Distanza Manhattan vs Database di Pattern

Michela Crulli

July 7, 2018

1 Introduzione

Il seguente elaborato tratta il confronto di A* con l' euristica basata su database di patterns e quella basata sulla distanza Manhattan.
In particolare l'intento è di voler confrontare i risultati relativi ai nodi espansi, penetranza ed effective branching factor.

2 Implementazione

L'implementazione è stata fatta con il linguaggio di programmazione Python utilizzando come guida $http://aima.cs.berkeley.edu/python/readme.html$.
Il programma è composto da due file, **puzzle.py** dove si sviluppa l'intero codice risolutivo e **test.py** in cui attraverso chiamate a puzzle.py vengono fatti dei test su di essa.
L'idea principale su cui si basa l'intera struttura è quella di generare un 15-puzzle con una determinata difficoltà e di andare a risolverlo attraverso l'implementazione di A* con l' euristica basata su database di patterns e quella basata sulla distanza Manhattan.

- **test.py:** in questo file vengono richiamate le funzioni definite in puzzle.py. Per prima cosa si determina una certa difficoltà, poi in ordine: si chiama la funzione "train" che genererà i pattern, si chiama la funzione "generate" passandogli il goal e la difficoltà scelta e successivamente attraverso la chiamata a "solve", per entrambe le euristiche, verranno restituite le soluzioni in termini di nodi espansi, penetranza e branching factor.
- **puzzle.py:** in questo file invece troviamo l'implementazione vera e propria di ciò che vogliamo ottenere.

Gli aspetti principali di questo file sono: l'implementazione di A* con le due euristiche e la generazione dei PatternDatabase.
In particolare vengono creati i Pattern Database in base al numero che si vuole.
Successivamente si genera il puzzle che dovrà essere risolto e infine lo si risolve con le due implementazioni di A*.
Nei passi necessari per arrivare al goal, si tiene traccia dei nodi espansi.
Mentre per il calcolo della penetranza e dell'effective branching factor si devono applicare due formule:

- penetranza: $\frac{d}{N}$ dove d = profondità della soluzione, N = numero di nodi espansi
- effective branching factor: $\frac{B}{B-1}$ (B^d-1).
  In particolare B viene determinato attraverso una funzione che calcola il metodo di newton ("optimize.newton").

```
def effective_bf(total, frontier):

    fun = lambda b:(((b/(b-1))*(b**(frontier)-1)))- total

    br = optimize.newton(fun, 1.5)
    return br
```

dove total = N = numero di nodi espansi.

Nell'output il programma fa vedere il puzzle generato e i risultati in ordine di: mosse per arrivare alla soluzione (da leggere da destra verso sinistra) con relativa difficoltà e profondità in cui si trova la soluzione, nodi visitati, ripetizioni, nodi espansi, penetranza e effective branching factor.

3 Risultati

I risultati di seguito riportati sono stati approssimati per eccesso alla seconda cifra decimale.

	Distanza Manhattan				Database di Pattern		
d	nodi espansi	penetranza	b*	d	nodi espansi	penetranza	b*
3	11	0.3	1.81	3	11	0.3	1.81
5	19	0.27	1.48	5	19	0.27	1.48
8	25	0.33	1.25	8	25	0.33	1.25
11	72	0.15	1.29	11	33	0.34	1.17
14	230	0.06	1.34	14	68	0.20	1.19
17	607	0.02	1.34	17	106	0.16	1.18
20	1224	0.02	1.34	20	156	0.12	1.17
23	2207	0.01	1.33	23	257	0.09	1.18
26	3453	0.01	1.31	26	428	0.06	1.17
29	14810	0.00	1.32	29	4294	0.01	1.23
32	20946	0.00	1.30	32	9136	0.00	1.25
35	65178	0.00	1.31	35	18648	0.00	1.26
38	139889	0.00	1.31	38	35688	0.00	1.26

4 Osservazioni

Nei risultati sopra riportati possiamo notare il grande vantaggio dato dall'utilizzo dei Pattern Database, i quali consentono di raggiungere il goal con un numero molto inferiore di nodi espansi rispetto a quelli necessari per A* con la distanza Manhattan.
Infatti, come riportato in: *"Richard E.Korf, Ariel Fernel" - Disjoint pattern database heuristics"*, i database di patterns contengono il numero minimo di mosse necesssarie per risolvere un gruppo di celle, per tutte le possibili posizioni della casella vuota, permettendo,una volta creati, di raggiungere velocemente il goal.
Il numero di nodi espansi, come previsto, aumenta all'aumentare della profondità in cui si trova la soluzione. Poichè questa crescita è esponenziale, la penetranza, essendo data dal rapporto tra la profondità (d) e il numero di nodi espandi, tenderà a zero all'aumentare di quest'ultimo dato.
Per quanto riguarda l'effective branching factor (b*) notiamo che all'aumentare della profondità (d) e dunque della difficoltà, esso rimane circa costante con un valore non troppo lontano da 1, permettendo di risolvere problemi abbastanza grandi in tempi computazionali ragionevoli.

5 Conclusione

Al termine dell'analisi effettutata possiamo concludere che A* risolto con la distanza Manhattan, se pur in maniera peggiore rispetto ai Database di Patter, fino ad una certa difficoltà permette comunque di risolvere il 15-puzzle espandendo un numero ragionevole di nodi.
Il problema riscontrato nei database di pattern è stato nella loro creazione per

la risoluzione di puzzle difficili in quanto hanno richiesto più tempo computazionale.
Per quanto riguarda il tempo computazionale per la risoluzione del puzzle, questo, non è stato inserito nel test poichè risulta essere proporzionale al numero di nodi espansi e quindi non fornisce informazioni maggiori di quelle già ottenute testando gli altri valori.