Euristiche per A*

Bernardo Tiezzi

02 Settembre 2018

1 Introduzione

La seguente relazione risulta necessaria al fine di fornire una descrizione adeguata riguardo il progetto realizzato per conseguire l'esame per il corso di *Intelligenza Artificiale* del corso di laurea triennale in Ingegneria Informatica, nell'anno 2017/2018. Il progetto svolto richedeva di sviluppare quattro diverse tipologie di euristiche, necessarie per l'implementazione del'algoritmo A*, riproducendo i risultati presentati nella **Tabella 1** dell'articolo di ***Korf & Felner* (2002)**. Dopo una breve descrizione teorica del tipo di problema presentato, viene descritto il lavoro svolto ed i risultati ottenuti.

2 Alla luce della Teoria

L'algoritmo **A*** utilizzato nell'elaborato rappresenta la forma più diffusa di ricerca *best first*, che utilizza una strategia di **ricerca informata**. La ricerca informata è una tecnica che sfrutta una conoscenza specifica del problema al di là della semplice definizione, per trovare soluzioni in modo più efficiente rispetto ad una strategia di ricerca non informata. La scelta del nodo da espandere durante la ricerca varia in base al valore di *f(n)*, nota come **funzione di valutazione**. Inoltre, vengono utilizzate, per il caso di A*, altre due funzioni, *g(n)* e *h(n)*. Quindi per la ricerca A*, la *f(n)* è definita nel seguente modo: $f(n) = g(n) + h(n)$, dove $g(n)$ rappresenta il costo del cammino dal nodo iniziale al nodo n e $h(n)$ rappresenta il costo stimato del cammino minimo da n all'obiettivo. Dunque $f(n)$ risulta essere pari al costo stimato della soluzione più conveniente passante per n. La funzione $h(n)$ è nota come **funzione euristica** ed in generale rappresenta *il costo stimato del cammino più conveniente dallo stato del nodo n allo stato obiettivo*. Affinché la ricerca A* risulti sia completa che ottima è necessario tuttavia che la funzione euristica $h(n)$ soddisfi alcune condizioni. La prima condizione da soddisfare riguarda l'**ammissibilità**. Un'euristica si dice **ammissibile** se *non commette mai un errore per eccesso* riguardo la stima del costo per arrivare all'obiettivo. Dato che $g(n)$ rappresenta il costo effettivo per raggiungere n nel cammino corrente e $f(n) = g(n) + h(n)$, ne consegue che $f(n)$ non deve mai sbagliare per eccesso il costo reale della soluzione lungo il cammino corrente che passa per n. Una

seconda condizione, leggermente più forte, denominata **consistenza** è richiesta solo per alcune applicazioni di A*. Un'euristica $h(n)$ si definisce **consistente** se per ogni nodo n ed ogni suo successore n' generato da un'azione a, il costo stimato per raggiungere l'obiettivo partendo da n non è superiore al costo di passo per arrivare ad n' sommato al costo stimato per andare da n' all'obiettivo: $h(n) \leq c(n, a, n') + h(n')$. Tale formula è nota come **disuguaglianza triangolare**. E' condizione necessaria che ogni euristica **consistente** risulti anche **ammissibile**; il contrario non è necessariamente vero, tuttavia risulta difficile verificare che un'euristica ammissibile non sia consistente.

3 Analisi dell'elaborato

Il progetto richiedeva, come già precedentemente sottolineato, di realizzare 4 diverse euristiche per l'algoritmo di ricerca informata A*, applicato su uno Sliding-tiles Puzzle di dimensione $4 * 4$ al fine di poter trovare la soluzione ottima, posizionando le caselle in ordine crescente con la posizione vuota in alto a sinistra, a partire da un qualsiasi stato casuale del puzzle. In seguito dovevano essere svolti test adeguati e riportati i risultati sperimentali opportunamente commentati. Le funzioni euristiche sviluppate sono le seguenti:

1. *Distanza di Manhattan*: La distanza di Manhattan è una tecnica che consiste nel calcolare il valore dell'euristica per lo stato corrente misurando la distanza effettiva di ogni casella, esclusa la posizione vuota, dalla sua posizione obiettivo, sommando tra loro i risultati. Tale funzione euristica risulta essere sia ammissibile che consistente.

2. *Linear Conflict*: Il principio alla base dell'euristica Linear Conflict si definisce come segue: due caselle t_j e t_k sono in "conflitto lineare" se t_j e t_k sono sulla stessa linea, le posizioni obiettivo di t_j e t_k sono entrambe su quella linea, t_j è a destra di t_k e la posizione obiettivo di t_j è alla sinistra della posizione obiettivo di t_k. Il controllo riguardo i possibili conflitti tra le caselle del puzzle viene effettuato sia sulle righe che sulle colonne. Gli eventuali conflitti presenti vengono aggiunti al valore della distanza di Manhattan relativa allo stato corrente, incrementandone il valore.

3. *Disjoint Pattern Database*: L'euristica generata utilizzando la seguente tecnica risulta essere l'argomento principale attorno a cui ruota l'intero progetto. Per poter realizzare euristiche ammissibili è possibile anche suddividere il problema iniziale in più sottoproblemi semplificandone l'analisi. Per realizzare un Disjoint Pattern Database è necessario suddividere il puzzle in gruppi di caselle disgiunti, così che ogni casella faccia parte di uno ed un solo gruppo. Per ogni gruppo viene inizialmente calcolato il numero minimo di mosse necessarie a posizionare le caselle, appartenenti a quell'unico gruppo, nella loro posizione obiettivo. E' **fondamentale** ricordarsi di tenere memoria solo delle mosse applicate alle caselle del

gruppo considerato al momento, trascurando i possibili spostamenti delle altre. Per ogni raggruppamento viene così creato un database contenente una qualunque configurazione delle caselle selezionate, assieme al valore associato ad essa. La distanza di Manhattan può essere vista come un caso particolare di Disjoint Pattern Database, dove esiste un gruppo per ogni casella. Ciò che caratterizza questa particolare tecnica è il fatto che non tiene conto della posizione della casella vuota così come il fatto che, tenendo conto delle sole mosse relative ai componenti di un gruppo alla volta, per poter ricavare il valore dell'euristica del problema originario è sufficiente sommare i valori relativi alle configurazioni considerate di ogni gruppo.

4. *Disjoint Pattern Database reflected*: Una variante del Disjoint Pattern Database originale che prevede di creare un ulteriore database dato dalla riflessione del pattern iniziale. In questo modo si realizzano due Disjoint Database che produrranno, per ogni configurazione dello stato del puzzle, due euristiche differenti, delle quali sarà scelta quella a valore maggiore.

Risulta interessante notare come ogni tipologia di euristica presentata in precedenza risulti sia ammissibile che consistente. Il progetto è stato realizzato in codice *Python 2.7* utilizzando come ambiente di sviluppo il programma *PyCharm* sviluppato dalla compagnia **JetBrains**. Come specificato nel file README relativo al progetto, il codice si presenta suddiviso in 4 python file: *main.py*, *permutation.py*, *puzzle.py* e *test.py*. Inoltre sono stati creati 6 file pickle, uno per ogni disjoint group, che contengono i database di 2 differenti disjoint pattern.

4 Presentazione ed Analisi dei risultati

Di seguito sono presentati i risultati ottenuti eseguendo 100 test random per ogni tipologia di euristica sviluppata. Per un'analisi del codice implementato ed un'opportuna spiegazione su come riprodurre i risultati sperimentali si rimanda alla lettura del file README, relativo al progetto.

Funzione Euristica	*Valore*	*Nodi Espansi*	*Tempo medio*	*Nodi/Sec*
Distanza di Manhattan	2308	4138907	42,396	97624,914
Linear Conflict	2317	1077738	10,852	99315,855
Disjoint Pattern	2392	575758	2,345	245476,070
Disjoint Pattern reflected	2694	281271	1,162	247245,330

La tabella riporta, per ogni funzione euristica, la somma dei valori iniziali di ogni ricerca effettuata, il numero totale di nodi espansi durante i cammini, il tempo medio impiegato per ogni ricerca calcolato in secondi, e il numero di nodi espansi al secondo. I test sono stati effettuati utilizzando come macchina un *PC Dell Vostro 5568* con processore Intel(R)Core(TM) i5-7200U CPU @

2.50GHz 2.70GHz e RAM installata da 8 GB. Confrontando i valori riportati in tabella notiamo come i risultati ottenuti concordino con i risultati sperimentali presenti nella tabella in Korf & Felner(2002). I valori delle euristiche sono superiori per le due tipologie di Disjoint Pattern Database rispetto ai valori ottenuti impiegando la distanza di Manhattan o Linear Conflict. Al contrario sia il tempo medio di esecuzione, sia il numero totale di nodi espansi lungo tutti i cammini dallo stato iniziale allo stato obiettivo, risultano nettamente inferiori per entrambe le euristiche Disjoint Pattern database. La quarta colonna si discosta dal valore riportato nella *Tabella 1* perchè avendo eseguito un numero di test inferiori, il numero di nodi espansi varia notevolmente. Non è stato possibile realizzare un numero di esperimenti superiori a causa dei limiti strutturali della macchina utilizzata. I risultati ottenuti durante le sperimentazioni hanno comunque confermato le ipotesi formulate dalla teoria sulla ricerca informata, sottolineando ulteriormente il grande vantaggio offerto dalle funzioni euristiche rispetto ai metodi della ricerca classica. Inoltre utilizzando euristiche che stimano il costo effettivo della ricerca con maggior precisione, è possibile ridurre fortemente i costi in termini sia di spazio che di tempo.