Labirinto a più uscite

LABORATORIO PROLOG

Pietro Sangermano e Michele Metta

Obiettivo

E’ stato quello di scrivere un programma in Prolog in cui abbiamo modellato un agente in grado di trovare autonomamente il percorso migliore all’interno di un labirinto con più uscite.

	1	2	3	4	5
1	START				GOAL
2					
3					
4					
5					GOAL

Strategia

- Abbiamo sviluppato la nostra soluzione con l'approccio IDA* quindi utilizzando una ricerca in profondità con euristica.
- Abbiamo testato 2 euristiche principali (Manhattan e Euclidea) e loro combinazioni. Nello specifico abbiamo usato:
  - Distanza di Manhattan/Distanza Euclidea e combinazione delle distanze tramite semplice somma
  - Distanza di Manhattan/Distanza Euclidea e scelta della distanza minima fra le distanze calcolate
- Si fa noto il fatto che ovviamente avendo più uscite sui calcolano distanze diverse e il passo da fare verrà scelto nelle 2 modalità esposte (combinazione o minimo)

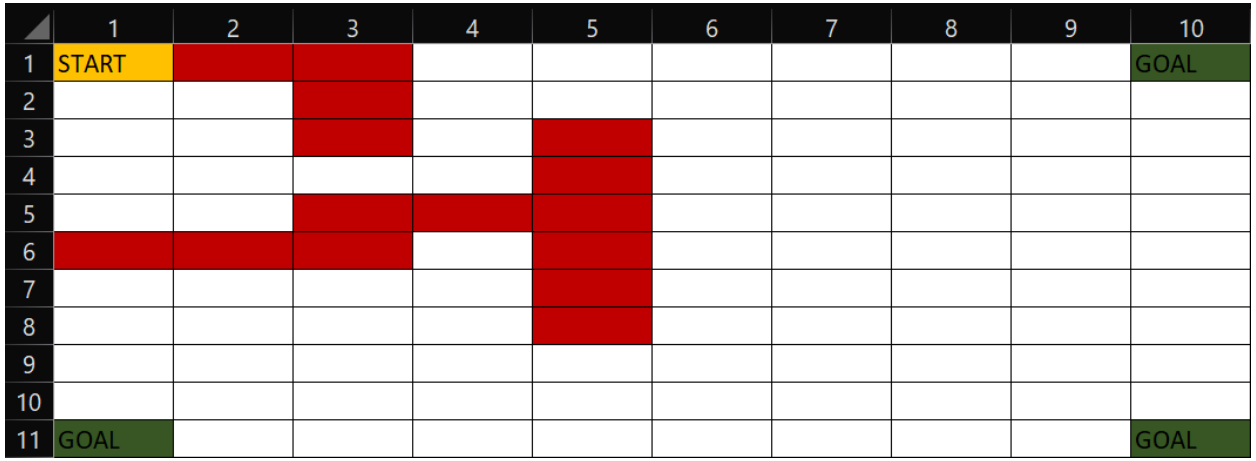
Struttura

- Il nostro programma è diviso in 3 file:
  - **Dominio.pl**, in cui viene definito lo start, i goals e gli ostacoli del labirinto
  - **Azioni.pl**, in cui vengono definiti le azioni applicabili (ci si può muovere nelle 4 direzioni cardinali) e i «trasforma» con cui effettivamente eseguiamo l'azione scelta (**N.B** per applicabilità noi intendiamo sia la non esistenza di un muro del bordo del labirinto sia di non essere già stati nella direzione in cui si vuole andare). Inoltre, è qui che vengono definite le 2 misure di distanza con cui viene valuta l'applicabilità di un'azione se i vincoli sono già stati rispettati.
  - **IDA star.pl**, questo è il fulcro centrale del programma in cui si è sviluppato il comportamento del nostro agente. Il suo funzionamento si basa sul predicato «risolvi» (che appunto trova il percorso verso l'uscita) che descrive 3 possibili situazioni:
    - *Caso base*: lo stato in cui ci troviamo è lo stato finale quindi ci fermiamo perché abbiamo la soluzione (vengono avviate routine della ricostruzione del cammino che verrà poi mostrato nel wrapper «prova»)
    - *Funzionamento Regolare*: Non ci troviamo nello stato finale. Vengono quindi trovate tutte le azioni applicabili (se ci sono), ordinate rispetto il loro valore di distanza **(«Fn»)** e l'azione applicabile meno costosa rispetto alla nostra euristica verrà «trasformata». A questo punto verrà richiamato ricorsivamente «risolvi» passandogli lo stato in cui siamo, il cammino fatto fin'ora, la lista degli stati già visitati, il numero di passi totali fatti fino a quel momento **(«Gn»)** e altri 3 parametri che ci permettono poi infine di avere diverse stampe del nostro cammino.
    - ***Backtracking:*** Siamo in vicolo cieco e non abbiamo nessuna azione applicabile. Se lo stato in cui siamo non è finale riprendiamo la lista degli stati già visitati e controlliamo, procedendo dalla testa, se negli stati visitati precedentemente ci sono percorsi non battuti, fermandoci al primo trovato. Questo viene fatto tramite un «for» artigianale che scansiona la lista e cerca lo stato da cui ripartire. Una volta trovato (se esiste altrimenti verrà restituito false) verrà richiamato il «risolvi» passandogli lo stato da cui deve ripartire e tornando quindi al funzionamento regolare.

Esperimento 1

Il primo esperimento che vogliamo proporre si basa su questo labirinto che verrà confrontato sia con il metodo illustrato precedentemente sia con la versione in profondità con la misura di deep mostrata a lezione:

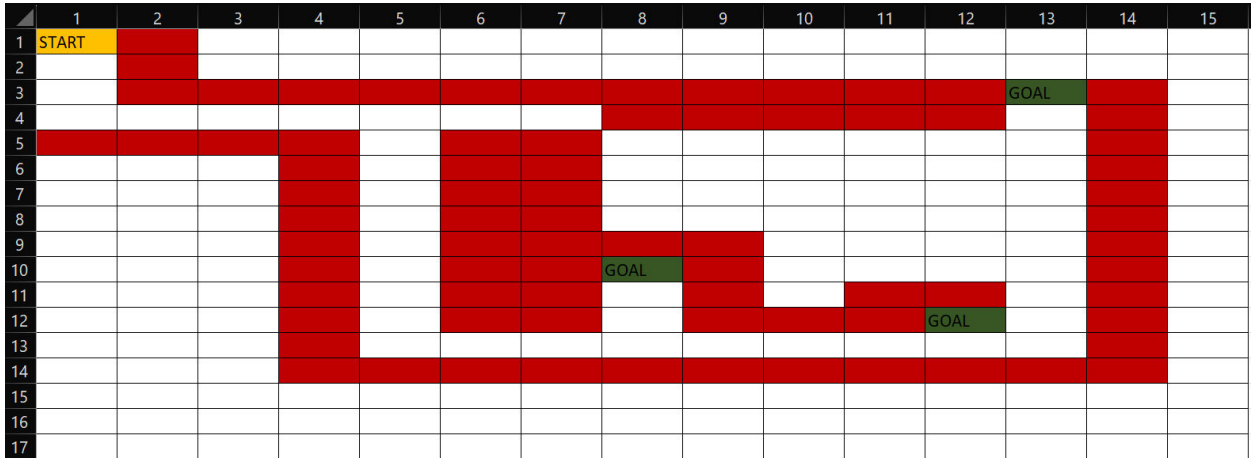
- Manhattan + combinazione => 22 passi, cammino: [giu,destra,giu,giu,destra,destra,su,su,destra,destra,destra,destra,destra,destra,giu,giu,giu,giu,giu,giu,giu,giu]
- Euclidea + combinazione => 38 passi, cammino: [giu,destra,giu,giu,destra,destra,su,su,destra,destra,giu,giu,giu,giu,destra,giu,destra,giu,sinistra,giu,destra,destra,su,su,su,sinistra,su,sinistra,su,destra,destra,giu,destra,giu,giu,giu,giu,giu]
- Manhattan + scelta della distanza minima: =>17 passi, Cammino: [giu,giu,giu,giu,destra,su,destra,destra,su,su,destra,destra,destra,destra,destra,destra,su]
- Euclidea +scelta della distanza minima => 17 passi, Cammino: [giu,giu,giu,giu,destra,su,destra,destra,su,su,destra,destra,destra,destra,destra,destra,su]
- Soluzione «itdeepening» => 15 passi, Cammino: [giu,giu,giu,giu,destra,su,destra,destra,su,su,destra,destra,destra,destra,destra,destra,su]
- Osservazioni:
  - La migliore soluzione possibile è l'itdeepening. Questo perché battendo tutte le soluzioni possibili (scalando per profondità), essendo la soluzione raggiungibile con pochi passi riesce a trovare facilmente la migliore.
  - Al secondo posto abbiamo le 2 distanze + la scelta della distanza minima. Probabilmente lo scostamento dei 2 passi rispetto alla itdeepening dipende dal fatto che l'euristica tiene conto della distanza fra tutte e 3 le uscite mentre itdeepening no e questo tende a ad allungare il percorso anche se di poco.
  - L a differenza più sostanziale l'abbiamo fra le prime 2 distanze. Questo è additabile alla struttura del labirinto e della disposizione dei goal rispetto allo start che favorisce la manhatta in quanto 2 di queste sono in direzione ortogonale allo start e anche perché la distanza euclide tiene conto delle diagonali che potrebbe portare (come si vede a distanze più lunghe). Si aggiunge al fatto che la combinazione lineare si è rivelata, anche nelle successive prove manchevole in termini di discriminazione precisa della direzione da prendere.



Esperimento 2

Il nuovo esperimento è con un labirinto 15 x 15 con gli stessi confronti (il labirinto è più articolato):

- Manhattan + combinazione => 26 passi, Cammino: [giu,giu,giu,destra,destra,destra,destra,destra,destra,sinistra,sinistra,giu,giu,giu,giu,giu,giu,giu,giu,giu,destra,destra,destra,su,su,su]
- Euclidea + combinazione =>26 passi, Cammino: [giu,giu,giu,destra,destra,destra,destra,destra,destra,sinistra,sinistra,giu,giu,giu,giu,giu,giu,giu,giu,giu,destra,destra,destra,su,su,su]
- Euclidea + scelta della distanza minima => 22 passi, Cammino: [giu,giu,giu,destra,destra,destra,destra,giu,giu,giu,giu,giu,giu,giu,giu,giu,destra,destra,destra,su,su,su]
- Manhattan + scelta della distanza minima => 22 passi, Cammino: [giu,giu,giu,destra,destra,destra,destra,giu,giu,giu,giu,giu,giu,giu,giu,giu,destra,destra,destra,su,su,su]
- Soluzione «itdeepening» => 22 passi, Cammino:[giu,giu,giu,dx,dx,dx,dx,giu,giu,giu,giu,giu,giu,giu,giu,giu,dx,dx,dx,su,su,su
- Osservazioni:
  - Questa volta i 2 metodi con la combinazione si equivalgono, probabilmente a causa della conformazione del labirinto e del fatto che tutti i goal sono concentrati in una zona del labirinto. Ma la cosa più interessante è che l'agente sbaglia esplorando anche (4,6) e (4,7) per poi tornare indietro facendo 4 passi in più rispetto alla soluzione ottimale. Questo problema viene superato nel momento in cui, accoppiato alla distanza euclidea (stessa cosa per la distanza Manhattan), prendiamo il minimo delle distanze che farà scegliere di andare giù quando si arriva in posizione (4,5).
  - Per quanto riguarda itdeepening , per quanto possa sembrare eguagliato dalla nostra soluzione, c'è da ricordare che per arrivare lì ha dovuto vagliare tutti i path fino alla profondità 22 utilizzando un approccio brute force cosa che noi abbiamo evitato con la nostra euristica



Conclusioni

- L’agente da noi creato riesce ad uscire autonomamente dal labirinto, scegliendo la migliore soluzione rispetto alla posizione iniziale.
- Abbiamo notato che la combinazione lineare delle distanze confonde molto l’agente che si «incastra» molto più facilmente e sceglie direzioni che molto spesso lo costringono a fare backtracking.
- L’ euristiche che abbiamo utilizzato, per quanto alcune volte non riescono esattamente ad arrivare alla soluzione ottimale ma solamente ad una che comunque le sia avvicina molto, riesce ad evitare di provare tutti i path possibili fino ad ottenere una soluzione, riuscendo ad orientarsi discretamente bene anche in labirinti abbastanza complessi.
- Il backtracking è limitato a quando l’agente, fuorviato dalla sua euristica, sbaglia ad un certo punto la scelta della direzione ma tornando indietro riesce a trovare l’ultima posizione nella quale è applicabile un’azione che non aveva ancora considerato.

Grazie Per l’Attenzione