Relazione IALab - Prolog

Roger Ferrod, Pio Raffaele Fina, Lorenzo Tabasso

Dipartimento di Informatica, Università degli Studi di Torino
roger.ferrod@edu.unito.it,
pio.fina@edu.unito.it,
lorenzo.tabasso@edu.unito.it

1 Introduzione

La seguente relazione illustra la realizzazione del progetto d'esame di Prolog. L'obiettivo del progetto consiste nello sviluppo di un *agente pathfinder* utilizzando il linguaggio di programmazione *Prolog*. Gli algoritmi di pathfinding utilizzati si basano su tre strategie di ricerca: una non informata e due con euristiche.

Dato in input una rappresentazione del labirinto, una posizione iniziale ed una posizione goal, l'agente muovendosi nello spazio ha l'obiettivo di trovare l'uscita del labirinto. L'agente può muoversi solo in quattro direzioni: nord, sud, est e ovest (*Von Neumann neighborhood*). Non può muoversi in diagonale e non può raggiungere una cella contenente un ostacolo. Inoltre, la configurazione del labirinto (dimensioni e posizione degli ostacoli) è **completamente conosciuta**.

Le strategie di ricerca implementate per risolvere tale problema sono:

- **IDDFS** (Iterative Deepening Depth-First Search): strategia di ricerca non informata molto efficiente, all'interno della quale è eseguita ricorsivamente una ricerca in profondità limitata, con un progressivo aumento del limite sino al raggiungimento di d, ossia la profondità più piccola in cui trovare lo stato goal.
- **A***: la strategia di ricerca basata su euristica più nota in letteratura. Calcola il costo per raggiungere la soluzione secondo la formula: $f(n) = g(n) + h(n)$, dove $g(n)$ rappresenta il costo per raggiungere il nodo n dal nodo di partenza, mentre $h(n)$ è la stima del costo del percorso più economico per raggiungere il goal a partire dal nodo n.
- **IDA*** (Iterative Deepening A*): strategia di ricerca nata per ridurre la complessità spaziale di A*. Intuitivamente adatta l'algoritmo *IDDFS* integrando una funzione costo $f(n) = g(n) + h(n)$, anzichè basarsi unicamente sulla profondità.

2 Esperimenti

Gli esperimenti che mostriamo in questa sezione sono stati eseguiti con la versione *swi-prolog 8.2.2* su una macchina con le seguenti caratteristiche:

```
CPU: Intel Core i7-4278U (2 core con 2 thread)
RAM: 16 GB
```

Gli esperimenti hanno l'obiettivo di:

- valutare il tempo d'esecuzione (Esperimento 2.1)
- valutare completezza degli algoritmi (Esperimento 2.2)
- valutare comportamento in presenza di più soluzioni (Esperimento 2.3)
- valutare dimensioni ed espansione della frontiera in A* (Esperimento 2.4)
- valutare l'impatto delle euristiche utilizzate in A* (Esperimento 2.5)

In base alla natura dell'esperimento sono stati definiti differenti labirinti, mostrati in Figura 1, a cui si aggiungono un labirinto casuale 160x160 e una mappa completamente priva di ostacoli. Laddove la configurazione della mappa non è particolarmente rilevante ai fini dell'esperimento, i labirinti sono stati generati in modo casuale utilizzando la tecnica *Randomized depth-first search*. Al contrario, nei casi in cui l'obiettivo sia incentrato sulla valutazione del comportamento dell'algoritmo in alcune configurazioni particolari, i labirinti sono stati creati manualmente utilizzando un'applicazione web sviluppata *ad hoc*.

2.1 Esperimento 1

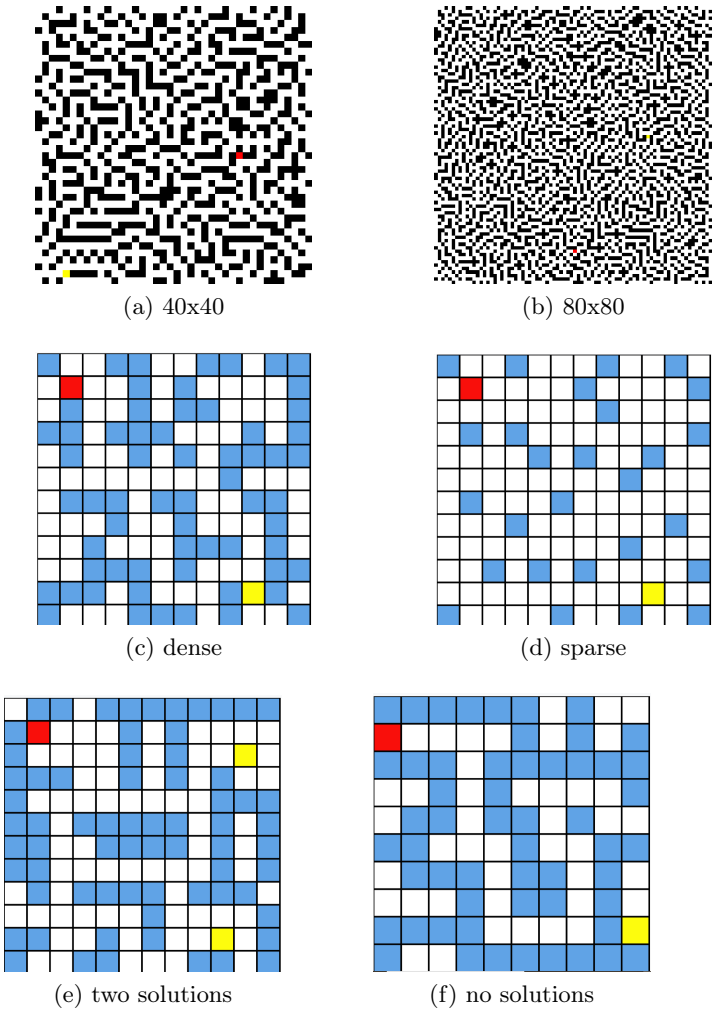
Sebbene la complessità, in termini asintotici, dei tre algoritmi sia equivalente e pari a $O(b^d)$, dove b rappresenta il branching factor e d la profondità a cui si colloca la soluzione, i tempi d'esecuzione possono variare notevolmente. In particolare, in riferimento alla Tabella 1, si può notare come la ricerca tramite IDDFS risulti la più veloce. Questo fenomeno può essere anche spiegato considerando il funzionamento dell'interprete Prolog e le molteplici integrazioni introdotte in IDA* e A*. Ad esempio, la sola implementazione di A* conta 42 regole applicabili, contro le 5 di iddfs.

Si presti inoltre attenzione alla complessità spaziale[1] che permette ad A* di risolvere il problema in minor tempo ($\approx$ 38 min) rispetto al più semplice IDA* ($\approx$ 2 hr). A tal proposito è utile ricordare che, oltre all'ottimalità della soluzione, ad A* è associata anche l'efficienza ottima; ciò significa che, a parità di euristica utilizzata, A* risulta l'algoritmo più efficiente in termini di numero di percorsi espansi.

Labirinto	IDDFS	IDA*	A*
40x40	2,92	13,5	25,5
80x80	4,75	98,09	17,71
160x160	971,64	7001,37	2264,12

Tabella 1. Risultati dell'esperimento 1. Sono riportati i tempi di esecuzione (in secondi) di ciascun algoritmo. IDA* e A* utilizzano come euristica la distanza euclidea.

[1] $O(d)$ in IDA* e $O(b^d)$ in A*



(a) 40x40 (b) 80x80

(c) dense (d) sparse

(e) two solutions (f) no solutions

Figura 1. Labirinti utilizzati negli esperimenti. I labirinti (a) e (b) sono stati generati automaticamente, i restanti manualmente.

2.2 Esperimento 2

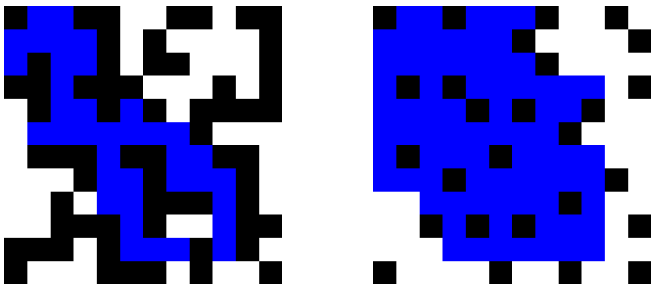
Come ben noto dal punto di vista teorico, l'algoritmo IDDFS non termina in presenza di labirinti senza soluzione (Figura 1f). Non avendo impostato una soglia massima infatti, l'algoritmo procede incrementando infinite volte la profondità massima a cui cercare una soluzione. Al contrario, gli algoritmi IDA* e A* forniscono la garanzia di terminazione se lo spazio di ricerca è finito, ritornando *false* in assenza di soluzione.

2.3 Esperimento 3

Per quanto concerne la possibilità di trovare più soluzioni, occorre ricordare che tutti gli algoritmi implementati sono *goal oriented* e non possono, per definizione, cercare contemporaneamente percorsi diversi, a meno ovviamente di eseguire più ricerche in parallelo o in serie. Di conseguenza, l'unica soluzione calcolata dagli algoritmi è determinata dal funzionamento dell'interprete stesso[2]. La ricerca di più goal (*Multiple-Goal Heuristic Search*) dev'essere tratta con un'apposita classe di algoritmi.

2.4 Esperimento 4

Poiché in A* è disponibile una rappresentazione esplicita della frontiera di ricerca[3], abbiamo analizzato la sua evoluzione al variare dell'ambiente di gioco. Graficamente (Figura 2) è evidente come, nel caso sparso, la frontiera si espanda maggiormente, mentre si mantiene più concentrata in presenza di ostacoli.



(a) Dense (b) Sparso

Figura 2. Percorsi visitati da A* in un labirinto denso (a) e sparso (b).

2.5 Esperimento 5

Analizzando più in dettaglio l'esecuzione di A* sulla mappa 40x40, è possibile apprezzare come il comportamento dell'algoritmo cambi a seconda della metrica utilizzata per calcolare l'euristica. In Figura 4 sono riportati gli esperimenti svolti, rispettivamente, con norma 1, 2 e infinito. Si può osservare, in riferimento all'ultimo istante temporale, il progressivo aumento del numero di percorsi esplorati. In particolare, la distanza calcolata con $||.||_\infty$ risulta essere la più dispersiva, con ovvie ripercussioni sul tempo d'esecuzione. La terminazione avviene infatti

[2] A tal proposito ricordiamo la risoluzione SLD di Prolog si basa sull'ordine della clausole, cosi come listate nel codice sorgente.

[3] Detta anche *open set* o *fringe* è realizzata grazie ad una coda di priorità.

in 30 secondi con distanza di Chebyshev, 15 sec con distanza euclidea e 13 sec con distanza di Manhattan.

La Figura 4 riporta solo 3 istanti di esecuzione dell'algoritmo; ulteriori considerazioni sull'evoluzione del percorso possono essere effettuate osservando una versione dinamica della Figura 4. A tal proposito viene allegato alla relazione un video contenente l'esecuzione dell'algoritmo su differenti labirinti.

In Figura 5 è inoltre disponibile un'analisi più accurata circa l'evoluzione della frontiera al variare della metrica utilizzata. In particolare si può osservare come la soluzione ottima dipenda dall'euristica presa in considerazione[4].

2.6 Labirinto 160x160

Considerando tutti i risultati ottenuti negli esperimenti effettuati sinora, riportiamo in Figura 3 la soluzione trovata dall'algoritmo A* sul labirinto casuale 160x160, utilizzando la distanza euclidea per il calcolo dell'euristica. Per ulteriori considerazioni rimandiamo alla visione delle animazioni video che accompagnano la relazione.

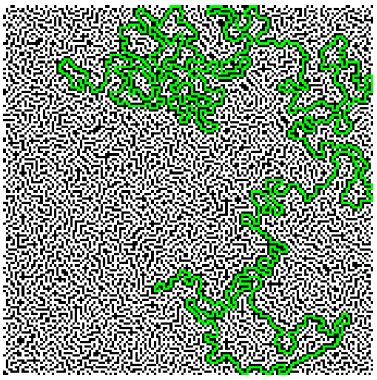


Figura 3. Soluzione del labirinto casuale 160x160 individuata da A*.

[4] A tal proposito ricordiamo che IDA* e A* sono algoritmi ottimi se l'euristica utilizzata è ammissibile.

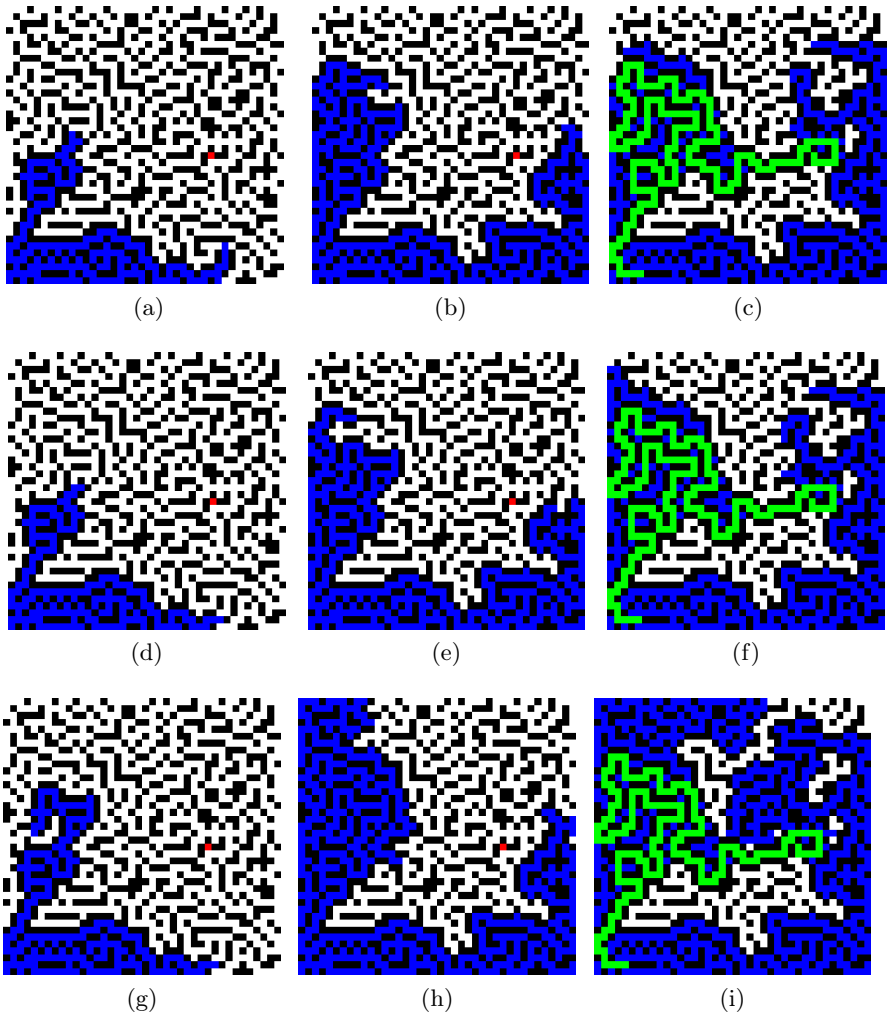


Figura 4. Evoluzione temporale di A* al variare dell'euristica. Nelle Figure 4a,4b,4c è utilizzata la norma $||.||_1$. Nelle Figure 4d,4e,4f è utilizzata la norma $||.||_2$. E infine, nelle Figure 4g,4h,4i è utilizzata la norma $||.||_\infty$.

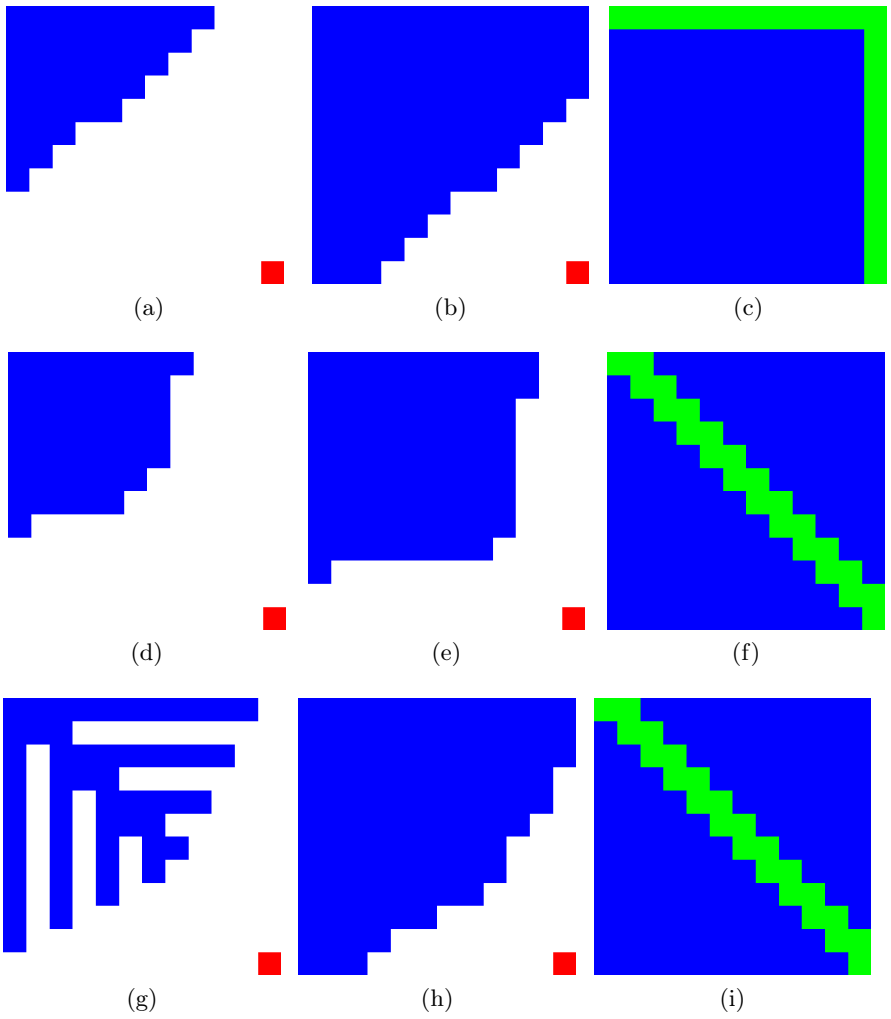


Figura 5. Evoluzione temporale di A* al variare dell'euristica. Nelle Figure 5a,5b,5c usando la norma $||.||_1$. Nelle Figure 5d,5e,5f usando la norma $||.||_2$. E infine, nelle Figure 5g,5h,5i usando la norma $||.||_\infty$.