UNIVERSITÀ DEGLI STUDI
DI NAPOLI FEDERICO II

Corso di Laurea in Ingegneria Informatica

Corso di Elementi di Intelligenza Artificiale
Prof. **Giancarlo Sperlì**
a.a. 2024-2025

Progetto

Pathfinding: A* vs Greedy Best-First

Autori

Alessio Romano N46007394 alessio.romano02dev@gmail.com

Mattia Gifuni N46007229 mat.gifuni@studenti.unina.it

July 5, 2025

Contents

1 Introduzione **3**

2 Fondamenti teorici **4**
2.1 Modello di problema 4
2.2 Euristiche 4

3 Algoritmi di ricerca informata **5**
3.1 A* Search 5
3.2 Greedy Best-First Search 5

4 Implementazione **6**
4.1 Metriche 6

5 Setup sperimentale **7**

6 Risultati **8**

7 Analisi dei risultati **9**

8 Conclusioni **10**

1 Introduzione

Il *path finding* (ricerca del percorso) è la disciplina che studia algoritmi e tecniche per trovare un cammino ottimale o "sufficientemente buono" tra due punti all'interno di un grafo. Applicazioni tipiche vanno dalla robotica, ai videogiochi, fino ai sistemi informativi geografici.

Uno dei primi esempi storici di path-finding in un videogioco è rappresentato da *Pac-Man* (Namco, 1980). In questo classico arcade, i quattro "fantasmi" non si muovono in modo del tutto casuale, ma calcolano ad ogni incrocio quale mossa (alto, basso, sinistra, destra) li avvicini di più a Pac-Man. Questo comportamento, basato implicitamente su una stima della distanza di Manhattan nel reticolo del labirinto, è una forma embrionale di ricerca informata: garantisce un inseguimento più realistico e competitivo rispetto a un semplice movimento randomico, anticipando di fatto le tecniche di A* e Greedy Best-First che vedremo successivamente.

2 Fondamenti teorici

2.1 Modello di problema

Il dominio del problema di pathfinding è modellato come un grafo non orientato $G = (V, E)$ definito su una griglia bidimensionale di dimensione $H \times W$:

- V è l'insieme dei vertici, ciascuno corrispondente a una cella libera o a un ostacolo. Indicando con (x, y) la cella alla riga y e colonna x, allora

$$V = \{\, (x, y) \mid 0 \leq x < W,\ 0 \leq y < H \}.$$

- $E \subseteq V \times V$ è l'insieme di archi non orientati che connettono coppie di celle adiacenti orizzontalmente o verticalmente:

$$E = \{\, \{(x, y), (x', y')\} \mid |x - x'| + |y - y'| = 1 \}.$$

- Ogni cella (x, y) può essere *libera* (codificata come 0) o *ostacolo* (codificata come 1).
- Il *costo* di attraversamento di ogni arco è unitario ($c(u, v) = 1$ per ogni $\{u, v\} \in E$).
- Dato un nodo di partenza $s \in V$ e un nodo d'arrivo $g \in V$, lo scopo è trovare un cammino semplice $P = (v_0 = s, v_1, \ldots, v_k = g)$ che minimizzi la somma dei costi

$$\text{Costo}(P) \ = \ \sum_{i=1}^{k} c(v_{i-1}, v_i) \ = \ k.$$

Questa rappresentazione a grafo consente di applicare algoritmi di ricerca standard (BFS, A*, Greedy BF) sfruttando strutture dati quali liste di adiacenza o matrici, e definendo in modo naturale le operazioni di espansione del nodo corrente verso i suoi "vicini" in quattro direzioni (alto, basso, sinistra, destra).

2.2 Euristiche

Un'euristica è una funzione che stima il costo residuo dal nodo corrente n al goal g. Le due euristiche più comuni per griglie ortogonali sono:

- **Distanza di Manhattan**

$$h_M(n) = |x_n - x_g| + |y_n - y_g|.$$

- **Distanza Euclidea**

$$h_E(n) = \sqrt{(x_n - x_g)^2 + (y_n - y_g)^2}.$$

La scelta dell'euristica influenza direttamente:

- Il numero di nodi espansi: euristiche più accurate riducono la dimensione dell'open set.
- Il costo computazionale per espansione: funzioni Euclidea e normali operazioni aritmetiche coinvolgono calcoli in virgola mobile, a differenza di Manhattan che usa solo somme e valori assoluti.
- La qualità e l'ottimalità del percorso: euristiche inadmissibili possono velocizzare, ma a rischio di percorsi non ottimi.

3 Algoritmi di ricerca informata

3.1 A* Search

A* è un algoritmo di ricerca informata che sfrutta una funzione di valutazione

$$f(n) = g(n) + h(n)$$

dove:

- $g(n)$ è il costo del cammino più breve noto dal nodo di partenza s fino al nodo n.
- $h(n)$ è l'euristica che stima il costo residuo da n al goal g.

L'algoritmo mantiene due insiemi di nodi:

- **Open set:** contiene i nodi scoperti ma non ancora espansi, ordinati in base al valore $f(n)$ minimo.
- **Closed set:** contiene i nodi già espansi, per evitare riespansioni.

L'algoritmo A* gode delle seguenti proprietà

- **Ottimalità:** A* restituisce sempre il percorso di costo minimo se $h(n)$ è *ammissibile* e *consistente*.
- **Complessità temporale:** nell'ordine di $O(b^d)$ nel peggiore dei casi, ma spesso molto inferiore grazie all'euristica, dove b è il branching factor e d la profondità del goal.
- **Complessità spaziale:** memorizza tutti i nodi esplorati in open e closed set, $O(b^d)$ nel peggiore dei casi.

3.2 Greedy Best-First Search

Greedy Best-First Search utilizza esclusivamente l'euristica $h(n)$ per guidare la selezione del prossimo nodo da espandere:

$$n \in \text{Open set che minimizza } h(n).$$

Non tiene conto del costo già sostenuto $g(n)$: ciò rende la ricerca più "avida" ed estremamente veloce nell'avvicinarsi al goal, ma senza garanzia di ottimalità. L'algoritmo Greedy Best-First gode delle seguenti proprietà

- **Ottimalità:** non garantita, perché ignora $g(n)$ e può seguire strade troppo "avide" che poi richiedono deviazioni.
- **Complessità temporale:** generalmente molto inferiore ad A*, poiché espande molti meno nodi inizialmente.
- **Complessità spaziale:** proporzionale ai nodi attivi in open set, solitamente minore di A*.

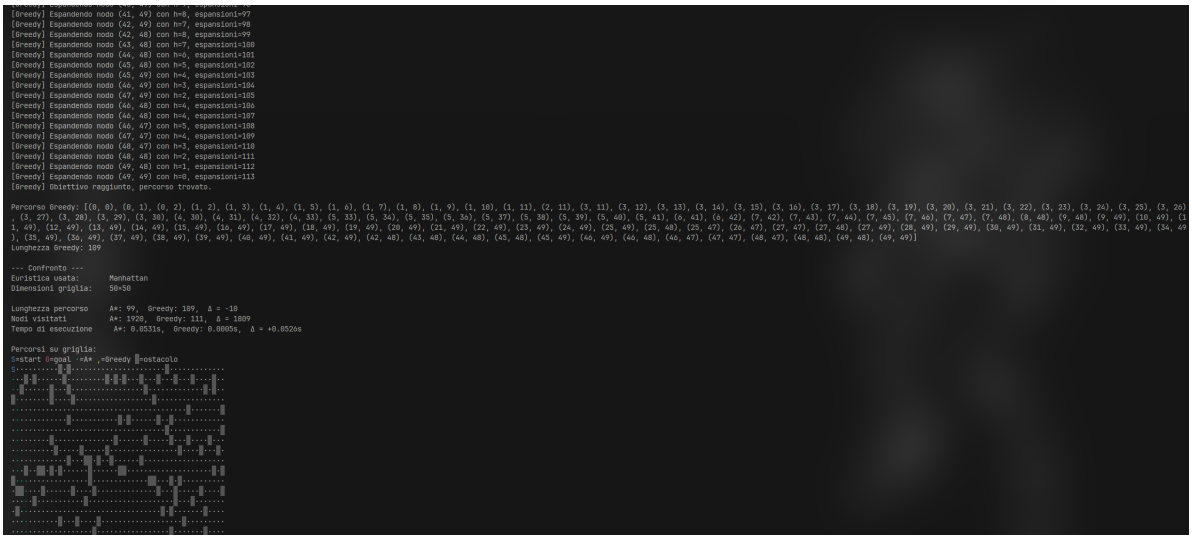
4 Implementazione

Il progetto è stato realizzato interamente in **Python 3.7+**, senza dipendenze esterne: vengono utilizzati soltanto moduli della *Standard Library*. L'applicazione è organizzata in cinque moduli principali, tutti rilasciati nella cartella di progetto:

- `utils.py`: generazione griglia e visualizzazione console.
- `heuristics.py`: definizione delle funzioni Manhattan ed Euclidea.
- `astar.py`: implementazione di A*.
- `greedy_best_first.py`: implementazione di Greedy BF.
- `main.py`: interfaccia utente per scelta dimensioni e euristica, esecuzione e confronto.

Per ciascuna configurazione:

- Abbiamo eseguito **5 repliche** con lo stesso seed per garantire riproducibilità.
- Misurato il tempo medio di esecuzione.
- Calcolato la media dei valori su tutte le repliche.



4.1 Metriche

1. **Lunghezza del percorso** trovato (numero di passi).
2. **Numero di nodi espansi** (dimensione del closed set).
3. **Tempo di esecuzione** medio (secondi).

5 Setup sperimentale

Per valutare l'impatto della dimensione della griglia e della scelta dell'euristica, abbiamo eseguito i seguenti esperimenti:

- **Griglia 50×50**, densità di ostacoli $p = 0.1$:
  - A* con euristica Manhattan
  - A* con euristica Euclidea
  - Greedy Best-First con euristica Manhattan
  - Greedy Best-First con euristica Euclidea
- **Griglia 500×500**, stessa densità $p = 0.1$:
  - A* con euristica Manhattan
  - A* con euristica Euclidea
  - Greedy Best-First con euristica Manhattan
  - Greedy Best-First con euristica Euclidea
- **Griglia 50×50**, densità di ostacoli $p = 0.3$:
  - A* con euristica Manhattan
  - A* con euristica Euclidea
  - Greedy Best-First con euristica Manhattan
  - Greedy Best-First con euristica Euclidea
- **Griglia 500×500**, stessa densità $p = 0.3$:
  - A* con euristica Manhattan
  - A* con euristica Euclidea
  - Greedy Best-First con euristica Manhattan
  - Greedy Best-First con euristica Euclidea

6 Risultati

Di seguito sono riportati i risultati ottenuti per le quattro configurazioni sperimentali: griglie 50×50 e 500×500, con densità di ostacoli $p = 0.1$ e $p = 0.3$.

Table 1: Risultati su griglia 50×50 con densità $p = 0.1$

Algoritmo / Euristica	Lunghezza	Nodi espansi	Tempo (s)
A* – Manhattan	99	2 084	0.0346
A* – Euclidea	99	2 226	0.0398
Greedy BF – Manhattan	113	115	0.0004
Greedy BF – Euclidea	99	98	0.0004

Table 2: Risultati su griglia 500×500 con densità $p = 0.1$

Algoritmo / Euristica	Lunghezza	Nodi espansi	Tempo (s)
A* – Manhattan	999	191 299	4.2289
A* – Euclidea	999	223 116	6.4353
Greedy BF – Manhattan	1 115	1 141	0.0084
Greedy BF – Euclidea	1 009	1 019	0.0265

Table 3: Risultati su griglia 50×50 con densità $p = 0.3$

Algoritmo / Euristica	Lunghezza	Nodi espansi	Tempo (s)
A* – Manhattan	99	297	0.0011
A* – Euclidea	99	1 308	0.0069
Greedy BF – Manhattan	145	330	0.0017
Greedy BF – Euclidea	111	120	0.0005

Table 4: Risultati su griglia 500×500 con densità $p = 0.3$

Algoritmo / Euristica	Lunghezza	Nodi espansi	Tempo (s)
A* – Manhattan	1 001	45 200	1.0494
A* – Euclidea	1 001	148 471	4.2039
Greedy BF – Manhattan	1 389	1 827	0.0306
Greedy BF – Euclidea	1 213	1 463	0.0330

7 Analisi dei risultati

Dall'osservazione dei dati sperimentali emergono considerazioni significative in merito al comportamento degli algoritmi A* e Greedy Best-First Search (GBF), rispetto a fattori quali la densità degli ostacoli, la dimensione della griglia e il tipo di euristica adottata. Di seguito si riportano le principali evidenze:

- **Influenza della densità degli ostacoli**:
  - A basse densità ($p = 0.1$), A* espande un numero molto elevato di nodi (oltre 2000 su 50×50, fino a 223 000 su 500×500), ma trova sempre il percorso ottimale.
  - Greedy BF, pur espandendo pochissimi nodi (meno di 120 su 50×50), produce percorsi più lunghi, anche se con prestazioni accettabili soprattutto usando l'euristica Euclidea.
  - A densità più alte ($p = 0.3$), A* mantiene percorsi ottimali ma con una riduzione dei nodi espansi su griglie piccole, mentre su griglie grandi l'espansione resta elevata (oltre 148 000 nodi con Euclidea).
  - GBF, in presenza di più ostacoli, risente maggiormente della qualità dell'euristica: Manhattan produce percorsi molto più lunghi, mentre Euclidea migliora la qualità del percorso mantenendo i tempi bassi.
- **Effetto della dimensione della griglia**:
  - Su griglie piccole (50×50), entrambi gli algoritmi completano l'esecuzione in tempi trascurabili (inferiori a 0.04 secondi per A*, e nell'ordine dei millisecondi per GBF).
  - Su griglie grandi (500×500), A* mostra tempi di esecuzione significativamente più elevati (fino a 6.4 secondi), mentre GBF resta sempre sotto i 35 millisecondi, dimostrando una forte scalabilità in termini di tempo.
- **Confronto tra le euristiche**:
  - L'euristica Euclidea risulta più efficace della Manhattan soprattutto per GBF, consentendo la generazione di percorsi più corti con un modesto incremento nei nodi espansi.
  - Per A*, l'euristica Euclidea può portare a una leggera riduzione della qualità computazionale (più nodi espansi, tempi maggiori), senza però migliorare la qualità del percorso che è già ottimale.
- **Sintesi e raccomandazioni**:
  - A* è preferibile in contesti dove è fondamentale garantire la qualità del percorso, accettando costi computazionali elevati (es. robotica, pianificazione autonoma).
  - GBF si rivela vantaggioso in contesti dove conta la velocità, a scapito dell'ottimalità (es. videogiochi, navigazione approssimata, scenari real-time).
  - L'adozione dell'euristica Euclidea in GBF è raccomandata, in quanto fornisce un buon compromesso tra qualità del percorso e rapidità di esecuzione.

8 Conclusioni

Dall'analisi dei risultati emerge una distinzione netta tra i due algoritmi. A* si conferma il più affidabile dal punto di vista dell'optimalità, trovando sistematicamente il percorso più corto grazie all'uso combinato della funzione costo $g(n)$ e dell'euristica $h(n)$. Tuttavia, questo vantaggio comporta un maggiore dispendio computazionale in termini di nodi espansi e tempo di esecuzione.

Greedy Best-First Search privilegia la rapidità, affidandosi esclusivamente all'euristica. Questo lo rende sensibilmente più veloce, soprattutto su griglie di grandi dimensioni o ad alta densità, ma a scapito della qualità del percorso, che tende a essere più lungo.

In sintesi, A* è indicato nei contesti in cui l'accuratezza del percorso è prioritaria, mentre Greedy BF è preferibile quando l'efficienza è il principale vincolo.