Intelligenza Artificiale I

Progetto di tipo 2 - Mondo Accidentato

Barban Gabriele, Chiesa Samuele, Dell'Oro Matteo

27 marzo 2017

1 Specifiche del problema

Il progetto consiste nell'implementazione di un gioco. Il gioco è basato su un agente che si muove in un mondo quadrato.

Il mondo è costituito da caselle composte da diversi territori: aria, mare, foresta, deserto. Il passaggio da un territorio ad un altro ha un costo specifico pari a: 1 nel caso dell'**aria**, 2 nel caso dell'**acqua**, 3 nel caso della **foresta**, 4 nel caso del **deserto**.

Nel mondo si possono trovare inoltre degli oggetti, tali: **aereo** (senza il quale non si può andare in caselle di tipo aria), **barca** (senza la quale non si può andare in caselle di tipo mare) e **carro** (il quale abbassa il costo del movimento nel deserto da 4 a 2).

Un oggetto particolare è il **magnete**: se l'agente passa da una casella che lo contiene perde tutti gli oggetti che possiede al momento.

Il programma creato consente di trovare una soluzione esplorando il grafo con strategia A* e potatura dei chiusi; questa ricerca può essere resa più efficiente tramite l'utilizzo di euristiche.

2 Scelte Effettuate

L'**ambiente** viene modellizzato come una griglia formata da caselle. Ogni **casella** è composta dalla posizione, da un terreno e da una lista di oggetti. Il mondo in base dati dinamica può essere stampato **graficamente** o come **lista di caselle**.

Il mondo può essere **generato random** attraverso un algoritmo che dà diversi pesi all'estrazione di determinati terreni e liste di ogetti; in particolare deserto e foresta sono 2 volte più frequenti di cielo e mare, e la presenza di un magnete ha frequenza 1/10 mentre quella di uno degli altri oggetti ha frequenza 9/40. Gli oggetti possono essere presenti esclusivamente in caselle di tipo **deserto** o **foresta**; inoltre il carro è presente esclusivamente in terreni di tipo foresta.

I **fluenti** sono `in(posizione)`, che rappresenta la posizione dell'agente, e `possiede(oggetto)`

che rappresenta l'inventario dello stesso. Le **azioni** possibili sono `va(posizione)` e `prende(oggetto)` - il costo dei movimenti è definito nel paragrafo sopra mentre quello delle azioni è unitario.

Come **euristiche** sono state utilizzate la distanza di **manhattan** e quella di **euclide** tra la casella in cui viene calcolata l'euristica e quella di goal.

3 Testing sulle Azioni

Il testing sulle **azioni** è stato effettuato tramite il predicato `debug_action(<strategia>, <start>)`, in cui vengono elencati tutti i possibili cammini a partire da uno stato (`start`) ordinati secondo la strategia inserita. Tramite l'utilizzo di mondi salvati e soprattutto di mondi generati random è stato possibile verificare che l'agente compiva sempre tutte le possibili combinazioni di azioni a sua disposizione.

```
18 ?- debug_action(s:breadth_first,[in(p(1,1))]).
--------------------
Stato di partenza: [in(p(1,1))]
--------------------
* Lista azioni *
va(p(2,1))
--------------------
Cammino di lunghezza: 1
Stato finale: [in(p(2,1))]
true ;
--------------------
Stato di partenza: [in(p(1,1))]
--------------------
* Lista azioni *
va(p(1,2))
--------------------
Cammino di lunghezza: 1
Stato finale: [in(p(1,2))]
true ;
--------------------
Stato di partenza: [in(p(1,1))]
--------------------
* Lista azioni *
va(p(2,1))
va(p(2,2))
--------------------
Cammino di lunghezza: 2
Stato finale: [in(p(2,2))]
true ;
--------------------
Stato di partenza: [in(p(1,1))]
--------------------
* Lista azioni *
va(p(1,2))
va(p(2,2))
--------------------
Cammino di lunghezza: 2
Stato finale: [in(p(2,2))]
true
```

4 Testing sulle Euristiche

Il testing sulle **euristiche** è stato effettuato tramite i predicati `debug_heuristic(<euristica>)` e `debug_heuristic(<start>, <goal>, <euristica>)` in cui per ogni soluzione venivano stampati i valori di **costo** del path seguito, di **euristica** della casella di goal (0

naturalmente se fosse il goal stesso) e la **somma** dei due. Si è potuto verificare che, sia per l'euristica di manhattan che per quella di euclide, il costo della somma era ordinato in modo **crescente**, garantendo così l'ottimalità delle prime soluzioni e il buon funzionamento dell'euristica.

```
Nodo raggiunto con Costo(C) - Euristica(H) - Costo di frontiera(C+H): (15 - 4.47213595499958 - 19.47213595499958)
[in(p(2,4))]
X = p(2, 4) ;
Nodo raggiunto con Costo(C) - Euristica(H) - Costo di frontiera(C+H): (16 - 3.605551275463989 - 19.605551275463988)
[in(p(3,4)),possiede(carro)]
X = p(3, 4) ;
Nodo raggiunto con Costo(C) - Euristica(H) - Costo di frontiera(C+H): (16 - 3.605551275463989 - 19.605551275463988)
[in(p(3,4)),possiede(carro)]
X = p(3, 4) ;
Nodo raggiunto con Costo(C) - Euristica(H) - Costo di frontiera(C+H): (16 - 3.605551275463989 - 19.605551275463988)
[in(p(3,4)),possiede(carro)]
X = p(3, 4) ;
Nodo raggiunto con Costo(C) - Euristica(H) - Costo di frontiera(C+H): (14 - 5.656854249492381 - 19.65685424949238)
[in(p(2,2)),possiede(carro)]
X = p(2, 2) ;
```