Università degli Studi di Perugia

Dipartimento di Matematica e Informatica

Corso di Laurea Magistrale in Informatica

Relazione 1' Esonero Intelligenza Artificiale

Studente:
Valerio Belli; Matricola: 283514
Studente:
Alessandro Bigiotti; Matricola: 281812

Professore:
Alfredo Milani

Introduzione

In questa relazione vedremo l'implementazione di due *Agenti* in linguaggio *Python*. L'ambiente in cui vengono caricati e testati gli agenti è una piattaforma realizzata con le librerie $Kivy - Python$.

Nell'ambito dell'$Artificial\ Intelligence$ un agente è un'entità in grado di perceperire l'ambiente circostante (attraverso sensori) ed esegue azioni (tramite attuatori).
Ai fini dell'esonero sono stati realizzati due agenti:

Reflex-Agent-Model-Based ($AgentXTypeOne$):dotato di una memoria limitata, gestisce un ambiente parzialmente osservabile e reagisce seguendo una serie di regole:
($percezione\_osservabile$, $stato\_corrente$) $->$ $azione$;

Utulity-Based-Agent ($AgentXTypeTwo$):dotato di memoria illimitata, gestisce un ambiente completamente osservabile. Dotato di un obiettivo:
Goal: Cerca di non perdere i punti guadagnati
Per cercare di raggiungere l'obiettivo si serve di una funzione *euristica*, basata sulle distanze euclidee tra i vari agenti, e sulle posizioni di tutte le celle visitate:

Best_Choise : Prendi la direzione libera che porta all'agente più distante;

Avarage_Choise : Prendi la prima direzione libera;

Back_Track : Se non ci sono direzioni libere torna alla precedente.

Stop : Ritorna nella posizione iniziale in stato $'NoOp'$.

In base alla quantità di memoria a disposizione gli agenti possono:

- Memorizzare le celle visitate (anche dagli altri agenti)
- Memorizzare la posizione di altri Agenti
- Memorizzare la posizione dei muri
- Muoversi in 4 direzioni

Vediamo il comportamento di due agenti che implementano, rispettivamente, le strategia sopra indicate.

Gli Agenti possono lavorare in ambiente con agente singolo, e in ambiente con più agenti.

1 Agente riflessivo Basato su Modello: AgentXTypeOne

L'agente *AgentXTypeOne* ha una memoria fissa. È un agente reattivo. Si basa sulle sue ultime 7 posizioni, il suo stato attuale e sulla posizione attuale degli altri agenti. Per progettare l'algoritmo (che procede in una *Blind Search*) ci siamo basati su una strategia "*greedy*", ossia:

- finchè trovi celle sporche in una direzione, continua su quella direzione!

Nel caso in cui l'agente non trova più celle sporche, o sbatte contro il muro, o trova altri agenti sul suo percorso, decide un'azione pseudo-casuale; ossia, controlla la posizione degli altri agenti, e sceglie una direzione casuale che lo porta su una cella in cui non ci sono altri agenti.
Mantiene un contatore che incrementa se pulisce e decrementa se passa su celle pulite; quando il contatore assume un valore < 0, l'agente si ferma.

La limitazione della memoria impedisce di poter sfruttare qualsiasi euristica, in quanto l'agente non conosce la storia delle azioni fatte, quindi non ha una base su cui poter operare una scelta.

Vedremo nella prossima sezione che con una memoria più ampia si possono costruire strategie di ricerca migliori.

2 Agente Basato su Utilità: AgentXTypeTwo

L'agente *AgentXTypeTwo* ha memoria illimitata. L'agente percepisce la presenza di altri agenti e memorizza:

- La storia delle sue posizioni
- La storia delle posizioni degli altri agenti
- I muri su cui è andato a sbattere
- L'albero di ricerca per effettuare backtracking

L'agente implementa una visita in profondità (DFS), procedendo con una ricerca informata ($Informed\ Search$).
Il calcolo dell'azione per lo spostamento avviene in 3 fasi, mutuamente esclusive. All'inizio cerca di effettuare uno spostamento in modo *euristico*, basato

sulla distanza tra gli altri agenti; se lo spostamento lo porta su una cella non valida, ossia una cella già visitata (da lui stesso o altri agenti) non esegue lo spostamente ed apre il primo nodo valido dello spazio di ricerca; se nello spazio di ricerca non trova soluzioni valide, va in backtracking, e risale l' albero di ricerca, in particolare:

Heuristic Move : controlla la distanza euclidea tra i vari agenti e sceglie di spostarsi verso quello più lontano; c è più probabilità di trovare celle sporche;

Can't Choose Heuristic : se lo spostamento scelto dall'euristica non è valido, allora apre il primo nodo valido non esplorato nello spazio di ricerca;

Can't Open a Node : se non ci sono movimenti validi (tutte le celle vicine sono già visitate), allora risale l'albero di ricerca in backtracking.

La funzione *euristica* si basa sul *goal*:"Cerca di minimizzare la visita di celle già pulite (cerca di perdere il minor numero di punti)".

Mantenendo in memoria le posizioni già visitate dagli altri agenti, riusciamo a potare l'albero di ricerca, evitando di passare su posizioni già pulite.
Se tutte le posizioni raggiungibili portano in celle già visitate, l'agente è in un vicolo cieco, allora esegue il passo di backtracking, e rotorna alla posizione precedente.