-Max?
2. Quali nodi si potrebbe evitare di valutare con Alfa-Beta.

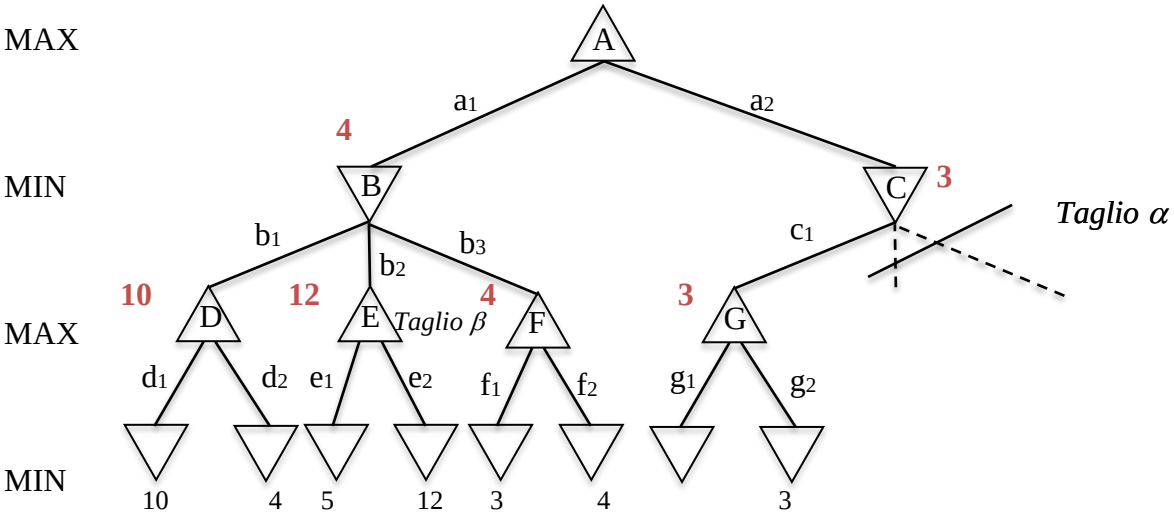
3. Esiste un ordinamento delle mosse migliore che consente di valutare ancora meno nodi?

Soluzione

1) Valutazione dei nodi intermedi secondo la funzione minimax. La mossa scelta è la a_1.

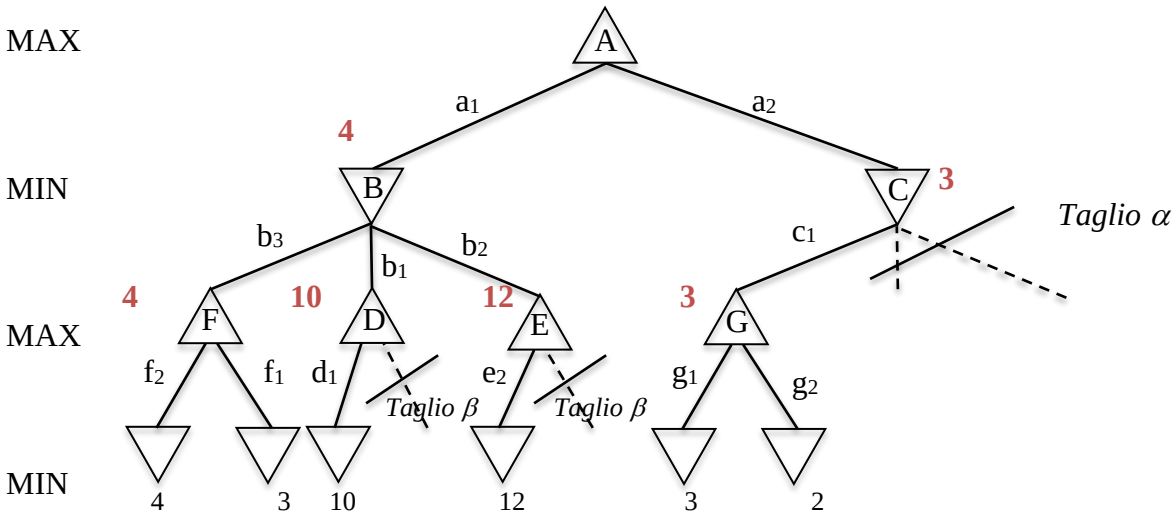


2) Visita dell'albero di gioco con Alfa–Beta.



Nota: il taglio sotto E avviene dopo aver valutato tutti i successori e quindi non comporta alcuna ottimizzazione.

3) Visita dell'albero di gioco con Alfa–Beta con un ordinamento ottimale delle mosse. Sotto i nodi MAX le mosse sono ordinate per valore minimax decrescente; sotto i nodi MIN sono ordinate per valore minimax crescente. Ovviamente altri ordinamenti potrebbero produrre le stesse potature.



Sistemi di soddisfacimento di vincoli (CSP)

Il puzzle della zebra

Si consideri il seguente puzzle logico (semplificazione del puzzle della Zebra di Einstein).
In tre case, ognuna di un colore diverso (bianca, rossa, gialla), vivono tre persone di nazionalità diverse (uno svizzero, un ítaliano e un greco), ognuna delle quali possiede un particolare (e distinto) animale (farfalle, serpente, gatto).

1. *Lo svizzero vive nella casa gialla.*
2. *La prima casa è Bianca.*
3. *C'è una casa tra la casa in cui vive il greco e quella più a sinistra in cui vive il serpente*
4. *La persona nella seconda casa non sopporta i gatti.*

a. Si formuli il puzzle come un problema di soddisfacimento di vincoli definendo le variabili, i loro domini e i vincoli tra le variabili.
b. Si cerchi di rispondere alla domanda "*Di che colore è la casa in cui vive il gatto e a chi appartiene?*", applicando una delle strategie viste a lezione.
Ad esempio MRV+grado+FC

Soluzione

Variabili: *Rossa, Bianca, Gialla, Svizzero, Italiano, Greco, Farfalle, Serpente, Gatto*
Domini: {1, 2, 3}
Vincoli:
Gialla = *Svizzero*
Bianca = 1
Serpente = *Greco* - 2
Gatto ≠ 2
Italiano ≠ *Greco* ≠ *Svizzero*
Bianca ≠ *Rossa* ≠ *Gialla*
Farfalle ≠ *Serpente* ≠ *Gatto*

Risolviamo con MRV (a parità di MRV euristica del grado) e FC. Risolviamo subito i vincoli unari.

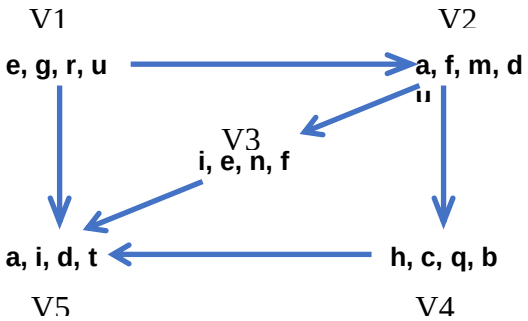
Step	*Rossa*	*Bianca*	*Gialla*	*Svizzero*	*Italiano*	*Greco*	*Farfalle*	*Serpente*	Gatto
0	{1, 2, 3}	{1}	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}	{1, 3}
1	{2, 3}	1	{2, 3}	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}	{1, 3}
2	{3}	1	2	{2}	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}	{1, 3}
3	{3}	1	2	2	{1, 3}	{1, 3}	{1, 2, 3}	{1, 2, 3}	{1, 3}
4	3	1	2	2	{1, 3}	{1, 3}	{1, 2, 3}	{1, 2, 3}	{1, 3}
5	3	1	2	2	{1}	~~1~~ 3	{1, 2}	{1}	{1, 3}
6	3	1	2	2	{1}	3	{2}	1	{3}
7	3	1	2	2	1	3	{2}	1	{3}
8	3	1	2	2	1	3	2	1	{3}
9	3	1	2	2	1	3	2	1	3

Step 0. Abbiamo risolto i vincoli unari riducendo i domini di *Bianca* e di *Gatto*.
Step 1. Per MRV scegliamo *Bianca*. Assegniamo *Bianca*=1 e applichiamo FC.
Step 1. Per MRV potremmo scegliere *Rossa*, *Bianca* o *Gatto*. Scegliamo *Gialla* per il grado. Poi FC.
Step 3. Per MRV e grado scegliamo *Svizzero*. Valore 2.
Step 4. Per MRV *Rossa* = 3
Step 5. MRV: *Italiano*, *Greco* o *Gatto*. Meglio *Greco* per il grado. *Greco* = 1. Il FC, che ci impone *Serpente* = *Greco* – 2 fallisce: il dominio di Serpente si svuota. Devo fare backtracking e provare con Greco = 3.
Step 6. MRV. *Italiano* o *Serpente*. *Serpente* = 1.
Step 7. *Italiano* = 1
Step 8. *Farfalle* = 2
Step 9. *Gatto* = 3

La casa in cui vive il gatto è rossa e ci vive il Greco.

Etichettatura di grafo

Sia dato il problema di etichettare il grafo in modo che per ogni arco orientato l'etichetta del nodo sorgente preceda in ordine alfabetico l'etichetta del nodo destinazione. Sono date inizialmente un insieme di etichette possibili per ogni nodo, come nell'esempio.



Si imposti il problema come CSP e si risolva applicando Min-Conflicts.

Soluzione

Introduciamo una variabile per ogni nodo del grafo, con dominio l'insieme delle etichette possibili.

$V1 \in \{e,g\ r,u\}$

V2∈{a,f,m,d}
V3∈{i,e,n,f}
V4∈{h,c,q,b}
V5∈{a,i,d,t}

I vincoli sono di questo tipo:

Se i --> j (c'è un arco orientato da i a j) allora il valore di Vi (l'etichetta del nodo) deve precedere in ordine alfabetico il valore di Vj.

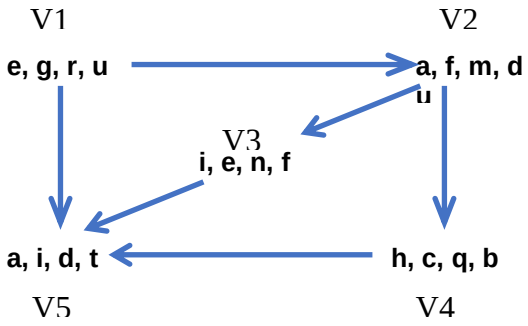
Il grafo dei vincoli in questo caso coincide con il grafo sopra.

Ad ogni passo si assegna una variabile e si ripristina la consistenza degli archi: un arco è consistente se per ogni valore del nodo sorgente c'è almeno un valore ammissibile (che rispetta i vincoli) nel nodo destinazione.

Etichettatura grafo con min-conflicts

Essendo un metodo locale ci serve una formulazione del problema a stato completo.

Stato iniziale: un assegnamento a caso di valori a variabili
Azioni: cambiare di valore a una variabile che è responsabile della violazione di qualche vincolo
Euristica: si prende a caso una variabile responsabile della violazione di qualche vincolo e le si assegna il valore che minimizza il numero di conflitti (si vede per ogni possibile valore quanti conflitti avrebbe).



Soluzione

	V1	V2	V3	V4	V5
Ass. random	g	**a**	e	q	d
Scelgo V2=a [f: 2; **m: 1**; d: 1]	**g**	m	e	q	d
Scelgo V1=g [**e: 2**; r: 2; u: 2]	e	m	e	q	**d**
Scelgo V5=d [a: 3; i: 1; **t: 0**]	e	m	**e**	q	t
Scelgo V3=e [i: 1; **n: 0**; f: 1]	e	m	n	q	t

Soluzione: tutti vincoli sono soddisfatti.