Monkey World

Carmine Dodaro, Simone Spaccarotella
{carminedodaro, spa.simone}@gmail.com

13 febbraio 2011

Sommario

Monkey World è un framework a singolo agente, per la simulazione di sistemi intelligenti. L'agente in questione è una scimmietta che può muoversi lungo un ambiente monodimensionale. Il suo obiettivo è quello di rubare un casco di banane appeso per aria e ritornare a cuccia. Essa ha a disposizione una cassa che può spostare liberamente, sulla quale dovrà salire per poter raggiungere il suo scopo. In Monkey World è stato implementato un comportamento che permettesse alla scimmietta di portare a termine la sua missione in maniera razionale.

1 Introduzione

Abbiamo un ambiente monodimensionale, composto da un nastro di dieci celle adiacenti, più una cella extra che rappresenta la cuccia della scimmietta. Questa cuccia può essere adiacente ad una sola cella del nastro e la sua posizione viene inizialmente settata da input, insieme alla cassa e al casco di banane.

Le azioni che può compiere l'agente scimmietta sono: uscire dalla cuccia, tornare a cuccia, spostarsi sul nastro verso sinistra/destra, spostare la cassa verso sinistra/destra, salire sulla cassa, rubare il casco di banane, scendere dalla cassa. Per spostare la cassa, o per salirci sopra, la scimmietta deve trovarsi nella sua stessa locazione. Lo scopo è quello di portare la cassa sotto il casco di banane, salirci sopra, rubare il casco, riportare la cassa al suo posto e ritornare a cuccia.

Sono previsti tre tipologie di ambiente. Un ambiente statico, in cui la scimmietta ha tutto il tempo di pianificare le sue mosse ed agire. Un ambiente dinamico, dove in questo caso il casco di banane si muove in una posizione random, allo scadere di un intervallo di tempo fissato da input. Infine un ambiente ancora dinamico, ma questa volta il casco di banane viene mosso dall'utente via interfaccia grafica, mediante l'utilizzo del mouse. In tutti e tre i casi, l'ambiente è deterministico, episodico, discreto, a singolo agente e completamente osservabile, il che significa che l'agente ha piena percezione della posizione di tutti gli oggetti in ogni momento.

2 Ambiente Statico

In questo tipo di ambiente, nulla cambia con il passare del tempo. Questo permette alla scimmietta di pianificare le sue mosse in maniera del tutto off-line, ed eseguirle in un secondo momento.

La pianificazione è stata fatta mediante l'utilizzo di *DLV-K*, un frontend di dlv che implementa K, un linguaggio di pianificazione basato sulla logica dei predicati, per la rappresentazione di conoscenza incompleta.

L'idea è la seguente. Viene calcolato solo metà piano, ovvero la parte in cui la scimmia esce dalla cuccia e ruba il casco di banane. Questa metà è stata suddivisa in due ulteriori sotto pianificazioni, una in cui la scimmietta deve raggiungere la cassa, ed un'altra in cui deve spostare la cassa fin sotto il casco di banane. In questo modo è stato possibile tagliare enormemente lo spazio di ricerca, diminuendo di conseguenza la complessità computazionale (in termini di tempo) del calcolo dell'intero piano.

Per l'interfacciamento con *DLV-K* è stata utilizzata la classe *ProcessBuilder* messa a disposizione dalla JDK. Grazie a questa classe è stato possibile lanciare DLV come un sotto processo. Dopodiché è bastato agganciarsi allo stream di output, parserizzare la stringa e creare una struttura dati da dare in pasto all'agente, sotto forma di lista delle azioni da compiere.

Come specificato in precedenza, il piano calcolato è solo metà. Questo perché la sequenza di azioni da compiere per riportare la cassa al suo posto e ritornare a cuccia è perfettamente speculare alla prima. E' bastato dunque, invertire la sequenza, e per ogni azione della prima fase, utilizzare la sua opposta.

Ad esempio: se nella prima fase la scimmia si sposta a destra per due volte, e poi sposta la cassa a sinistra per tre volte, nella seconda fase, invertendo la sequenza, la scimmietta dovrà prima spostare la cassa per tre volte, ma questa volta a destra, e poi dovrà spostarsi per due volte, ma questa volta a sinistra.

3 Ambiente dinamico

In questo tipo di ambiente, la banana è spostata in una posizione casuale alla scadenza di un timer impostato dall'utente. La scimmia deve, quindi, adattarsi di conseguenza alla nuova situazione dell'ambiente. In questo caso, la scimmia apprende da un'osservazione del mondo in più stati dopo quanto la banana è spostata.

Dopo questa prima fase di studio, ad ogni spostamento della banana, la scimmia calcola il numero di passi necessario a raggiungere la banana e confronta questo numero con il timer impostato. A questo punto, decide di spostarsi solo se riesce a prendere la banana nel tempo fissato.

Un'ulteriore raffinamento è il caso in cui la banana si sposta ad ogni passo. Infatti, normalmente la scimmia non riuscirebbe mai a prendere la banana in quanto per salire sulla scatola e per compiere l'operazione di presa sono necessari due passi. Quindi, abbiamo approntato una variante all'algoritmo che in questo specifico caso fa sì che la scimmia salga subito sulla scatola e aspetti che la banana arrivi sopra la sua posizione.

E' utile notare che per la legge dei grandi numeri, questo algoritmo consente alla scimmia di prendere sempre la banana.

4 Ambiente dinamico, con intervento dell'utente

Nell'ultimo caso sottoposto, il mondo evolve in questo senso: la banana è spostata dall'utente in un punto qualsiasi delle locazioni possibili e ad istanti di tempo

casuali.

E' importante notare, che in questo caso, la scimmia potrebbe non arrivare mai a prendere la banana. L'utente, infatti, ha sempre la possibilità di spostarla nel momento in cui la scimmia sta per prenderla.

L'agente è stato progettato di conseguenza. In una prima fase iniziale, segue l'andamento della banana sperando di riuscire a prenderla. Durante queste azioni, la scimmia tiene traccia di tutti gli spostamenti della banana, conteggiando il numero di volte in cui è stata spostata nella i-sima locazione. Quindi, dopo un certo numero di fallimenti, la scimmia decide che è il momento di fermarsi. Calcola la locazione che ha ospitato più volte la banana, e aspetta lì salendo sulla scatola. In questo tipo di ambiente, non abbiamo rassicurazioni sul fatto che la scimmia riesca a prendere la banana. Infatti, l'agente può solamente sperare che l'utente segua un pattern prefissato e che quindi alla fine l'algoritmo converga su un risultato.