

# Relazione su “Gioco dell'Otto”

## Search problem: 8-puzzle

### ➤ Introduzione ai “Search Problem”:

Si considera un problema di ricerca come un' “indagine” su un numero elevato di possibili configurazioni. Questo avviene a partire da uno stato iniziale detto  $x_0$  fino a che non viene trovato un *match* tra una configurazione possibile e quella cercata, detta “goal”.

Spesso esplorare ciecamente tutto lo spazio degli stati può essere troppo costoso i.e. [Blind Search], così si concepiscono euristiche, legate allo specifico problema che si vuole risolvere, le quali “guidano” l'algoritmo verso la soluzione. Nella nostra implementazione è usata **l'euristica Manhattan** in quanto sufficientemente veloce per il Gioco dell'Otto.

Gli algoritmi che hanno permesso la soluzione sono **A\*** e **IDA\***.

Vediamo più nella pratica in cosa consiste il problema.

### ➤ Presentazione del Gioco:

Il gioco dell' Otto è una versione più semplificata del famoso gioco con 16 tessere. In questo caso ci sono solo 9 tessere posizionate su una scacchiera 3x3. Il gioco consiste nel partire da una configurazione iniziale casuale

7	8	1
4		5
2	6	3

Figura1 - Configurazione iniziale casuale

e via via spostare le tessere in tutte le possibili direzioni, tranne che in diagonale, con lo scopo di giungere alla configurazione *goal*:

1	2	3
4	5	6
7	8	

Figura2 - Configurazione finale o goal

E' da notare che con 9 tessere si ottiene un numero di configurazioni iniziali pari a  $9! = 362\ 880$

➤ **Risolvibilità:**

Una configurazione iniziale si dice *risolvibile* se l'algoritmo ha la possibilità di arrivare alla soluzione.

Non tutte le configurazioni sono risolvibili.

La procedura che abbiamo utilizzato per distinguere le configurazioni risolvibili da quelle non risolvibili è la seguente:

*“Contare il numero di volte che una tessera è maggiore di una alla sua destra (questo per ogni casella e per ogni configurazione iniziale).*

*Se il numero che risulta è pari, allora quella determinata configurazione sarà risolvibile, altrimenti no. Questo perché la configurazione goal da noi cercata è pari.”* [Vedi Figura 2]

In questo modo si riesce a distinguere tra casi risolvibili e non risolvibili. Ora applicando questa regola ad un numero di configurazioni iniziali pari a 54 000 si ottiene la seguente divisione:

- ✓ Risolvibili: 27 750 **(51.3 %)**
- ✓ Non Risolvibili: 26 250 **(48.7%)**

### ➤ Implementazione e Risvolti Tecnici :

La funzione euristica da noi utilizzata e implementata in entrambi gli algoritmi (A\* e IDA\*) è stata **l'euristica Manhattan**. Un'euristica alternativa potrebbe essere quella delle **tessere fuoriposto**.

L'euristica Manhattan è definita, nel nostro caso, come *la somma delle distanze, in termini di coordinate, delle caselle fuori posto dalla loro posizione goal*. Per esempio la distanza Manhattan tra due punti si calcola in questo modo:

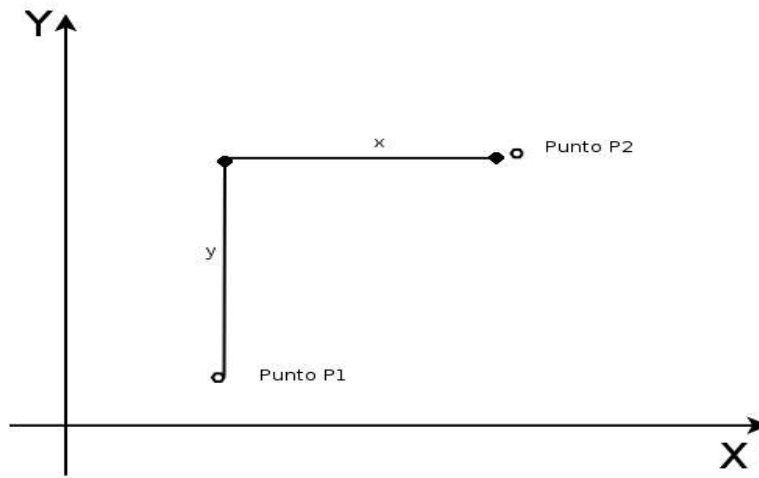


Figura 3 - Distanza tra due punti P1 P2 calcolata con la Manhattan

$$\text{Distanza} := x + y$$

Sia la Manhattan che l'euristica delle tessere fuoriposto sono entrambe ammissibili, ossia sono sottostime della distanza reale dal goal:  $h'(n) \leq h(n) \forall n$  nodi dove  $h'(n)$  è l'euristica e  $h(n)$  la distanza vera dal goal. La **“Manhattan”** risulta molto più **“informata”** in quanto fornisce al calcolatore un'indicazione negativa:  $h'(n)$  cresce nel momento in cui azzarda una mossa sbagliata, che lo allontana dalla configurazione goal. Un'indicazione che l'altra euristica non forniva.

Come sappiamo entrambi gli algoritmi da noi utilizzati sono algoritmi ottimi, cioè ordinano i nodi tramite un min-heap dove la priorità non è data SOLO dalla **minor profondità**, ma da un **fattore di qualità**. Come facciamo a calcolare la qualità di un nodo?

Si definisce **funzione di valutazione** la seguente:  $f(n) = g(n) + w \cdot h'(n)$  dove:

- 1)  $g(n)$  è la profondità del nodo  $n$  ossia un valore esatto.
- 2)  $h'(n)$  la nostra euristica o stima della distanza da goal del nodo  $n$ .

Quindi in definitiva la funzione  $f(n)$  rappresenta una **stima del costo del percorso vincolato a passare dal nodo considerato**: in breve restituisce la bontà del nodo.

Nel nostro caso abbiamo utilizzato proprio questa funzione di valutazione con  $h'(n)$  = distanza Manhattan e facendo variare il coefficiente  $w$ .

### ➤ Presentazione dei due algoritmi:

Per risolvere il problema del gioco dell'otto abbiamo quindi utilizzato due algoritmi: A\* e IDA\*

**L'algoritmo A\* sfrutta come base l'algoritmo A**, infatti ordina i nodi non più in termini di distanza dal nodo goal, ma in termini di qualità. Esso calcola la funzione di valutazione dei nodi che incontra ed espande solo quello con  $f(n)$  minima. Il confronto lo effettua anche con i nodi passati contemplando anche eventuale *backtracking*.

Con A\* intendiamo tutti gli algoritmi A nei quali  $h'(n)$  è una funzione euristica ammissibile ovvero  $h'(n) \leq h(n) \forall n$ . Questo implica, per la definizione di ammissibilità, che A\* termina con un percorso ottimale se esistono percorsi dalla radice al nodo goal.

**Le varie implementazioni di A\* sono quindi un sottoinsieme di implementazioni di A in grado di trovare sempre un percorso ottimo.** Come vedremo anche dai risultati derivati dall'implementazione di questo algoritmo, A\* è un algoritmo molto veloce, ma per mantenere questa caratteristica occupa molta memoria.

Per cercare di mantenerne le caratteristiche ma **migliorare la gestione della memoria**, abbiamo utilizzato l'algoritmo **IDA\*** che implementa l'Iterative Deepening su A\*: ad ogni iterazione l'algoritmo ricerca in profondità con un limite dato dal valore della funzione di valutazione  $f(n)$ , quindi il livello di ricerca dell'algoritmo varia da nodo a nodo. Il limite viene aumentato ad ogni iterazione finché non viene trovata la soluzione. Anche IDA\* è un algoritmo ammissibile e perciò può essere confrontato con A\*. Grazie all'approfondimento iterativo l'algoritmo risparmia memoria, ovviamente a discapito della velocità.

### ➤ Analisi delle prestazioni dei due algoritmi:

Passiamo adesso all'analisi del parametro  $w$  nei due algoritmi, ovvero il peso della stima della distanza dal goal rispetto alla profondità del nodo.

Poiché nella funzione di valutazione  $f(n) = g(n) + w \cdot h'(n)$ ,  $w$  moltiplica il valore della funzione euristica, **grandi valori di questo parametro riescono ad enfatizzare il potere euristico dell'algoritmo implementato**, mentre valori piccoli riportano l'algoritmo a lavorare ciecamente. Quindi per  $w=0$  l'algoritmo A\* riprende le caratteristiche di Breadth First e IDA\* quelle dell'Iterative Deepening.

Come prima casistica consideriamo il caso in cui  $w=1$  in modo da non alterare l'implementazione dei due algoritmi. Successivamente andremo ad analizzare il comportamento dell'algoritmo A\* aumentando il valore di questo parametro cercando così di aumentare la spinta euristica verso la risoluzione del problema.

Facendo un'analisi parallela dei due algoritmi possiamo confrontarli con dati relativi all'esecuzione pratica e capire empiricamente i loro pregi ed i loro difetti.

## ➤ Risultati

La **profondità della soluzione** è un parametro di elevata importanza in quanto ci dice a che altezza si trova il nodo goal. Per ambedue gli algoritmi i risultati sono identici: la massima profondità della soluzione trovata è 30 (18 volte) mentre la profondità minima è 6 (4 volte). La profondità media è 22,003.

Profondità Soluzione	N° di volte
6	4
8	24
9	56
10	84
12	216
14	498
15	586
16	1395
18	2634
20	3521
22	6456
24	6556
26	4578
28	1128
30	18

<b>Media</b>	22,003
<b>Massimo</b>	30
<b>Minimo</b>	6

Tabella 1 - Profondità della soluzione

I dati relativi alla **distanza stimata dall'euristica** iniziale sono anche questi identici per i due algoritmi e va da un massimo di 22 (54 volte) a un minimo di 4 (42 volte). La media è quindi 13,87.

Euristica	Numero di Volte
4	14
6	62
8	329
10	1041
12	2138
14	2765
16	2201
18	1082
20	268
22	18

<b>Media</b>	13,870
<b>Massimo</b>	22
<b>Minimo</b>	4

Tabella 2 - Euristica

Altri risultati molto importanti riguardano il **numero di nodi generati** per arrivare alla soluzione: questo dato è molto importante in quanto può essere interpretato come indice di qualità e di confronto per i due algoritmi .

<b>Nodi Generati A*</b>		<b>Nodi Generati IDA*</b>	
<b>Media</b>	2 503	<b>Media</b>	2 977
<b>Massimo</b>	43 421	<b>Massimo</b>	91 244
<b>Minimo</b>	15	<b>Minimo</b>	25

Tabella 3 - Confronto tra i due algoritmi

Da notare come il numero di nodi generati da A\* sia più basso di quello di IDA\*. Questo rafforza quanto detto in precedenza nell'analisi teorica dei due algoritmi in quanto mostra la maggiore velocità di esecuzione di A\*.

➤ **Calcolo delle misure delle Prestazioni**

Per la misura delle prestazioni utilizziamo due parametri: la **penetranza** ed il **fattore di ramificazione** che sono formalmente definiti come segue:

Dati

$L$  : Lunghezza del cammino risolutivo

$T$  : Numero totale di nodi generati,

si definisce come Penetranza  $P : \frac{L}{T} \quad 0 < P \leq 1$

e Fattore di ramificazione<sup>1</sup>  $B$  t.c.  $B + B^2 + B^3 + \dots + B^L = T \Leftrightarrow \frac{B}{B-1}(B^L - 1) = T$

<b>L per A*</b>	
<b>Media</b>	22,003
<b>Massimo</b>	30
<b>Minimo</b>	6

<b>L per IDA*</b>	
<b>Media</b>	22,003
<b>Massimo</b>	30
<b>Minimo</b>	6

<b>T per A*</b>	
<b>Media</b>	2 503
<b>Massimo</b>	43 421
<b>Minimo</b>	15

<b>T per IDA*</b>	
<b>Media</b>	2 977
<b>Massimo</b>	91 244
<b>Minimo</b>	25

<b>P (L/T) per A*</b>	
<b>Media (L<sub>medio</sub>/T<sub>medio</sub>)</b>	0,0090
<b>Massimo (L<sub>min</sub>/T<sub>min</sub>)</b>	0,4000
<b>Minimo (L<sub>max</sub>/T<sub>max</sub>)</b>	0,0010

<b>P (L/T) per IDA*</b>	
<b>Media (L<sub>medio</sub>/T<sub>medio</sub>)</b>	0,0074
<b>Massimo (L<sub>min</sub>/T<sub>min</sub>)</b>	0,2400
<b>Minimo (L<sub>max</sub>/T<sub>max</sub>)</b>	0,0003

Tabella 4 – Misure delle prestazioni nei due algoritmi

Il fattore di ramificazione  $B$  per A\* risulta 1,041 mentre per IDA\* risulta 1,353.

Possiamo utilizzare questi valori ottenuti per fare nuovamente un confronto tra i due algoritmi: il fatto che la penetranza di A\* sia maggiore rispetto a quella di IDA\* e il suo fattore di ramificazione invece sia minore, sono entrambi indici della sua maggiore velocità nel calcolo della soluzione del problema. Purtroppo nessuno di questi parametri fornisce informazioni rispetto alla memoria impiegata dai due algoritmi per raggiungere la soluzione

<sup>1</sup> La prima è la definizione formale di fattore di ramificazione, mentre la seconda è la formula utilizzata nella pratica per il calcolo.

➤ **Analisi del peso  $w$**

Analizziamo adesso il comportamento dell'algoritmo A\* al variare del peso  $w$  della funzione di valutazione .

La prima parte di questo studio riguarda il calcolo della profondità media della soluzione al variare di  $w$ . Si può notare sia dalla tabella che dal grafico che fino ad un valore di  $w$  pari a 1,130 ci si mantiene nelle condizioni di ammissibilità ed è mantenuta quindi l'ottimalità dell'algoritmo. Il cammino medio fino al valore critico di  $w$  rimane infatti lo stesso per poi iniziare a crescere proporzionalmente alla crescita di  $w$ .

W	Profondità soluzione
1,000	22,003
1,005	22,003
1,010	22,003
1,050	22,003
1,100	22,003
1,115	22,003
1,116	22,003
1,117	22,003
1,118	22,003
1,119	22,003
1,120	22,003
1,121	22,003
1,122	22,003
1,123	22,003
1,124	22,003
1,125	22,003
1,127	22,003
1,128	22,003
1,129	22,003
1,130	22,003
1,135	22,004
1,150	22,010
1,200	22,036
1,250	22,087
1,300	22,157
1,350	22,247
1,400	22,313
1,450	22,399
1,500	22,530
2,000	23,802

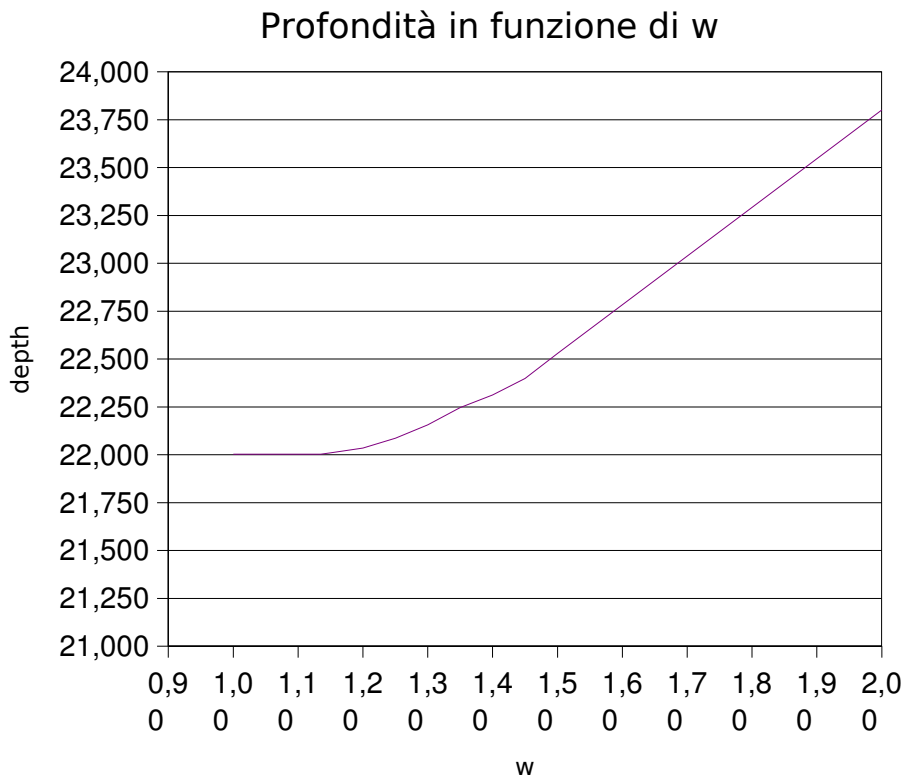


Tabella 5 – Profondità della soluzione

Adesso è necessario cercare di capire quanto, al variare di  $w$ , è lunga la soluzione media dell'algorithm. Con questo dato possiamo capire quanto ci è possibile e se ci è possibile aumentare  $w$  dal suo valore di default (1) e accelerare l'esecuzione dell'algorithm senza perdere l'ammissibilità.

Anche in questo caso riportiamo la tabella con i dati misurati al variare di  $w$  ed il grafico che ne illustra l'andamento. Possiamo vedere che la lunghezza della soluzione al variare di  $w$  è una funzione tendenzialmente decrescente, il che vuol dire che l'aumentare di questo parametro enfatizza il potere euristico dell'algorithm incrementandone la velocità. Dobbiamo però anche considerare il precedente risultato ottenuto, che ci assicurava l'ammissibilità per  $w \leq 1.130$ . Per valori maggiori di questo valore critico si ottengono soluzioni più veloci ma non più ammissibili. Si può vedere come in un intorno del valore critico di  $w$  la funzione risulta instabile con dei picchi anomali.

W	Lunghezza della soluzione
1,000	2.503
1,005	2.493
1,010	2.475
1,050	2.314
1,100	1.994
1,115	1.837
1,116	1.853
1,117	1.899
1,118	1.781
1,119	1.840
1,120	1.827
1,121	1.792
1,122	1.771
1,123	1.674
1,124	1.774
1,125	1.781
1,127	1.804
1,128	1.852
1,129	1.834
1,130	1.827
1,135	1.821
1,150	1.778
1,200	1.664
1,250	1.387
1,300	1.432
1,350	1.408
1,400	1.388
1,450	1.242
1,500	1.066
2,000	867

