# RELAZIONE PROGETTO **PROLOG**


Università degli studi di Torino

Corso di Laurea Magistrale in Informatica

Intelligenza Artificiale e Laboratorio 2015/2016

DOCENTE:

**Prof. Gian Luca Pozzato**

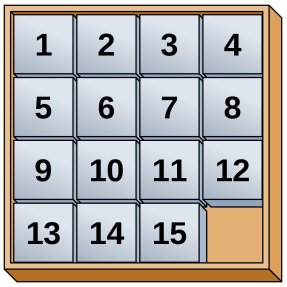
RELAZIONE A CURA DI:

823832 **Giovanni Bonetta**
762661 **Riccardo Renzulli**
763056 **Gabriele Sartor**

# I DOMINI

## IL GIOCO DELL'OTTO

Il gioco del quindici è un rompicapo che consiste nell'ordinare 15 caselle di una scacchiera di dimensione *4x4* nel modo mostrato in figura.




Le uniche operazioni possibili sono lo spostamento di caselle nello spazio vuoto, se esso è adiacente alla casella da spostare.
Questo problema può essere generalizzato su scacchiere di dimensioni *NxN* con *N* arbitrario, ordinando $n^2$*-1* caselle.
Nel nostro progetto prendiamo in considerazione la scacchiera *3x3* e proviamo ad ordinarla utilizzando le principali strategie di ricerca informate e non informate.
Per quanto riguarda le strategie informate A* e IDA*, l'euristica utilizzata per trovare la soluzione è la distanza di Manhattan. La funzione *H(n)* è la somma delle distanze di Manhattan di ogni numero dalla sua posizione finale.
In questo dominio il costo delle operazioni di spostamento delle caselle è unitario, di conseguenza la ricerca in ampiezza restituirà la soluzione ottima.
Inoltre, è stato tenuto in considerazione il fatto che il gioco dell'otto ha una configurazione della scacchiera impossibile da ordinare correttamente.
La configurazione impossibile è la seguente: *[1,2,3,4,5,6,empty,8,7]*.

```
d_search([1,2,3,4,5,6,empty,8,7],_):- write('Fail.\n'),abort.
```

Nel momento in cui una strategia incontra questa disposizione, essa si interrompe dichiarando l'occorrenza di un fallimento.

## IL LABIRINTO

Il dominio del labirinto prevede quattro azioni possibili con costo unitario per raggiungere una certa posizione goal a partire da un certo stato iniziale: *est*, *sud*, *ovest* e *nord*.

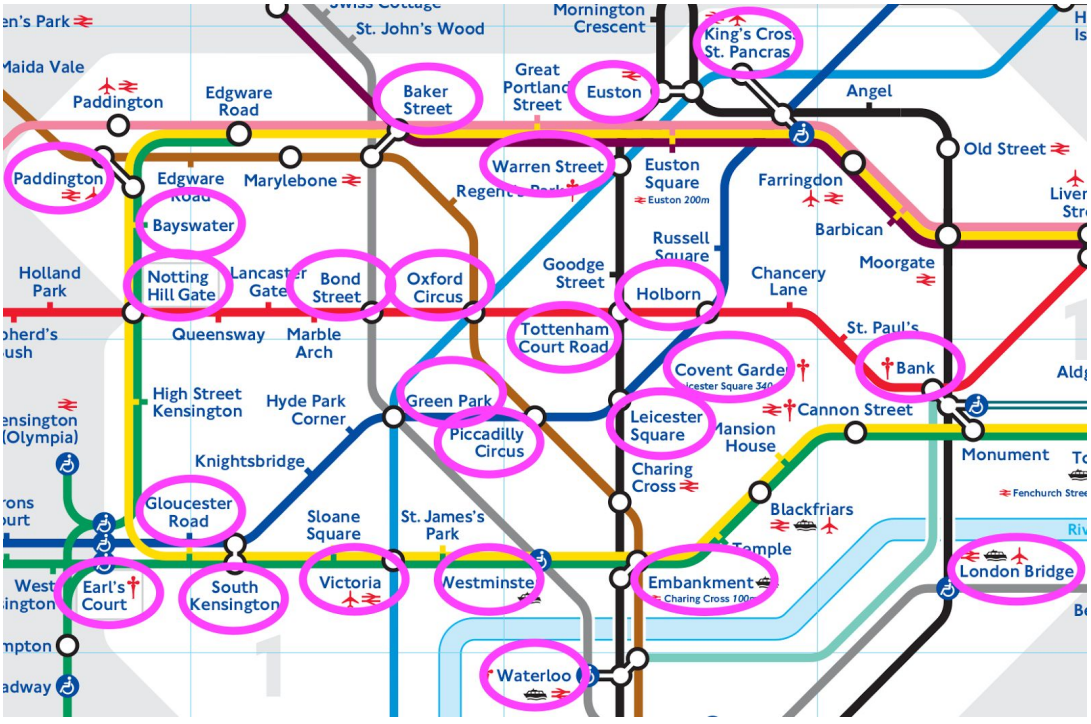
Per quanto riguarda le euristiche adottate per l'algoritmo *A**, abbiamo scelto come funzione *H(n)* la distanza di Manhattan tra il nodo *n* ed il nodo goal.

Come si vedrà meglio in seguito, le varie strategie sono state provate su mappe di varie dimensioni, ad esempio *10x10* e *20x20*.

## LA METROPOLITANA

Questo dominio è una versione ridotta della vera metropolitana di Londra, ossia ha meno stazioni e collegamenti di quella reale.

La struttura è simile a quella illustrata nella figura sottostante.



Data una stazione di partenza vogliamo trovare il percorso per arrivare ad una stazione di arrivo.
Le azioni possibili sono le seguenti: *geton*, *getoff* e *go*.
A differenza dei precedenti domini, il costo delle azioni non è unitario e questo ha un impatto sulla funzione *G(n)* sia delle strategie informate sia della strategia con ricerca in ampiezza. In questo caso è stata usata l'euristica di Manhattan sia per *G(n)* sia per *H(n).* Nelle strategie che non trovano la soluzione ottima non è stato necessario tener conto del costo non unitario delle azioni.

# SPERIMENTAZIONE

Per ogni coppia dominio-strategia abbiamo effettuato numerosi test e riportiamo i risultati nei prossimi paragrafi.
I test sono stati eseguiti su un pc MacBook Pro Intel Core i5 8GB Ram.
In ogni cella della tabella è contenuto il valore della tripla <numero di nodi chiusi; tempo impiegato per trovare la soluzione in millisecondi; numero di azioni della soluzione trovata> (<n, t, l>). Se in una cella invece non è contenuto nessun valore, significa che la computazione è troppo onerosa.
Le strategie implementate sono le seguenti: *breadth-first search*, *depth search*, *max-depth search*, *iterative deepening search*, *A** e *IDA**.
Per quanto riguarda il dominio del labirinto, le mappe utilizzate possono essere visualizzate in Appendice.

## IL GIOCO DELL'OTTO

Di seguito vengono riportati i risultati dei test su diverse configurazioni iniziali della scacchiera.

| gioco otto | A* | breadth-first | depth | IDA* | iterative deepening | max-depth |
|---|---|---|---|---|---|---|
| **history1-fail** | FAIL | FAIL | FAIL | FAIL | FAIL | FAIL |
| **history2-ok** | n = 2323<br>t = 3204<br>l = 25 | n = 327263<br>t = 17490775<br>l = 25 | n = 25086<br>t = 31387<br>l = 24461 | n = 5986<br>t = 174<br>l = 25 | n = 5914879<br>t = 106131<br>l = 25 | n = 935450<br>t = 12235<br>l = 25 |
| **history3-ok** | n = 2431<br>t = 3332<br>l = 28 | – | n = 37156<br>t = 84267<br>l = 36134 | n = 14761<br>t = 421<br>l = 28 | n = 43125018<br>t = 885563<br>l = 28 | n = 13556622<br>t = 215799<br>l = 28 |
| **history4-ok-simple** | n = 1<br>t = 8<br>l = 1 | n = 2<br>t = 8<br>l = 1 | n = 29<br>t = 10<br>l = 29 | n = 2<br>t = 9<br>l = 1 | n = 2<br>t = 22<br>l = 1 | n = 2<br>t = 0<br>l = 1 |
| **history5-ok** | n = 26<br>t = 10<br>l = 11 | n = 1157<br>t = 216<br>l = 11 | n = 73307<br>t = 464285<br>l = 70251 | n = 100<br>t = 11<br>l = 11 | n = 3680<br>t = 47<br>l = 11 | n = 1399<br>t = 15<br>l = 11 |
| **history6-fail** | FAIL | FAIL | FAIL | FAIL | FAIL | FAIL |

La *history1-fail* contiene uno stato iniziale impossibile da risolvere, quindi restituisce un fallimento immediatamente.

La configurazione di partenza di *history6-fail* è apparentemente risolvibile, ma dopo alcuni passi gli algoritmi raggiungono lo stato impossibile da risolvere e dichiarano fallimento.

Le *history2-ok* e *history3-ok* sono simili; infatti la strategia *iterative deepening search* restituisce, rispettivamente, una soluzione ottima di lunghezza 25 e 28. Come si evince dalla tabella, *A** e *IDA** trovano una soluzione ottima andando ad aprire molti meno nodi rispetto alle strategie *iterative deepening search* e *max-depth search*. Bisogna notare che gli stati possibili di questo dominio sono 9!, ovvero 362880, ma queste ultime due strategie considerano dei nodi già visitati in precedenza.

Nonostante la *breadth-first search* esamini meno nodi delle precedenti due strategie, non sono sufficienti diverse ore per trovare la soluzione. Questo è dato dal fatto che il tempo di esecuzione della ricerca in ampiezza in Prolog è fortemente influenzato

dalla sua complessità spaziale. Come sappiamo *iterative deepening search* ha complessità spaziale lineare nella lunghezza del cammino preso in considerazione, mentre quella della *breadth-first search* è esponenziale. A parte le configurazioni della scacchiera già visitate, questa strategia tiene in memoria gli stati successori ancora da visitare con i rispettivi cammini per raggiungere quello stato.
Nelle *history4-oksimple* e *history5-ok* abbiamo due configurazioni iniziali meno complesse da risolvere, la prima necessita di una mossa e la seconda di 11. Si noti che la *depth search* trova soluzioni di lunghezza maggiore perché, dopo aver espanso un nodo, va subito ad esaminarlo e cosi via, anche se la soluzione è molto vicina.

## Il labirinto

Anche in questo dominio la prima mappa *map1-fail-10x10* è una situazione di fallimento (l'interprete Prolog restituisce *false*) in quanto dallo stato iniziale è impossibile raggiungere quello finale per via dei muri circostanti. Possiamo notare come per la mappa *map2-simple-10x10* la strategia migliore è proprio quella *A** dato che trova la soluzione ottima in poco tempo. Come ci aspettavamo, la ricerca in ampiezza trova la soluzione ottima in tempi maggiori rispetto a quella in profondità che comunque ne trova una non ottimale di lunghezza 48. Inoltre la ricerca *IDA** è migliore sia rispetto alla *iterative deepening search* sia alla *max-depth search* in termini di nodi visitati e tempi.
In *map3-20x20* vediamo come la maggior parte delle strategie "esplodano" in spazio e tempo per via della complessità della mappa e solamente *A**, *depth search* e *IDA** trovano una soluzione in tempi ragionevoli; anche in questo caso *A** si dimostra essere la migliore.
Casi molto interessanti sono le ultime due mappe: *map4-middle-10x10* e *map5-guided-10x10*. In queste situazioni infatti le prestazioni della *depth search* sono ottime paragonate a quelle di *A**. In *map5-guided-10x10* la prima è migliore della seconda strategia sia in termini di nodi chiusi sia di tempo e questo è dovuto principalmente all'ordine in cui sono scritte le regole di applicabilità delle azioni: la *depth search* prova ad andare subito in direzione est e poi sud trovando così la

soluzione ottima in meno tempo rispetto ad *A** la quale considera anche la strada verso sud.

| labirinto | A* | breadth first | depth | IDA* | iterative deepening | max-depth |
|---|---|---|---|---|---|---|
| map1-fail-10x10 | FAIL | FAIL | FAIL | FAIL | FAIL | FAIL |
| map2-simple-10x10 | n = 50<br>t = 9<br>l = 20 | n = 1264<br>t =155<br>l = 20 | n = 1192;<br>t = 20<br>l = 48 | n = 13078<br>t = 91<br>l = 20 | n = 548376<br>t = 4945<br>l = 20 | n = 16661<br>t = 101<br>l = 20 |
| map3-20x20 | n = 72<br>t = 12<br>l = 28 | – | n = 38<br>t = 8<br>l = 38 | n = 15796<br>t = 117<br>l = 28 | – | – |
| map4-middle-10x10 | n = 53<br>t = 9<br>l = 23 | – | n = 55<br>t = 9<br>l = 55 | n = 33009<br>t = 212<br>l = 23 | n = 21913117<br>t = 19884898<br>l = 23 | n = 1304<br>t = 15<br>l = 23 |
| map5-guided-10x10 | n = 55<br>t = 9<br>l = 36 | n = 56<br>t = 11<br>l = 36 | n = 36<br>t = 8<br>l = 36 | n = 486<br>t = 12<br>l = 36 | n = 1767<br>t = 13<br>l = 36 | n = 36<br>t = 0<br>l = 36 |

## LA METROPOLITANA

Il dominio della metropolitana ha un numero di stati sicuramente minore di quelli precedenti. Questo è dovuto al minor numero di azioni applicabili (3 azioni possibili) in un dato nodo e al numero di possibili luoghi raggiungibili nella mappa (23 stazioni).
Per ogni test abbiamo ottenuto la soluzione in tempi abbastanza rapidi.
Da notare che non è stato necessario nemmeno il test con un goal fallimentare in quanto esso non esiste poiché nella metropolitana si può raggiungere ogni stazione da qualsiasi punto di partenza.
Per esempio in *history1* il goal è raggiungere la stazione *covent garden* partendo da *bayswater*. Questa history presenta il punto di partenza e arrivo in due linee diverse quindi obbligando l'algoritmo a scegliere in modo efficiente quando cambiare linea per minimizzare i costi.

*A** e *breadth-first search* tengono conto del differente costo delle azioni di spostamento (salire e scendere dalla metro non ne hanno). Questo permette loro di calcolare la soluzione ottima di lunghezza 12 con costo 11.9.
Da notare come tutte le altre strategie trovino una soluzione con minor numero di azioni proprio perché non sono informate e non garantiscono l'ottimalità sui costi.
Anche *IDA**, pur calcolando il valore delle varie azioni, è comunque limitata dalla sua componente iterativa che la blocca prima di raggiungere la soluzione con costo minore.
Per quanto riguarda la *history2-oksimple* abbiamo testato le strategie di ricerca su una soluzione banale che consta di una sola azione di spostamento. Ricordiamo che la soluzione finale è comunque formata da tre azioni : *geton*, *go* e *getoff* (3 azioni come riportato in tabella). Vediamo anche come *depth search* si comporti male andando ad allungare di molto il percorso. Questo è dovuto all'ordine di esecuzione delle azioni nell'algoritmo; purtroppo la strategia sceglie come prima opzione quella di percorrere la linea nella direzione opposta a quella che avrebbe portato immediatamente al raggiungimento del goal.
Infine in *history3-straight* abbiamo semplicemente testato le strategie su una configurazione in cui la stazione di partenza e di arrivo giacciono sulla stessa linea della metropolitana. Anche in questo caso gli algoritmi si sono comportati bene dando la stessa soluzione corretta con l'unica eccezione di *depth search* come ci si poteva aspettare.

| metropolitana | A* | breadth-first | depth | IDA* | iterative deepening | max-depth |
|---|---|---|---|---|---|---|
| **history1** | n = 33<br>t = 9<br>l = 12 | n = 66<br>t = 9<br>l = 12 | n = 358<br>t = 13<br>l = 37 | n = 75<br>t = 8<br>l = 9 | n = 512<br>t = 11<br>l = 9 | n = 126<br>t = 8<br>l = 9 |
| **history2-oksimple** | n = 3<br>t = 0<br>l = 3 | n = 8<br>t = 0<br>l = 3 | n = 18<br>t = 8<br>l = 16 | n = 13<br>t = 0<br>l = 3 | n = 8<br>t = 13<br>l = 3 | n = 8<br>t = 7<br>l = 3 |
| **history3-straight** | n = 15<br>t = 10<br>l = 7 | n = 48<br>t = 12<br>l = 7 | n = 4186<br>t = 61<br>l = 25 | n = 96<br>t = 8<br>l = 7 | n = 239<br>t = 9<br>l = 7 | n = 57<br>t = 9<br>l = 7 |

# APPENDICE

Queste sono le mappe che abbiamo utilizzato per i test sul dominio del labirinto.

**MAP1 10x10 FAIL**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | G |
| 2 | | | | | | | | | | |
| 3 | | | | | | | | | | |
| 4 | | S | | | | | | | | |
| 5 | | | | | | | | | | |
| 6 | | | | | | | | | | |
| 7 | | | | | | | | | | |
| 8 | | | | | | | | | | |
| 9 | | | | | | | | | | |
| 10 | | | | | | | | | | |

**MAP2 10x10 SIMPLE**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | |
| 2 | | | | | | | | | | |
| 3 | | | | | | | | | | |
| 4 | | S | | | | | | | | |
| 5 | | | | | | | | | | |
| 6 | | | | | | | | | | |
| 7 | | | | | | | | | G | |
| 8 | | | | | | | | | | |
| 9 | | | | | | | | | | |
| 10 | | | | | | | | | | |

**MAP4 10x10 MIDDLE**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | G |
| 2 | | | | | | | | | | |
| 3 | | | | | | | | | | |
| 4 | | S | | | | | | | | |
| 5 | | | | | | | | | | |
| 6 | | | | | | | | | | |
| 7 | | | | | | | | | | |
| 8 | | | | | | | | | | |
| 9 | | | | | | | | | | |
| 10 | | | | | | | | | | |

## MAP3 20x20

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | | | |
| 10 | | | | | | | | | | S | | | | | | | | | | |
| 11 | | | | | | | | | | | | | | | | | | | | |
| 12 | | | | | | | | | | | | | | | | | | | | |
| 13 | | | | | | | | | | | | | | | | | | | | |
| 14 | | | | | | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | | | | | | | | | |
| 17 | | | | | | | | | | | | | | | | | | | | |
| 18 | | | | | | | | | | | | | | | | | | | | |
| 18 | | | | | | | | | | | | | | | | | | | | |
| 20 | | | | | | | | | | | | | | | | | | | | G |

**MAP5 10x10 GUIDED**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | S | | | | | | | | | |
| 2 | | | | | | | | | | |
| 3 | | | | | | | | | | |
| 4 | | | | | | | | | | |
| 5 | | | | | | | | | | |
| 6 | | | | | | | | | | |
| 7 | | | | | | | | | | |
| 8 | | | | | | | | | | |
| 9 | | | | | | | | | | |
| 10 | | | | | | | | | | G |