Risol. Problemi

Criteri di valutazione

Esistono diversi criteri per valutare le strategie di ricerca dette "non informate" o cieche:

- Completezza
  - **descrive se la soluzione viene trovata**
- Complessità temporale
  - **tempo richiesto per trovare la soluzione**
- Complessità spaziale
  - **memoria richiesta per ricercare la soluzione**
- Ottimalità
  - **descrive se si trova la soluzione migliore**

Alberto Broggi

Risol. Problemi

Criteri di valutazione

	Ampiezza	Costo Uniforme	Profondità	Profondità Limitata	Profondità Iterativa	(Bidirezionale)
Tempo	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Spazio	b^d	b^d	bm	bl	bd	$b^{d/2}$
Ottimalità	SI(1)	SI(2)	NO	NO	SI(1)	SI
Completezza	SI	SI	NO	SI(3)	SI(4)	SI

(1): con operatori con costo uniforme
(2): per costi non negativi
(3): se si conosce il limite alla profondità
(4): quando $l \geq d$

Alberto Broggi

Risol. Problemi

Esempio

- Problema del commesso viaggiatore
  - deve visitare una lista di citta' una sola volta, seguendo il cammino migliore
- E' possibile esplorare tutti i cammini e scegliere il migliore solo se il numero di citta' e' basso
- Con n citta' il numero di cammini diversi e':
  $c = 1x2x3x \ldots x (n-1) = (n-1)!$
- Per n=10 si ha 3.628.800 **Esplosione combinatoria**

Alberto Broggi

Risol. Problemi

Commesso viaggiatore

- Si puo' migliorare la strategia con la tecnica detta *branch-and-bound* (dividi e limita)
  - si inizia generando cammini completi, e si memorizza quello piu' corto generato fin'ora
  - ogni altro cammino che si genera viene comparato al piu' corto memorizzato
  - se il nuovo cammino e' piu' lungo di quello memorizzato, si ferma l'esplorazione
- Fornisce la garanzia di trovare il cammino piu' corto, ma richiede ancora un tempo elevatissimo
- Soluzione inadeguata per problemi di grosse dimensioni

Alberto Broggi

Risol. Problemi

Ricerca Euristica

- Alcuni problemi sono così complessi che occorre costruire una strategia di controllo mirata non a trovare **la soluzione migliore**, ma una **soluzione buona**
- Si utilizzano *euristiche*, cioè tecniche che:
  - **migliorano l'efficienza della ricerca**
  - **però sacrificano la completezza**
- Migliorano la qualità media dei cammini esplorati

Alberto Broggi

Risol. Problemi

Esempio di euristica

- Generalmente una buona euristica è *l'euristica dell'elemento più vicino*
  - ad ogni passo si seleziona l'alternativa localmente migliore
- Per il commesso viaggiatore si ha:
  - per selezionare la prossima città, si sceglie la più vicina tra tutte quelle non ancora visitate
  - si ripete fino a che non sono state visitate tutte le città
- In questo caso il tempo è $O(n^2)$, meglio di $O(n!)$

Alberto Broggi

Risol. Problemi

Limite superiore all'errore

- Talvolta è possibile determinare un limite superiore per l'errore in cui si può incorrere
- E' importante perché assicura quanto possiamo pagare in termini di accuratezza in cambio della maggiore efficienza
- Purtroppo non lo si riesce a determinare per tutte le euristiche perché:
  - per problemi reali è difficile misurare la bontà di una soluzione
  - generalmente si usano euristiche basate su conoscenza poco strutturata

Alberto Broggi

Risol. Problemi

Euristiche

- Senza l'utilizzo di euristiche molti problemi non potrebbero essere risolti
- Anche se non portano alla soluzione ottima, almeno portano ad una soluzione
- Vi sono problemi in cui non c'è bisogno della soluzione ottima ma di una soluzione sufficientemente buona (esempio del parcheggio)
- Cercare di capire perché un'euristica funziona o non funziona porta ad una migliore comprensione del problema

Alberto Broggi

Risol. Problemi

Utilizzo delle euristiche

- La conoscenza euristica specifica di un dato settore può essere inclusa nel processo di ricerca in due modi:
  - **nelle regole:**
    le regole non descriveranno solo le mosse possibili, ma le mosse "sensate" (determinato da colui che scrive le regole)
  - **nella funzione euristica:**
    la funzione valuta gli stati del problema e determina quanto sono buoni e desiderabili per giungere alla soluzione

Alberto Broggi

Risol. Problemi

La funzione euristica

- Determina la desiderabilità di ogni stato
- Funzioni euristiche progettate con cura possono giocare un ruolo importante nella guida efficiente di un processo di ricerca
  - serve per determinare la direzione più vantaggiosa quando è disponibile più di un cammino
- I programmi cercano di massimizzare o minimizzare le funzioni euristiche

Alberto Broggi

Risol. Problemi

La funzione euristica

- Più la funzione euristica stima accuratamente i meriti di ogni nodo, più il processo di soluzione sarà diretto
- Al limite una funzione ottima potrebbe evitare la ricerca, ma generalmente il costo per calcolare una funzione di questo tipo sarebbe paragonabile al costo della ricerca
  - sarebbe possibile calcolare la funzione perfetta dopo una ricerca completa
- Compromesso tra:
  - il costo per il calcolo della funzione euristica
  - il risparmio di tempo consentito dall'utilizzo della funzione

Alberto Broggi

Risol. Problemi

IA con euristiche

- La soluzione di problemi di IA è stata definita come un processo basato sulla ricerca
- Ora si può aggiungere: "...processo basato sulla ricerca euristica"
- In generale l'IA è lo studio delle tecniche per risolvere problemi difficili (esponenziali) in un tempo polinomiale sfruttando la conoscenza sul dominio

Alberto Broggi

Risol. Problemi

Caratterizzazione dei problemi

Per determinare le euristiche migliori è necessario caratterizzare i problemi

- Annullamento di passi:
  - problemi in cui **si possono ignorare** passi che non hanno portato a miglioramenti (dimostrazione di teoremi)
  - problemi in cui **si possono annullare** dei passi (puzzle dell'8)
  - problemi in cui **non si possono annullare** passi (scacchi)

Alberto Broggi

Risol. Problemi

Caratterizzazione dei problemi

- Pianificazione: definita come un metodo di soluzione senza retroazione da parte dell'ambiente
  - problemi **con risultato certo**:
    è possibile pianificare una sequenza di mosse ed essere sicuri sullo stato di arrivo; è necessario prevedere backtracking durante la pianificazione (puzzle dell'8)
  - problemi **con risultato incerto**:
    non è possibile pianificare una sequenza completa di mosse (giochi delle carte in cui non si conosce la distribuzione delle carte, giochi con avversario). E' necessario prevedere un processo di "*revisione del piano*"
  - problemi **con esplorazione**:
    nessuna conoscenza sul mondo non permette di generare nessuna pianificazione; è necessario provare ad agire

Alberto Broggi

Risol. Problemi

Caratterizzazione dei problemi

- Soluzione assoluta o relativa:
  - problemi con **una soluzione**, tipicamente problemi di classificazione; non è importante come si è giunti alla soluzione, è importante la soluzione
  - problemi con **molte soluzioni**, ad es. il commesso viaggiatore; è necessario selezionare la soluzione migliore tra tutte. Più complessi dal punto di vista computazionale
- Soluzione: stato o cammino:
  - problemi in cui la soluzione è uno **stato** (classificazione)
  - problemi in cui la soluzione è un **cammino** (secchi d'acqua)

Alberto Broggi

Risol. Problemi

Tecniche di ricerca euristica

- Le tecniche mostrate sono indipendenti dal campo di applicazione specifico, e la loro efficacia dipende da quanta conoscenza si sfrutta
- Sono detti *"metodi deboli"*
- E' riconosciuta la loro limitata efficacia, ma rimangono l'unico modo per risolvere problemi difficili (combinatori, esponenziali,...)

Alberto Broggi

Risol. Problemi

Ricerca lungo il cammino migliore (Best-First Search)

- I metodi precedenti si basano su una misura del costo delle sequenze di azioni, ma non hanno nessun modo per valutare quanto queste possibili soluzioni parziali siano vicine ad un goal del problema
- La valutazione viene effettuata localmente sulla base del costo di ogni singolo passo (Uniform Cost Search)

Alberto Broggi

Risol. Problemi

Ricerca lungo il cammino migliore (Best-First Search)

- Non è in generale completa: c'è sempre il rischio di dirigersi verso un cammino che *sembra* promettente, ma non porta alla soluzione
- La completezza e l'ottimalità dipendono dalle specifiche euristiche utilizzate
- Variante: *Beam-Search*
  - se l'occupazione di memoria è troppa, si tengono solo i k nodi più promettenti
  - k è detto ampiezza del raggio
  - la Beam-Search non è completa

Alberto Broggi

Ricerca lungo il cammino più vicino al goal (Greedy search)

- Questo metodo cerca di minimizzare il costo per raggiungere il goal. Il nodo che viene considerato più vicino al goal viene espanso.
- In molti casi questo costo può essere solo stimato e non calcolato deterministicamente.
- La funzione che fa questa stima del costo è detta funzione euristica ed è indicata dalla lettera h.
- h(n) stima il costo per andare dal nodo n al goal.

Ricerca lungo il cammino più vicino al goal (Greedy search)

- Simile alla ricerca in profondità
- Quindi non è ottimo e non è completo (esempio del commesso viaggiatore)
- La ricerca può essere velocizzata da una buona euristica.

L'algoritmo A*

- L'algoritmo lungo il cammino più vicino al goal (Greedy search) minimizza il costo, h(n), per raggiungere il goal
  - spesso permette di ridurre considerevolmente il costo della ricerca, ma non è completo e neppure ottimo.
- L'algoritmo di ricerca a costo uniforme minimizza il costo, g(n), del percorso dalla radice al nodo corrente
  - è completo e ottimo, ma può essere molto inefficiente.

L'algoritmo A*

- Combinando i due metodi si ottiene un algoritmo, detto A*, che offre i vantaggi di entrambi.
- Per fare ciò, l'algoritmo A* somma le loro due funzioni di valutazione: f(n) = g(n) + h(n).
- f(n): costo del cammino dalla radice al goal
- g(n): costo del cammino dalla radice a n
- h(n): costo del cammino da n al goal

L'algoritmo A*

- Si può provare che l'algoritmo A* è ottimo e completo se h è un'euristica *ammissibile*, cioè se non sovrastima il costo per raggiungere il goal
  - una sottostima può farci compiere del lavoro inutile,
  - ma non ci fa perdere il cammino migliore (mentre una sovrastima sì)
- La componente g fa abbandonare cammini che vanno troppo in profondità
- L'algoritmo A* è ottimamente efficiente per ogni funzione euristica, cioè non esiste alcun altro algoritmo ottimo che espande un numero di nodi minore dell'algoritmo A*

L'algoritmo A*

- Si può dimostrare che l'algoritmo A* è ottimamente efficiente per ogni funzione euristica, cioè non esiste alcun altro algoritmo ottimo che espande un numero di nodi minore dell'algoritmo A*:
  - Sia OG un goal a costo minimo f* e sia SG un goal a costo $f_g^* > f^*$
  - Supponiamo che A* abbia scelto SG e dimostriamo che ciò non è possibile
  - Sia n un nodo della frontiera sul percorso ottimo verso OG; dato che la funzione h è ammissibile, allora risulta: $f^* \geq f(n)$
  - Poiché si ha che $f(n) \geq f_g^*$ (altrimenti verrebbe preferito SG a n), allora segue che: $f^* \geq f_g^*$, che contraddice l'ipotesi

L'algoritmo A*

- L'algoritmo A* è completo: questo algoritmo espande nodi in ordine al valore di f. Quindi se un goal ha valore f*, l'algoritmo A* è completo se non ci sono infiniti nodi con f < f*.
- Anche se l'algoritmo A* è l'algoritmo di ricerca ottimo che espande il minor numero di nodi, in molti casi la complessità è esponenziale in funzione della lunghezza della soluzione.
- Si è provato che il numero di nodi espansi cresce esponenzialmente a meno che l'errore tra la h e il vero costo h* non cresca molto lentamente.

Scelta dell'euristica

- Consideriamo un problema molto semplice come 8-puzzle. Una soluzione tipica ha 20 mosse.
- Dato che il fattore di ramificazione è circa 3, allora una ricerca in profondità esaustiva visiterebbe $3^{20} = 3.5 \times 10^9$ nodi.
- Una parte della ricerca si può tagliare eliminando la visita di nodi già visitati, dato che gli stati possibili sono 9! = 362880.
- Tuttavia, sono numeri troppo elevati; occorre quindi cercare di tagliare di molto la ricerca con delle buone euristiche.

Scelta dell'euristica

- Due euristiche candidate per l'8-puzzle sono:
  - h_1 : il numero di quadrati fuori posizione;
  - h_2 : la somma delle distanze dei quadrati dalla loro posizione finale.
- Da risultati sperimentali si ricava che h_2 ha un migliore funzionamento:
  - quindi h_2 è una stima migliore del costo per raggiungere il goal.
- La scelta dell'euristica dovrebbe essere guidata dalla quantità di conoscenza sul problema che riesce a codificare

Scelta dell'euristica

- **Confronto tra la ricerca iterativa in profondità e A* con h_1 e h_2:**

d	IDS	A*(h1)	A*(h2)
2	10	6	6
4	112	13	12
6	680	20	18
8	6384	39	25
10	47127	93	39
12	364404	227	73
14	3473941	539	113
16		1301	211
18		3056	363
20		7276	676
22		18094	1219
24		39135	1641

Algoritmi per il miglioramento iterativo

- Ci sono dei problemi per cui la descrizione dello stato contiene tutte le informazioni necessarie per la soluzione, cioè non importa come si è arrivati a quello stato.
- In questi casi l'idea generale per la soluzione può essere quella di partire con una configurazione completa e fare delle modifiche per migliorare la configurazione attuale.
- Questo metodo può essere rappresentato come il movimento su una superficie che rappresenta lo spazio degli stati cercando di raggiungere la vetta più elevata (problemi di ottimizzazione).

Generazione e verifica

- E' il più semplice da implementare
  - si produce una soluzione candidata
  - si verifica se questa è davvero soluzione del problema
  - se si è trovata la soluzione si termina, altrimenti si ripete dal punto iniziale
- Se la produzione avviene in modo sistematico, allora si è certi che si arriverà alla soluzione
  - nella forma più sistematica coincide con una ricerca esaustiva
- E' una tecnica di ricerca in profondità perchè si devono produrre delle soluzione complete per poi verificarle

Risol. Problemi

Generazione e verifica

- La produzione delle soluzioni può anche avvenire in modo casuale (esempio della radice quadrata)
- Tra la completa sistematicità e la casualità esistono delle vie intermedie:
  - ricerca sistematica, tralasciando la generazione di soluzioni poco probabili (utilizzando euristiche)
- Utilizzato in DENDRAL (analisi chimica)

Alberto Broggi

Risol. Problemi

Ricerca in salita (Hill Climbing)

- L'algoritmo di ricerca in salita è un algoritmo ciclico che ad ogni passo si muove verso una direzione più promettente
- E' una versione migliorata della ricerca per "*generazione e verifica*": viene utilizzato un feedback proveniente dal processo di verifica per decidere verso quale direzione muoversi nello spazio degli stati
- E' utilizzato quando si ha una buona funzione euristica (trovare il centro di una città)

Alberto Broggi

Risol. Problemi

Ricerca in salita

- Esistono due varianti:
  - salita semplice
  - salita ripida
- Salita semplice:
  - si valuta lo stato iniziale e si ripete fino a trovare una soluzione
  - ad ogni iterazione si effettua una mossa
  - se la mossa porta ad uno stato migliore,
    - » allora lo stato di arrivo diventerà lo stato corrente
    - » altrimenti non ci si muove
- E' necessario definire cosa si intende per *migliore*, utilizzando un'opportuna euristica

Alberto Broggi

Risol. Problemi

Ricerca in salita

- Salita ripida:
  - si valuta lo stato iniziale e si ripete fino a trovare una soluzione
  - ad ogni iterazione si effettua una mossa
  - si sceglie la mossa tra tutte quelle possibili con lo scopo di muoversi verso la direzione più promettente
- E' nota come *ricerca lungo il gradiente*
- Ci si sposta nelle direzioni verso le quali la funzione di valutazione aumenta più velocemente

Alberto Broggi

Risol. Problemi

Ricerca in salita

- Entrambi i metodi di ricerca in salita possono non portare alla soluzione (quando ad esempio non si può generare nessuna mossa che porta ad un miglioramento)
- Questo succede quando ci si trova in:
  - **un massimo locale**: stato migliore dei vicini, ma non migliore in assoluto (soluzione)
  - **un pianoro:** zona piatta in cui stati vicini hanno valori uniformi
  - **un crinale:** zona con due pendenze di cui solo una interessante

Alberto Broggi

Risol. Problemi

Varianti per la ricerca in salita

- Per superare i problemi precedenti ci sono alcune tecniche:
  - tornare indietro e procedere verso una direzione leggermente meno promettente di quella scelta in precedenza
  - ripartire da un punto scelto a caso
  - fare un balzo (più mosse senza valutazione intermedia) verso una direzione particolare
  - non effettuare la valutazione dopo ogni mossa, ma dopo alcune mosse
- Purtroppo però la ricerca in salita è inadatta se la funzione ha comportamenti non lineari
- La ricerca in salita è un metodo **locale**

Alberto Broggi

Risol. Problemi

Ricerca in salita: esempio

Ricerca del centro della città:

- Se l'euristica è locale (determinazione della distanza con i grattacieli) è possibile che non si arrivi alla soluzione (in presenza di sensi unici)
- Se l'euristica viene definita utilizzando conoscenza globale, allora è possibile arrivare alla soluzione

Alberto Broggi

Risol. Problemi

Simulated Annealing

- L'algoritmo è così chiamato perché simula l'annealing, cioè il processo graduale di raffreddamento di un liquido fino al suo congelamento
- Se la temperatura viene diminuita abbastanza lentamente, allora il liquido raggiunge una configurazione a minima energia; altrimenti può portarsi in diversi stati di aggregazione

Alberto Broggi

Risol. Problemi

Simulated Annealing

- L'algoritmo di simulated annealing è molto simile all'algoritmo di ricerca in salita
- La differenza è che è possibile eseguire anche mosse che portano a stati peggiori
- Se la mossa migliora la situazione, allora viene eseguita, altrimenti viene eseguita con probabilità $e^{-\Delta E/T}$
  - ΔE è il valore di peggioramento causato dall'esecuzione della mossa
  - T è un valore che tende a zero con l'aumentare dei cicli e quindi rende sempre meno probabile l'esecuzione di mosse peggiorative
- Si deve tenere memorizzato lo stato migliore in assoluto incontrato durante tutto il ciclo

Alberto Broggi

Risol. Problemi

Simulated Annealing

- L'algoritmo di simulated annealing è definito come segue:
  - si parte dalla configurazione iniziale
  - si genera una mossa possibile
  - se la mossa porta a stati migliori, la si accetta
  - altrimenti
    - » si genera un numero casuale [0-1]
    - » se il numero è minore di $e^{-\Delta E/T}$, si accetta la mossa
    - » altrimenti non la si accetta
  - si calcola un nuovo valore di T dall'*annealing schedule*
  - si continua fino a che non è possibile effettuare altre mosse

Alberto Broggi

Risol. Problemi

I giochi

- I giochi sono stati uno dei primi campi in cui è stata applicata l'IA.
- La presenza di un avversario rende più complicato il problema della ricerca dato che l'avversario introduce incertezza.
- La caratteristica che spesso li distingue è il grandissimo spazio di ricerca.
- Ad esempio gli scacchi hanno un fattore di ramificazione circa uguale a 35 e le partite spesso arrivano a 50 mosse per giocatore.

Alberto Broggi

Risol. Problemi

I giochi

- Quindi per il gioco degli scacchi abbiamo uno spazio di ricerca di 35^{100} mentre le posizioni legali sono 10^{40}.
- Un gioco per due persone può essere definito come un problema di ricerca con le seguenti componenti:
  - lo stato iniziale;
  - gli operatori, le mosse legali;
  - un test di terminazione;
  - una funzione di utilità (utility or payoff function) indicante il risultato della partita.

Alberto Broggi

Risol. Problemi

L'algoritmo Minimax

- L'algoritmo Minimax calcola la mossa ottima per il primo giocatore (Max).
- L'algoritmo è composto dai seguenti passi:
  - genera l'albero completo del gioco;
  - applica la funzione di utilità a tutti gli stati terminali
  - partendo dagli stati terminali assegna in modo ricorsivo agli stati intermedi:
    - » il massimo della funzione di utilità degli stati figli se i figli corrispondono a una mossa di Max.
    - » il minimo della funzione di utilità degli stati figli se i figli corrispondono a una mossa di Min.
  - l'algoritmo termina con l'assegnazione della prima mossa a Max che è quella con la massima funzione di utilità tra le prime mosse possibili.

Alberto Broggi

Risol. Problemi

L'algoritmo Minimax

- L'algoritmo Minimax si basa sull'ipotesi che sia possibile generare tutti gli stati terminali.
- Questo anche per giochi non molto complicati può essere impraticabile.
- Quindi è necessario che l'algoritmo Minimax consigli una mossa senza dover esplorare tutto l'albero di ricerca.
- Questo può essere fatto introducendo una funzione euristica di valutazione, che valuterà la bontà anche di stati non terminali e di un test di terminazione (Cutoff-Test) per decidere a che livello di mosse terminare la ricerca.

Alberto Broggi

Risol. Problemi

L'algoritmo Minimax

- La funzione di valutazione fornisce una stima della bontà della mossa.
- Ad esempio negli scacchi la stima può essere effettuata in base alla quantità/qualità di pezzi dei due giocatori.
- Il test di terminazione cerca di tagliare la ricerca ad un livello d tale che sia rispettato il limite di tempo fissato dal gioco.
- Tuttavia un test di terminazione così semplice può causare grossi danni se accompagnato da una funzione di valutazione non troppo sofisticata.

Alberto Broggi

Risol. Problemi

L'algoritmo Minimax

- Per superare questi problemi bisogna arrestare la ricerca in configurazioni quiescenti, cioè in cui le prime mosse successive non cambieranno sensibilmente il valore della sua valutazione.
- Un problema difficilmente superabile è il problema dell'orizzonte, cioè prevedere mosse dell'avversario che per il limite allo spazio di ricerca sembrano essere impossibili.

Alberto Broggi

Risol. Problemi

Alpha-Beta Pruning

- L'associazione di un buona funzione di valutazione e un buon test per decidere dove tagliare l'espansione dell'albero di ricerca può non essere sufficiente per realizzare un programma Minimax che possa giocare realmente.
- È tuttavia possibile decidere una mossa corretta senza dover controllare tutti i nodi che l'algoritmo Minimax controlla. Questo viene fatto usando la tecnica della potatura alpha-beta:
  - se un giocatore può muoversi in un nodo n, ma ha una migliore scelta m in un nodo a profondità minore nell'albero, allora il nodo n non sarà mai raggiunto.

Alberto Broggi