Ricerca informata

Una ricerca informata sfrutta la conoscenza specifica del dominio applicativo per fornire suggerimenti su dove si potrebbe trovare l'obiettivo, più efficientemente di una non informata. I suggerimenti hanno la forma di una funzione euristica.

Ricerca Best-First Greedy

E' una forma di ricerca best-first che espande prima il nodo con il valore più basso $h(n)$, cioè quello che appare più vicino all'obiettivo $(f(n) = h(n))$. Non ottimo e non completo in spazio degli stati infinito.

A*

Una ricerca best-first con $f(n) = g(n) + h(n)$ dove $g(n)$ è il costo del cammino dal nodo iniziale al nodo n e $h(n)$ rappresenta il costo stimato del cammino più breve dal nodo n al nodo obiettivo.
$f(n)$ è il costo stimato del cammino migliore che continua da n fino al goal.

A* con $\epsilon > 0$ (limite inferiore al costo di ogni azione) è completo.
Infatti sia n^* un nodo generico in frontiera nel cammino soluzione, questo (a meno che non si trovi prima la soluzione) verrà espanso.
L'ottimalità dipende da alcune proprietà
Diciamo che un'euristica è ammissibile se non sovrastima mai il costo effettivo per raggiungere un obiettivo. Un'altra proprietà è la consistenza. Un'eurista $h(n)$ è consistente se per ogni nodo n e ogni suo successore n' generato da un'azione a, abbiamo:
$h(n) \leq c(s, a, s') + h(n')$

CONSISTENTE $\implies$ AMMISSIBILE.

Con un'euristica consistente non bisognerà modificare raggiunti o riaggiungere uno stato alla frontiera , perché una volta raggiunto uno stato questo è sicuramente in un cammino ottimo.
Con un'euristica inconsistente potremmo trovarci con più cammini che raggiungono lo stesso stato, e se ogni cammino nuovo ha costo inferiore al precedente finiremmo con avere più nodi corrispondenti allo stesso stato in frontiera, con un aggravio nei costi temporali.
Con un'euristica inammissibile, A* potrebbe essere ottimo in due casi:

- Se vi è anche un solo cammino ottimo rispetto al costo lungo cui $h(n)$ è ammissibile per tutti i nodi n nel cammino.

- Se la soluzione ottima ha costo C^* e la seconda migliore C_2 e se $h(n)$ sovrastima alcuni costi ma mai più di $C_2 - C^*$, allora A* restituisce sempre il cammino ottimo.

E' chiaro che quando si estende un cammino, i costi $g(n)$ sono monotoni: il costo di un cammino incrementa sempre mentre lo si percorre ($\epsilon > 0$). Quando si estende un cammino da n a n', il costo da $g(n) + h(n)$ diventa $g(n) + c(n, a, n') + h(n')$. Eliminando il termine $g(n)$ vediamo che il costo del cammino sarà monotono crescente se e solo se $h(n) \leq c(n, a, n') + h(n')$, quindi se l'euristica è consistente.

Se C^* è il costo del cammino ottimo:

- A* espande tutti i nodi che possono essere raggiunti dallo stato iniziale su un cammino in cui per ogni nodo n si ha che $f(n) < C^*$. Questi sono **certamente espansi**.
- A* potrebbe espandere alcuni nodi dove $f(n) = C^*$ prima di arrivare al nodo obiettivo.
- A* non espande alcun nodo $f(n) > C^*$.

A* è **ottimamente efficiente** (altri algoritmi che usano la stessa euristica di A*, espandono gli stessi nodi, espansi da A*).
A* è efficiente perché esegue la **potatura** dell'albero di ricerca dei nodi non necessari per trovare una soluzione ottima, ovvero scarta alcune possibilirà senza doverle nemmeno esaminare.
In più e completo e ottimo rispetto al costo. Ma per molti problemi il numero di nodi espansi può aumentare esponenzialmente con la lunghezza della soluzione. La memoria usata è $O(b^{d+1})$.

Beam Search

Mantiene solo i k nodi con i migliori costi f scartando ogni altro nodo espanso, ciò rende la ricerca subottima e incompleta.

IDA*

Nel ID la soglia è la profondità, che aumenta di 1 ad ogni iterazione. Nel IDA* la soglia è il costo $f(g + h)$ a ogni iterazione il valore soglia è il più piccolo costo f di qualsiasi nodo che abbia superato la soglia nella precedente iterazione. L'algoritmo è completo e ottimale se:

- le azioni hanno costo costante k e soglia viene incrementato di k.
- le azioni hanno costo variabile e l'incremeneto di soglia è $\leq \epsilon$ (limite inferiore costo azioni).
- la nuova soglia = min valore f nodi generati ed esclusi dall'iterazione precedente.

Memoria usata $O(bd)$.

Ricerca Best-First ricorsiva

l'algoritmo sembra la DF ricorsiva solo che invece di continuare a seguire il cammino corrente, utilizza la variabile soglia per tener traccia del valore $f(n)$ del miglior cammino alternativo che parte da uno qualsiasi degli antenati del nodo corrente. Se il nodo corrente supera questo limite la ricorsione torna indietro al cammino alternativo. Durante il ritorno viene sostituito il valore f di ogni nodo lungo il cammino con un valore di **backup**, il miglior valore f dei suoi nodi figli.
In questo modo RBFS ricorda il valore f della foglia migliore nel sottoalbero abbandonato e può quindi decidere in seguito di riespanderlo.

SMA*

Il problema dei due algoritmi è che usano troppa poca memoria.
SMA* procede esattamente come A* finché la memoria è piena. A questo punto se ne deve aggiungere un altro di nodo nell'albero scarta sempre il nodo foglia peggiore (quello con costo $f(n)$ più alto).
Memorizza nel nodo padre il valore del nodo dimenticato. Con questa informazione viene rigenerato il sottoalbero dimenticato solo quando tutti gli altri cammini premettono di comportarsi peggio.
A parità di f si sceglie il nodo migliore più recente e si dimentica il nodo peggiore più vecchio.
Completo se c'è una soluzione raggiungibile ovvero se $d <$ dimensione memoria.
La strategia è ottima se c'è una soluzione ottima altrimenti viene restitutita quella migliore.
Le limitazioni di memoria possono rendere un problema intrattabile dal punto di vista temporale.

Funzioni euristiche