Relazione progetti Prolog e Clingo

Andrea Cacioli Matricola: 914501

[TOC]

Prolog

In questa prima fase si é sperimentato l'utilizzo del paradigma di programmazione logica in prolog. Il progetto consiste nello sviluppo di un codice in grado di risolvere il problema del labirinto attraverso diverse strategie di ricerca nello spazio degli stati.

File prodotti (prolog)

File principali

- astar: file di definizione della strategia di ricerca informata A*
- azioni: file di definizione azioni possibili e applicabilitá di esse
- bfs: file di definizione strategia blind di ricerca in ampiezza
- heuristic: file di definizione di eventuali euristiche per la ricerca informata
- profonditá: file di definizione della strategia di ricerca blind in profonditá e iterative deepening.

Labirinti

- lab1.pl: file creato in classe di test 10x10
- zigzag15x15.pl: labirinto in figura
- hard20x20.pl: labirinto in figura
- hard15x15.pl: labirinto pensato per mandare in confusione A*
- easy17x17.pl: labirinto classico con molti blocchi
- easy25x25.pl: labirinto classico con molti blocchi
- MultipleExits.pl: labirinto con numerose uscite

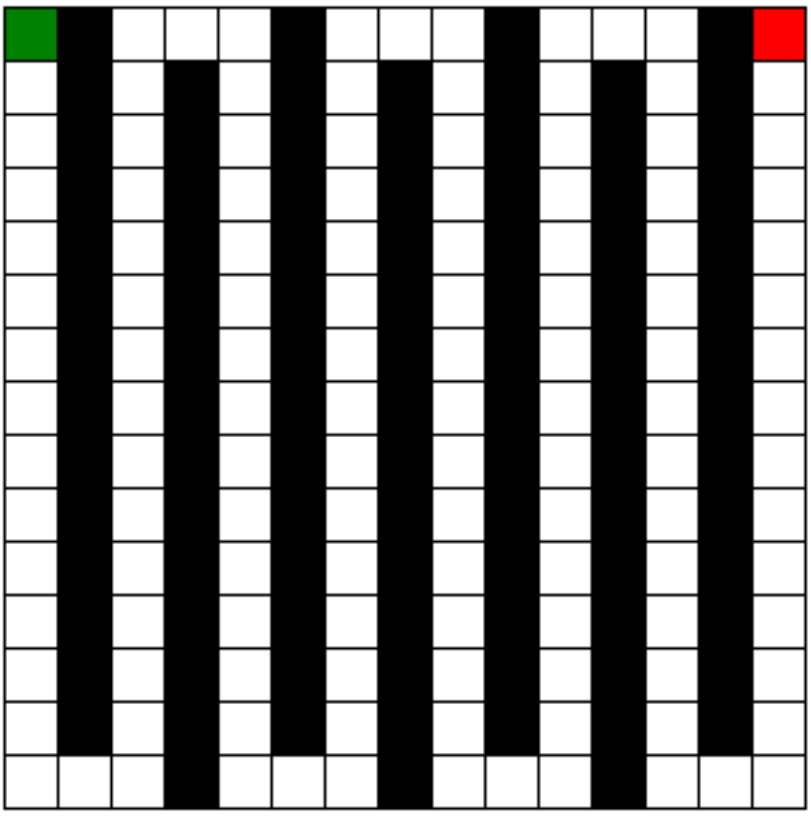

Figure 1: zigzag15x15.pl

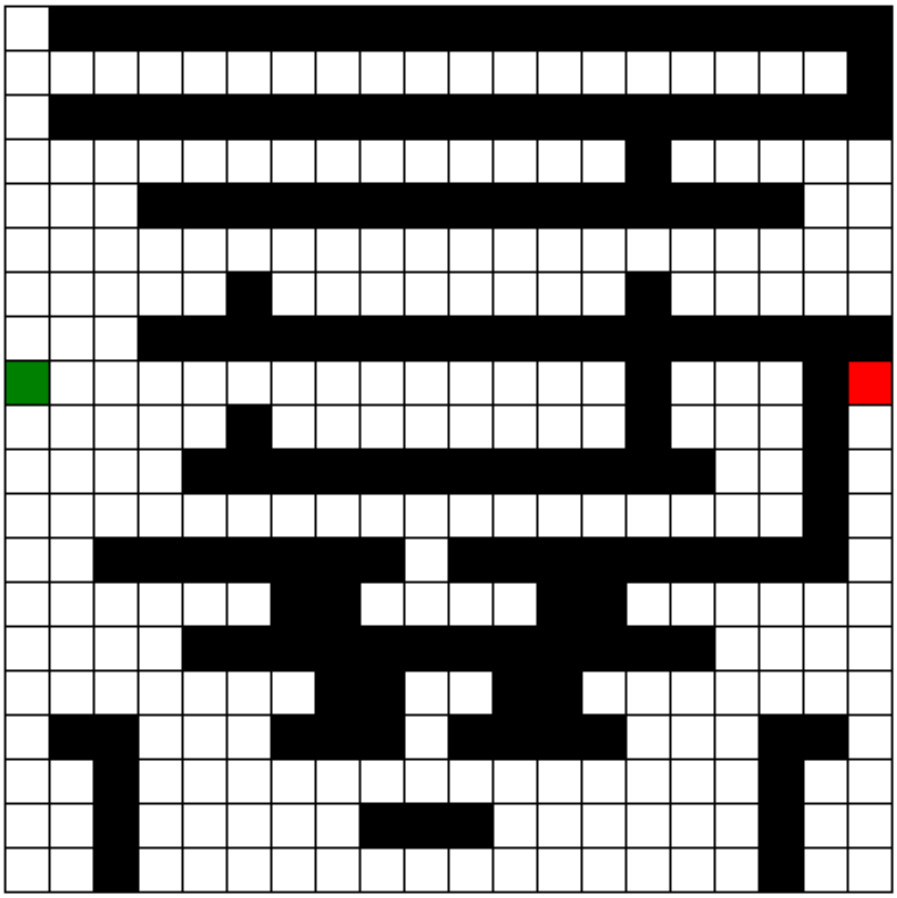

Figure 2: hard20x20.pl

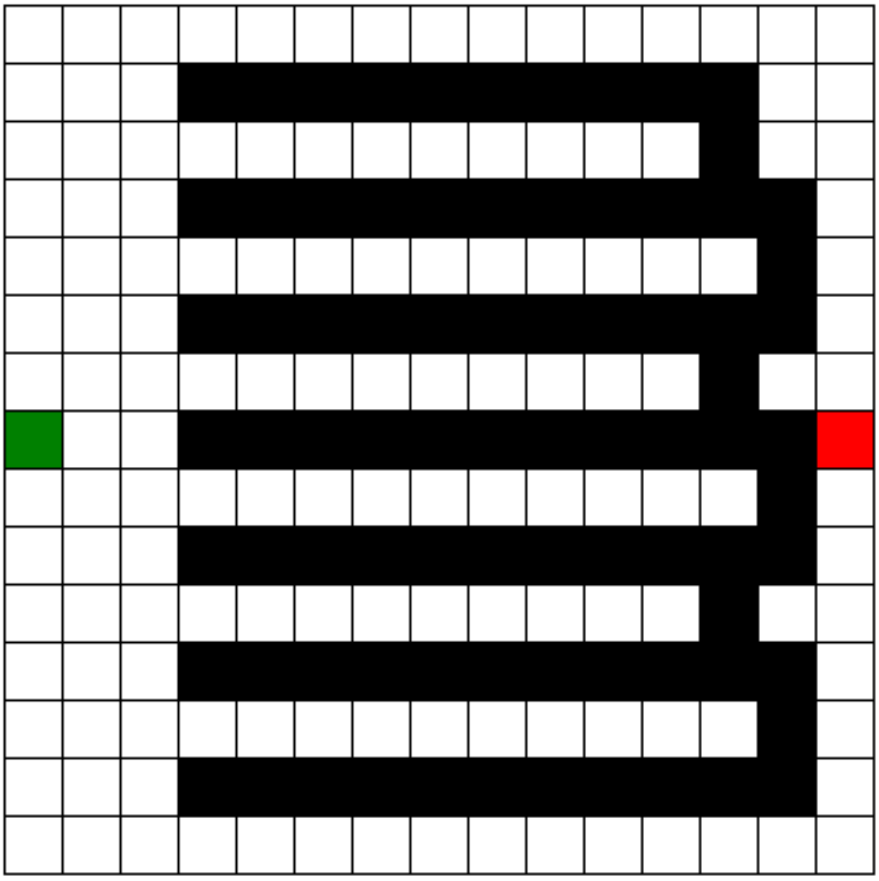

Figure 3: hard15x15.pl

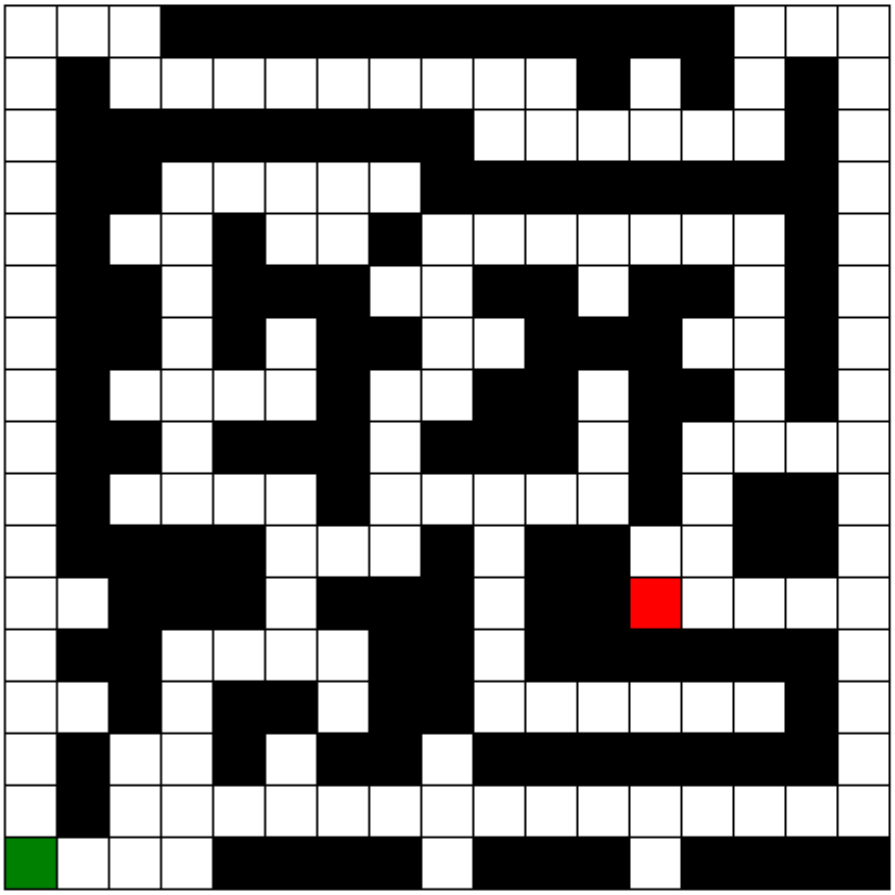

Figure 4: easy17x17.pl

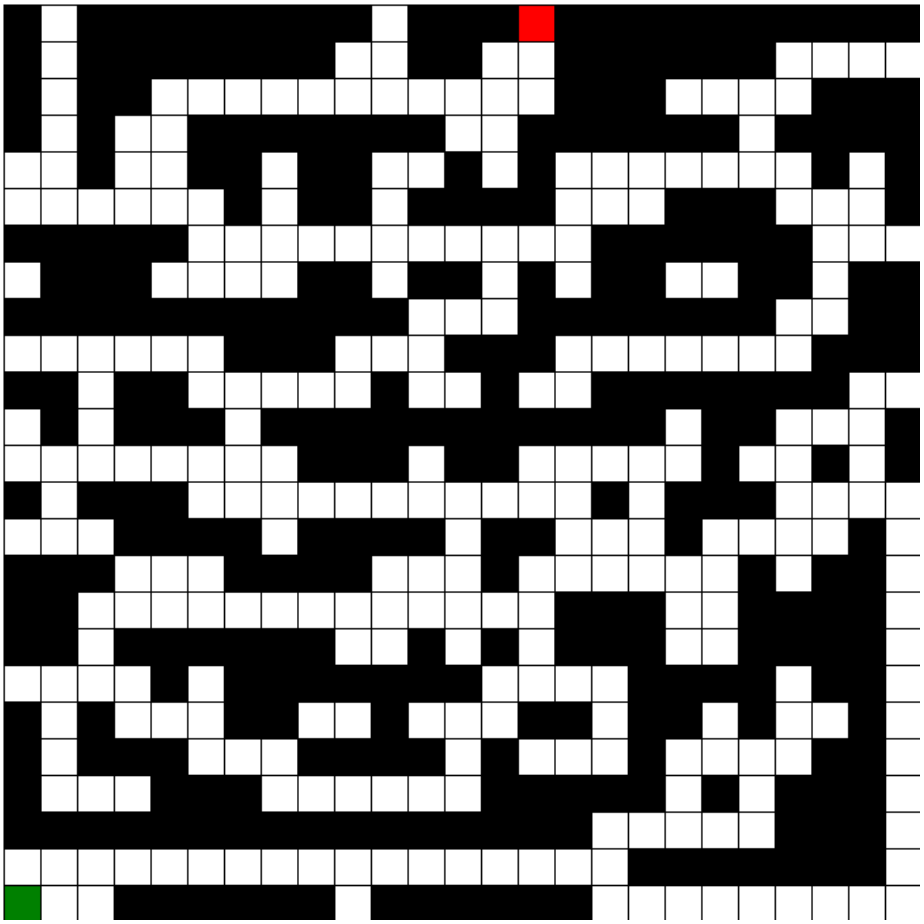

Figure 5: easy25x25.pl

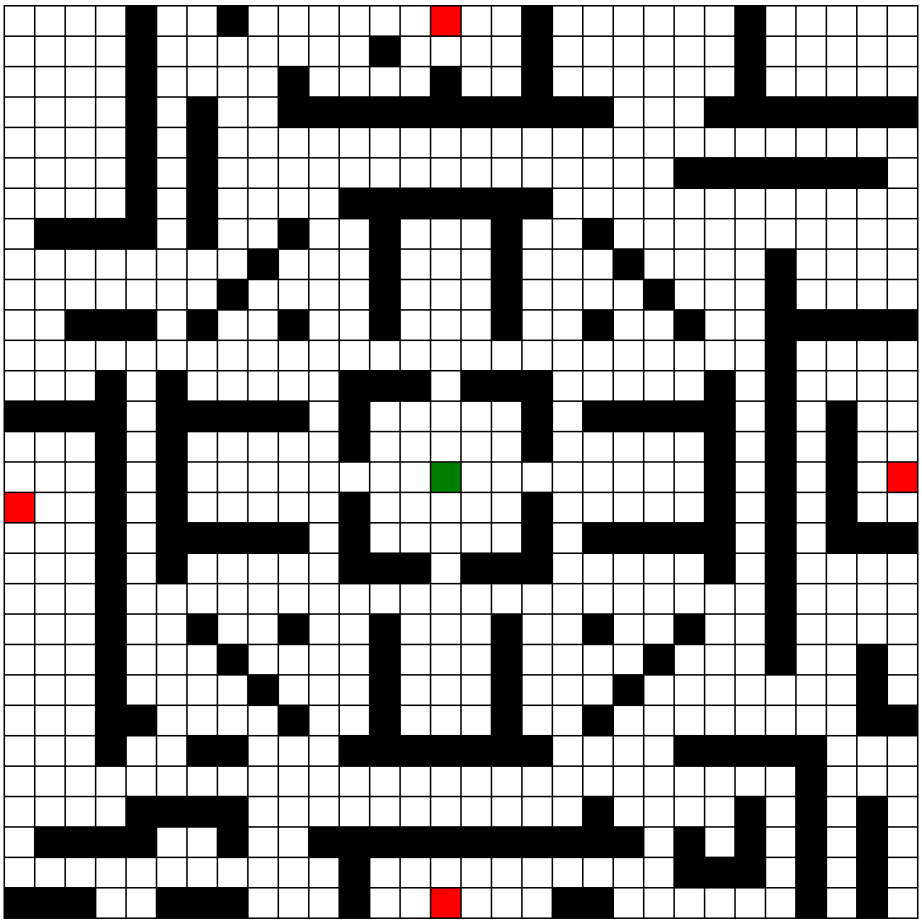

Figure 6: MultipleExits.pl

File Benchmarker Tale file serve solamente a provare le varie strategie su un labirinto precedentemente specificato.

Utilizzo Per far partire le seguenti strategie si carichi tramite consultazione il labirinto che si desidera testare e si proceda come segue:

1. Ricerca Iterative Deepening
   - Caricare i file ['prof.pl']
   - Dimostrare con prolog il fatto start.
2. Ricerca in profonditá
   - Caricare i file ['prof.pl']
   - asserire una profonditá massima n grande a piacere con assert(maxProf(n))
   - Dimostrare con prolog i fatti prova(Cammino), write(Cammino).
3. Ricerca in ampiezza
   - Caricare i file ['bfs.pl']
   - Dimostrare con prolog il fatto start.
4. Ricerca con A*
   - Caricare i file ['astar.pl']
   - Dimostrare con prolog il fatto start.

Performance Tabella dei test effettuati: I = Instantaneous (probably 10 to 20 milliseconds) DNF = Did Not Finish

Labirinto	A*	Ampiezza	Iterative Deepening (MaxProf)	Profonditá
lab1	I	I	I (16)	I
zigzag15x15	I	I	I (126)	I
easy17x17	I	I	I (25)	I
easy25x25	I	390 ms	51.313 s (98)	15 ms
hard15x15	I	47	22.297 s (28)	I
hard20x20	1.906 s	67.188 s	DNF (> 3h) (33)	6.078 s
Multiple Exits	5 ms	27.449 s	612.5 s (27)	1 ms

Osservazioni

- La strategia in profonditá é molto buona quando non deve trovare soluzioni ottime, infatti ha fatto molto bene in tutti i labirinti, tuttavia la lunghezza

del cammino in hard20x20 era di ben 87 contro i 37 della strategia in ampiezza.
- Non sono riuscito a disegnare un labirinto che mettesse in difficoltá seriamente A*
- Iterative Deepening, sebbene trovi sempre la soluzione ottima in termini di numero di passi, é la piú lenta in termini di tempo di esecuzione.
- In generale i labirinti con pochi muri risultano piú dispendiosi da calcolare per le strategie che trovano una soluzione ottima, eccetto A* grazie alla sua euristica.
- Labirinti che all'apparenza sembrano molto difficili da risolvere per noi esseri umani sono in realtá molto piú semplici per la macchina perché diminuiamo il branching factor dell'albero di ricerca nello spazio degli stati.

Clingo

Nel progetto di Asp é richiesto di scrivere un programma clingo in grado di generare un calendario per una competizione sportiva. Tale calendario é esprimibile in linguaggio clingo tramite una serie di vincoli di integritá e di aggregati.

File prodotti (clingo)

I file del progetto sono diversi ma di due tipi principali:

1. campionato_vincoli.cl: il file contenente tutti i vincoli che prescindono dal numero di squadre presenti nel campionato.
2. campionato[X].cl: diversi file in cui al posto di X c'é il numero di squadre del campionato e che contiene i vincoli che cambiano in funzione del numero di squadre.

Utilizzo

Per far funzionare il programma e avere in output un campionato su di un file di testo chiamato **out.txt** utilizzando **12 thread** per il calcolo bisogna eseguire il seguente comando:

```
clingo ./campionato_vincoli.cl ./campionato[X].cl  -t 12 > out.txt
```

Rimpiazzando [X] con il numero di squadre di cui si vuole il calendario.

Output

L'output é una lista di predicati assegna con arietá 3 in cui il primo argomento é la giornata di riferimento della partita, il secondo é la squadra di casa e il terzo é la squadra in trasferta. Siccome l'output puó essere spesso confusionario é opportuno processarlo con un editor di testo e il risultato di un campionato da 14 squadre é il seguente:

```
assegna(1,bulls,celtics)
assegna(1,cavaliers,nets)
assegna(1,seventysixers,mavericks)
assegna(1,warriors,lakers)
assegna(1,clippers,rockets)
assegna(1,pistons,pacers)
assegna(1,heat,nuggets)
assegna(2,pacers,celtics)
assegna(2,bulls,seventysixers)
assegna(2,heat,cavaliers)
assegna(2,mavericks,pistons)
assegna(2,warriors,clippers)
assegna(2,lakers,rockets)
assegna(2,nets,nuggets)
...
assegna(26,celtics,bulls)
assegna(26,pistons,mavericks)
assegna(26,lakers,warriors)
assegna(26,cavaliers,clippers)
assegna(26,nuggets,rockets)
assegna(26,nets,pacers)
assegna(26,seventysixers,heat)
```

I 3 puntini indicano che ci sono le altre giornate nel file di output ma che non le ho riportate qui per non rendere la relazione troppo lunga.

Predicati

I predicati che sono stati utilizzati sono i seguenti:

- assegna/3:
    1. Giornata di campionato
    2. Squadra di casa
    3. Squadra in trasferta
- squadra/1:

    1. Una squadra del campionato

- giornata/1:
    1. Una giornata del campionato
- citta/2:
    1. Una squadra del campionato
    2. La sua cittá di afferenza

Performance

Una delle maggiori difficotá é stata quella di riuscire a far terminare il programma con un elevato numero di squadre pur rispettando tutti i vincoli. Questo é particolarmente difficile in questo paradigma poiché non si specifica come ottenere la soluzione ma come deve essere tale soluzione: in pieno rispetto del **paradigma dichiarativo**.

Tuttavia sono state fatte delle misurazioni di tempi impiegati dal programma in utilizzando diverse formulazioni per gli stessi vincoli.

Tentativi Un **primo tentativo** é stato quello di rimuovere l'aggregato che obbliga lo **svolgimento di tutte le partite in qualche giornata**:

```
1 { assegna(G, Squadra1, Squadra2): giornata(G)  } 1 :- partita(Squadra1,Squadra2).
```

Con il seguente predicato e relativo vincolo.

```
% Vincolo alternativo ad aggregato (non posso assegnare a giornate diverse la stessa partita
:- assegna(G1, Squadra1, Squadra2), assegna(G2, Squadra1, Squadra2), G1 <> G2.
```

Purtroppo tale cambiamento ha solamente rallentato l'esecuzione del programma.

Il **secondo tentativo** per velocizzare il programma cerca metodi alternativi per risolvere il problema di **non avere la stessa squadra che fa partite diverse nella stessa giornata**

Metodo 1:

```
:- assegna(G, Squadra1, Squadra2), assegna(G, Squadra1, Squadra3), Squadra2 != Squadra3.
:- assegna(G, Squadra2, Squadra1), assegna(G, Squadra3, Squadra1), Squadra2 != Squadra3.
:- assegna(G, Squadra2, Squadra1), assegna(G, Squadra1, Squadra3), Squadra2 != Squadra3.
```

Metodo 2:

```
gioca(G, A) :- assegna(G,A,_).
gioca(G, A) :- assegna(G,_,A).
:- squadra(A), giornata(G), not gioca(G,A).
```

Il metodo 2, sebbene piú leggibile, é notevolmente piú lento del metodo 1. Pertanto si é preferito il metodo 1.

Importante: il metodo 1 non é esaustivo per tutti i casi possibili di partite della stessa squadra nella stessa giornata, per esempio assegna(G,A,B) e assegna(G,B,A) non é impedito, tuttavia esiste un altro vincolo che obblga ad avere andata e ritorno rispettivamente nella prima metá di campionato e nella seconda metá.

Tempo Il tempo di esecuzione sulla mia macchina é il seguente DNF = Did Not Finish

Dimensione Campionato	Numero squadre che condividono lo stadio	numero di Thread	Tempo di Esecuzione
10 squadre	2	1	.01 s
12 squadre	2	1	11 s
12 squadre	4	1	58 s
14 squadre	2	1	2 min 33 s
14 squadre	2	12	6 s
16 squadre	2	12	1 min 12 sec
18 squadre	2	12	21 min 19 sec
20 squadre	2	12	DNF (> 14h)