Intelligenza Artificiale

Risoluzione di Problemi

Alberto Broggi

Risoluzione di Problemi

- Un problema è definito dal goal e dall'insieme di strumenti usati per raggiungere il goal.
- La prima fase della risoluzione di un problema è la **formulazione** del problema.
- In questa fase, oltre a dare una buona definizione del goal, bisogna decidere quali sono le azioni e qual'è lo spazio degli stati e quale è lo stato da cui si parte (stato iniziale).
- Cosa molto importante è scegliere delle azioni e quindi degli stati che permettano di ottenere delle soluzioni al giusto livello di dettaglio.

Risoluzione di Problemi: Esempio

- Sono dati 2 secchi non graduati da 3 e 4 litri e una pompa per riempirli. Come è possibile mettere 2 litri d'acqua nel secchio da 4 litri?

- Per la soluzione di devono definire:
  - **lo spazio degli stati**
  - **lo stato iniziale**
  - **il goal**
  - **le azioni**
  - **il meccanismo di controllo per applicare le regole**

Esempio

- Lo spazio degli stati:
  - coppie di valori (x,y) tali che x=0,1,2,3,4 ed y=0,1,2,3 indicano il numero di litri nei due secchi
- Lo stato di partenza:
  - (0,0)
- Lo stato obiettivo:
  - (2,n) per un qualsiasi valore di n (poiché il problema non specifica quanti litri devono trovarsi nel secondo secchio
- Gli operatori:
  - regole che portano ad un mutamento dello stato
  - è necessario aggiungere delle regole non specificate dall'enunciato
- Il controllo:
  - La scelta della regola e il ciclo influenzano la velocità di raggiungimento della soluzione

Esempio: regole di produzione

• se x<4	(x,y) -> (4,y)	Riempi il secchio da 4 litri
• se y<3	(x,y) -> (x,3)	Riempi il secchio da 3 litri
• se x>0	(x,y) -> (0,y)	Vuota il secchio da 4 litri
• se y>0	(x,y) -> (x,0)	Vuota il secchio da 3 litri
• se x+y>=4 e y>0	(x,y) -> (4, y-(4-x))	Versa l'acqua del secchio da 3 litri nel secchio da 4 litri finchè è pieno
• se x+y>=3 e x>0	(x,y) -> (x-(3-y), 3)	Versa l'acqua del secchio da 4 litri nel secchio da 3 litri finchè è pieno
• se x+y<=4 e y>0	(x,y) -> (x+y,0)	Versa l'acqua dal secchio da 3 litri in quello da 4 litri
• se x+y<=3 e x>0	(x,y) -> (0,x+y)	Versa l'acqua dal secchio da 4 litri in quello da 3 litri
• se x>0	(x,y) -> (x-d,y)	Versa un po' d'acqua dal secchio da 4 litri
• se y>0	(x,y) -> (x,y-d)	Versa un po' d'acqua dal secchio da 3 litri

Regole di produzione: commenti

- Le condizioni devono essere soddisfatte prima che l'operatore venga applicato
  - **aumentano l'efficienza del programma, anche se opzionali**

- Le ultime due regole formalizzano la possibilità di versare un quantitativo ignoto; è permesso dal problema, ma probabilmente non porterà alla soluzione
  - **differenza tra regole che descrivono il problema e regole che, oltre a descrivere il problema, aggiungono della conoscenza per raggiungere la soluzione**

Risoluzione di problemi

- Questo approccio va bene per problemi **semplici** (problemi artificiali e strutturati)
- Per i problemi **naturali** questo risulta più complesso (se non impossibile)
  - interpretare il significato di una frase
  - percezione audio/visiva
- Questi problemi necessitano di regole, ma anche di un'appropriata strategia di controllo per muoversi nello spazio degli stati (scegliere quali regole applicare)
  - Obiettivo: trovare un cammino da uno stato iniziale ad uno stato obiettivo

Risoluzione di Problemi

- Quindi bisogna trovare una sequenza di azioni, detta soluzione, che soddisfi il goal del problema.
- L'esecuzione di ogni azione ha un costo, questo costo può variare da azione ad azione e quindi ad ogni sequenza di azioni bisogna assegnare un costo in modo da scegliere la soluzione migliore.
- Quest'attività corrisponde alla seconda fase della risoluzione di un problema e viene individuata con il nome di ricerca.
- Infine, trovata la soluzione, nella terza fase, detta di esecuzione, le azioni sono eseguite per raggiungere il goal.

Risoluzione di problemi

- Importante:
  - Anche se usiamo la ricerca per arrivare alla soluzione, si possono sfruttare -dove possibile- anche tecniche più dirette

    Ad esempio: nel problema dei secchi d'acqua, il risultato di y-(4-x) si calcola, non si ricerca

- In generale, la ricerca è un metodo generale che può essere usato quando non è noto alcun metodo più diretto

Sistemi di produzioni

- La ricerca è il nucleo di molti processi "intelligenti"
- E' necessario trovare un modo per facilitare la descrizione e l'esecuzione del processo di ricerca
- I Sistemi di Produzioni servono a questo

Sistemi di produzioni

- Un sistema di produzioni è costituito da:
  - **un insieme di regole**: per modificare lo stato del problema
  - **una base di conoscenza**: contiene le informazioni sul problema
  - **una strategia di controllo**: specifica l'ordine con cui le regole vengono scelte per evitare conflitti nel caso più regole possano essere applicate contemporaneamente
  - **un sistema per applicare le regole**
- Ogni elemento ha la propria importanza, ma forse la parte fondamentale la gioca la strategia di controllo

Strategie di controllo

- L'obiettivo è raggiungere lo stato finale nel modo più efficiente possibile
  - tempo, numero di operazioni, quantità di dati elaborati...
- La strategia di controllo serve per decidere quale regola applicare, quando più di una regola ha la parte sinistra corrispondente allo stato attuale
- La scelta delle regole impatta sulla velocità e anche sulla possibilità di raggiungere la soluzione

Strategie di controllo

- Una buona strategia di controllo deve:
  - **causare movimento tra stati**: non ci si deve muovere tra pochi stati ma si deve spaziare
  - **essere sistematica**: se si sceglie a caso la regola è probabile trovarsi -durante il processo- in stati già visitati e quindi aver percorso passi non necessari

Ricerca in ampiezza (Breadth-First Search)

- Un algoritmo di ricerca in ampiezza si basa sul fatto che ogni nodo di profondità *d* è espanso prima di qualsiasi altro nodo di profondità maggiore.

- Si genera l'albero a partire dalla radice (stato iniziale) e si continua fintanto che qualche regola produce uno stato obiettivo

Ricerca in ampiezza

- Vantaggi:
  - Non viene mai intrappolata in un vicolo cieco, anche se forse puo' seguire un cammino inutile per lungo tempo
  - Se esiste una soluzione la ricerca in ampiezza la trova sicuramente
  - Se esistono piu' soluzioni, trovera' la soluzione minima (che richiede il minor numero di passi)
- Svantaggi:
  - E' problematica se vi sono cicli (esempio dei secchi d'acqua)

Ricerca in ampiezza

- Ottimalità: è ottima nel caso in cui
  - il costo è una funzione crescente con la profondità del nodo;
  - il costo degli operatori è uguale.
- Utilizza una strategia sistematica che considera prima le soluzioni di lunghezza 1, poi quelle di lunghezza 2 e così via.
- Se b è il fattore di ramificazione e d è la profondità della soluzione, allora il numero massimo di nodi espansi è:

  $1 + b + b^2 + b^3 + \ldots + b^d$

Ricerca in ampiezza

- Complessità temporale: b^d
- Complessità spaziale: b^d
- Ad esempio con b uguale a 10 e per ogni nodo un tempo di 1 millisecondo e un'occupazione di 100 byte avremo i seguenti valori:

Profondità	Nodi	Tempo	Memoria
0	1	1 millisec	100 byte
2	111	100 millisec	11 kilobyte
4	11111	11 secondi	1 Megabyte
6	10^6	18 minuti	111 Megabyte
8	10^8	31 ore	11 Gigabyte
10	10^{10}	128 giorni	1 Terabyte
12	10^{12}	35 anni	111 Terabyte
14	10^{14}	3500 anni	11111 Terabyte

Ricerca a costo uniforme (Uniform Cost Search)

- L'algoritmo di ricerca in ampiezza trova la soluzione col valore di profondità più basso.
- Se consideriamo il costo della soluzione, questa soluzione può non essere la soluzione ottima.
- L'algoritmo di ricerca a costo uniforme permette di trovare la soluzione a minor costo espandendo tra i nodi della frontiera il nodo a minor costo.
- Se il costo è uguale alla profondità del nodo il funzionamento di questo algoritmo è uguale all'algoritmo di ricerca in ampiezza.
- Ha la stessa complessità temporale e spaziale dell'algoritmo di ricerca in ampiezza.

Ricerca in profondità (Depth-First Search)

- L'algoritmo di ricerca in profondità ad ogni livello di profondità espande il primo nodo fino a:
  - raggiungere un goal o
  - raggiungere un punto in cui il nodo non può essere più espanso.

- Si analizza prima una parte dell'albero fino al fondo e poi le altre parti.

Ricerca in profondita'

- Vantaggi:
  - Richiede meno memoria (sono memorizzati solo i nodi del cammino corrente)
  - Per caso (o se la scelta e' accurata) puo' trovare una soluzione senza esplorare gran parte dello spazio di ricerca
- Svantaggi:
  - Se la soluzione e' molto vicina allo stato iniziale, puo' darsi che il tempo per raggiungerlo sia comunque elevato
  - Non usa nessun criterio per favorire le soluzioni a costo minore e quindi la soluzione proposta può non essere ottima.
  - Può imbattersi in un percorso di profondità infinita

Ricerca in profondità

- Se b è il fattore di ramificazione e m è la massima profondità dell'albero, allora il numero massimo di nodi espansi è:

  $1 + b + b^2 + b^3 + \ldots + b^m$

- Complessità temporale: b^m
- Complessità spaziale: bm

Ricerca in profondità limitata (Depth-Limited Search)

- Per superare il problema dell'algoritmo di ricerca in profondità nell'affrontare alberi con rami con un numero infinito di nodi si può imporre un limite alla profondità dei nodi da espandere.
- La scelta del limite dovrebbe essere condizionata dalla previsione della profondità della soluzione.

Ricerca in profondità limitata

- Non usa nessun criterio per favorire le soluzioni a costo minore e quindi la soluzione proposta può non essere ottima.
- Trova la soluzione nel caso in cui il limite sia superiore alla profondità della soluzione.
- Se b è il fattore di ramificazione e se fissiamo il limite di profondità dell'albero a l, allora il numero massimo di nodi espansi è:

  $1 + b + b^2 + b^3 + \ldots + b^l$

- Complessità temporale: b^l
- Complessità spaziale: bl

Ricerca iterativa in profondità (Iterative Deepening Search)

- Il problema principale dell'algoritmo di ricerca limitato in profondità è la scelta del limite a cui fermare la ricerca.
- Il modo per evitare del tutto questo problema è quello di applicare iterativamente l'algoritmo di ricerca limitato in profondità con limiti crescenti, cioè 0, 1, 2, ...
- In questo caso l'ordine di espansione dei nodi è simile a quello della ricerca in ampiezza con la differenza che alcuni nodi sono espansi più volte.

Ricerca iterativa in profondità

- Se b è il fattore di ramificazione e se la profondità della soluzione è d, allora il numero massimo di nodi espansi dall'algoritmo di ricerca in profondità iterativa:

  $(d+1)1 + (d)\, b + (d-1)b^2 + (d-3)\, b^3 + \ldots + 1b^d$

- Complessità temporale: b^d
- Per b = 10 e d = 5 rendendo iterativo l'algoritmo passiamo da 111111 a 123456 nodi espansi.
- Complessità spaziale: bd
- Questo metodo è il preferito quando lo spazio di ricerca è grande e la profondità della soluzione non è conosciuta.

Ricerca bidirezionale (Bidirectional Search)

- L'idea è quella di cercare contemporaneamente in avanti dallo stato iniziale e indietro dal goal.
- Se b è il fattore di ramificazione e d è la profondità della soluzione, allora la complessità temporale dall'algoritmo di ricerca bidirezionale è: $2b^{d/2}$ che può essere approssimato con: $b^{d/2}$.
- Per utilizzarlo bisogna superare alcuni problemi:
  - la ricerca all'indietro richiede di generare i predecessori di un nodo. Per fare ciò, gli operatori devono essere reversibili.
  - come si tratta il caso in cui ci sono diversi goal possibili?
  - bisogna controllare in modo efficiente che un nodo non sia stato trovato dall'altro metodo di ricerca
  - che tipo di ricerca viene utilizzata nei due sensi?