

---

ANNO ACCADEMICO 2024/2025

---

# Sistemi Intelligenti

---

## Teoria

Altair's Notes



**UNIVERSITÀ  
DI TORINO**



---

DIPARTIMENTO DI INFORMATICA

---



<b>CAPITOLO 1</b>	<b>INTRODUZIONE</b>	<b>PAGINA 5</b>
1.1	Che Cos'è l'Intelligenza Artificiale? Un Inizio — 6 • Test di Turing — 6 • Strong e Weak AI — 8 • Esempi — 9 • Definire AI — 10	5
1.2	Risoluzione Automatica di Problemi I Problemi — 12 • Metodi di Ricerca non Informati (Blind Search) — 13 • Lista di Strategie — 15	12
1.3	Ricerca Informatata A* — 18 • Approfondimento su Euristiche — 21	18
1.4	Strategie di Ricerca con Avversario	22
<b>CAPITOLO 2</b>	<b>TEST2</b>	<b>PAGINA 24</b>



# Premessa

## Licenza

Questi appunti sono rilasciati sotto licenza Creative Commons Attribuzione 4.0 Internazionale (per maggiori informazioni consultare il link: <https://creativecommons.org/licenses/by/4.0/>).



## Formato utilizzato

Box di "Concetto sbagliato":

### Concetto sbagliato 0.1: Testo del concetto sbagliato

Testo contenente il concetto giusto.

Box di "Corollario":

### Corollario 0.0.1 Nome del corollario

Testo del corollario. Per corollario si intende una definizione minore, legata a un'altra definizione.

Box di "Definizione":

### Definizione 0.0.1: Nome delle definizioni

Testo della definizione.

Box di "Domanda":

### Domanda 0.1

Testo della domanda. Le domande sono spesso utilizzate per far riflettere sulle definizioni o sui concetti.

Box di "Esempio":

### Esempio 0.0.1 (Nome dell'esempio)

Testo dell'esempio. Gli esempi sono tratti dalle slides del corso.

**Box di "Note":**

**Note:-**

Testo della nota. Le note sono spesso utilizzate per chiarire concetti o per dare informazioni aggiuntive.

**Box di "Osservazioni":**

**Osservazioni 0.0.1**

Testo delle osservazioni. Le osservazioni sono spesso utilizzate per chiarire concetti o per dare informazioni aggiuntive. A differenza delle note le osservazioni sono più specifiche.



# 1

## Introduzione

### Note:-

DISCLAIMER: questi appunti sono stati scritti da una persona che ha dovuto dare questo esame in magistrale (yeah, l'ho skippato in triennale e ora mi tocca darlo): essendo che ho dato molti esami che dipendono da questo insegnamento è possibile che in alcune sezioni dia per scontato delle cose.

### 1.1 Che Cos'è l'Intelligenza Artificiale?

Nell'immaginario l'intelligenza artificiale viene solitamente assimilata a quella di un robot antropomorfo che risolve problemi complessi e impara da essi.

Risolve problemi complessi



Impara

Robot antropomorfo



Risolve problemi complessi (?)

Robot antropomorfo



Impara

Però esistono altri tipi di IA:

- Servizi di streaming: portali per l'accesso a molti files. Utilizzano meccanismi di personalizzazione.
- Social network.
- Assistenti virtuali.
- Macchine fotografiche/Smartphone.

### 1.1.1 Un Inizio

#### Definizione 1.1.1: Intelligenza

Complesso di facoltà psichiche e mentali che consentono all'uomo di pensare, comprendere o spiegare i fatti o le azioni, elaborare modelli astratti della realtà, intendere e farsi intendere dagli altri, giudicare, e lo rendono insieme capace di adattarsi a situazioni nuove e di modificare la situazione stessa quando questa presenta ostacoli all'adattamento; propria dell'uomo, in cui si sviluppa gradualmente a partire dall'infanzia e in cui è accompagnata dalla consapevolezza e dall'autoconsapevolezza, è riconosciuta anche, entro certi limiti (memoria associativa, capacità di reagire a stimoli interni ed esterni, di comunicare in modo anche complesso, ecc.), agli animali.

#### Note:-

Artificiale indica che non è naturale.

#### Prospettiva storica:

- 1936: Alan Turing formalizza la Turing Machine.
- 1940: ENIAC: primo computer "moderno".
- 1950: Test di Turing, quando un computer può dirsi intelligente?
- Il dubbio nasce dal contesto bellico in cui vennero sviluppati i primi computer: all'epoca solo poche persone istruite riuscivano a fare i calcoli necessari.
- 1956: Nasce l'intelligenza artificiale.

#### Breve storia dell'automazione:

- *Automazione del calcolo*: metà anni '50, pochi dati, molti calcoli.
- *Automazione di procedure amministrative e contabili*: metà anni '60, pochi calcoli, grandi molti di dati alfanumerici.
- *Automazione di fabbrica*: metà anni '70, primi robot industriali, ambiente predeterminato.
- *Automazione di ufficio*: metà anni '80, primi PC, primi strumenti per utenti non esperti.
- *Automazione della ricerca delle informazioni*: fine anni '90, internet, WEB, motori di ricerca.

#### Domanda 1.1

L'automazione è intelligenza?

**Ragionando:** la calcolatrice è automatica, ma non si può dire intelligente. Una lavatrice è automatica, ha diversi programmi e si adatta. Un rover che gira su Marte effettua esperimenti e si adatta, ha una certa autonomia. Infine, gli LLM eseguono un programma e hanno la capacità di comunicare mediante linguaggio naturale.

### 1.1.2 Test di Turing

#### Domanda 1.2

Quando un programma può dirsi intelligente?

#### Definizione 1.1.2: Turing Test (The Imitation Game)

Un'intervistatore deve capire se un'entità misteriosa è umana o è una macchina. Può fare tutte le domande che vuole su qualsiasi argomento e l'entità deve rispondere (il tutto per scritto). Al termine l'intervistatore enuncia il suo verdetto. Se dice uomo ed era macchina, la macchina ha superato il test.

**AI:**

- Data e luogo di nascita:
  - Dartmouth Conference (USA), 1956.
  - Nome scelto da John McCarthy.
- In precedenza:
  - Una macchina può pensare ed essere considerata intelligente?
  - Vari approcci: cybernetica, teoria degli automi, etc.
  - Turing test.
- Successivamente:
  - Scacchi.
  - Giochi.
  - Dimostrazioni automatiche.

**Domanda 1.3**

Basta produrre gli output attesi per dire che vi è comprensione?

- Si può dare una risposta "giusta" avendo certe conoscenze e ragionamento.
- Ma si può dare una risposta "giusta" anche tirando a caso.

**Definizione 1.1.3: Esperimento della Stanza Cinese**

Una persona interagisce con un computer, programmato per rispondere con certi ideogrammi cinesi ad altri ideogrammi cinesi ricevuti in input. Il computer parla cinese? Lo capisce?

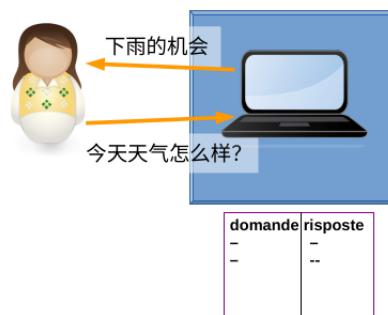


Figure 1.1: Esperimento di Searle.

**Note:-**

Ma supponiamo che una persona chiusa in una stanza ha istruzioni per rispondere con certi ideogrammi cinesi in risposta ad altri ideogrammi cinesi. Parla cinese? Lo capisce?

**Definizione 1.1.4: Test di Turing Inverso**

Usati per intercettare bot che cercano di accedere a form o a dati (C.A.P.T.C.H.A.).

**Note:-**

Una variante di questo test è usata in "Ma gli androidi sognano pecore elettriche?" (Blade Runner).



Figure 1.2: Esperimento di Searle.

### 1.1.3 Strong e Weak AI

Due tipi di intelligenza:

- **Strong AI:** è possibile riprodurre l'intelligenza umana?
- **Weak AI:** è possibile trovare dei modi per risolvere problemi che, se risolti dagli esseri umani richiederebbero intelligenza?

Obiettivo della weak AI:

- Programmare un agente artificiale in grado di:
  - Rilevare ostacoli.
  - Rilevare oggetti in movimento.
  - Costruire un piano d'azione.
  - Rilevare segnali significativi.
- In un ambiente che è:
  - Complesso.
  - Parzialmente prevedibile.
  - Parzialmente collaborativo.

**Note:-**

Nasce il binomio Agente-Ambiente.

**Definizione 1.1.5: Agente**

Un agente è un'astrazione che rappresenta un qualsiasi sistema che percepisce il proprio ambiente tramite i sensori e agisce su di esso tramite degli attuatori.

**Osservazioni 1.1.1 Caratteristiche dell'ambiente**

- Completamente osservabile: in ogni istante i sensori danno accesso a tutti gli aspetti dell'ambiente rilevanti per la scelta dell'azione.
- Parzialmente osservabile: i sensori danno accesso solo a parte dell'informazione rilevante (cause: sensori imprecisi oppure non in grado di rilevare alcuni dati).
- Deterministico: lo stato successivo è determinato dallo stato corrente e dall'azione applicata.
- Stocastico: applicando più volte una stessa azione in uno stesso stato si possono raggiungere stati diversi. Si dice strategico quando è stocastico solo per quanto riguarda le azioni degli altri agenti.
- Epistodico: l'esperienza degli agenti è divisa in episodi atomici: un episodio è dato da una percezione seguita da una singola azione (esempio: classificazione).

- Sequentiale: attività composta da più passi ognuno dei quali in generale influenzera i successivi.
- Statico: l'ambiente non cambia mentre l'agente "pensa".
- Dinamico: l'ambiente cambia mentre l'agente "pensa".
- Discreto: possono essere discreti stato, tempo, percezioni, azioni (esempio: gli scacchi hanno stati, percezioni, azioni discreti).
- Continuo: possono essere continui stato, tempo, percezioni, azioni (esempio: gli scacchi hanno tempo continuo).
- Singolo agente: viene modellata come agente una sola entità.
- Multi agente: vengono modellate come agenti più entità

### Paradigmi di programmazione:

- Approccio tradizionale:
  - Imperativo.
  - A oggetti.
  - *Non è AI*: risolve un singolo compito ed è strutturato come una sequenza di passi.
  - Descrivono il COME.
- Approccio dichiarativo:
  - Separa una descrizione del COSA da un programma generale.
  - Lo stesso programma è applicato a diverse descrizioni per risolvere problemi diversi.

#### 1.1.4 Esempi

##### Definizione 1.1.6: Mondo dei Blocchi

Tipico Toy Problem in ambito AI. Si hanno un tavolo con  $n$  posizioni e  $m$  blocchi. L'obiettivo è passare da uno stato iniziale a uno stato finale. Un agente può spostare un blocco per volta seguendo determinate regole.

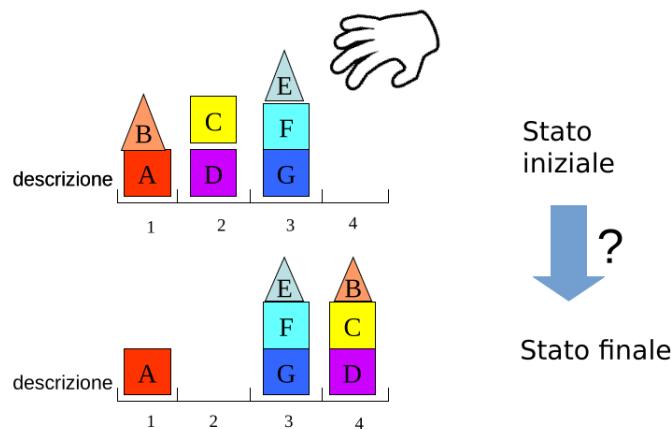


Figure 1.3: Mondo dei blocchi.

**L'agente:**

- Percepisce la situazione iniziale.
- Costruisce i passi per andare dalla situazione iniziale alla situazione finale.
- Deve determinare quali azioni lo avvicinano al *goal*.

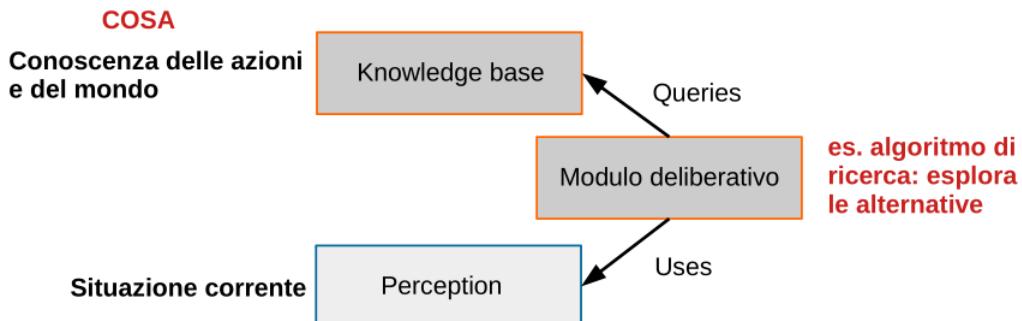


Figure 1.4: Meccanismo di deliberazione.

**Domanda 1.4**

Come scegliere le mosse?

- Dipende dal tipo di problema.
- A volte basta la prima mossa, a volte si vuole trovare una soluzione ottima.

### 1.1.5 Definire AI

**Definizione 1.1.7: Automazione**

Si deve programmare la macchina per fare ogni passo: è applicabile in domini fortemente ripetitivi.

**Definizione 1.1.8: Autonomia**

Un agente artificiale riceve compiti ad alto livello, l'utente demanda all'agente la risoluzione.

**Un agente autonomo:**

- Riceve solo compiti ad alto livello.
- Ragiona ed esplora alternative (molte mosse possibili a ogni istante).
- Riconosce quando non si può andare avanti su una strada <sup>1</sup>.
- Riconosce che si è già stati in quella situazione.
- Prima si ragiona e poi si agisce.

**Note:-**

In AI gli agenti autonomi sono un modo di concepire i programmi, in cui controllo e logica (o modello) sono chiaramente separati.

Un agente fa sempre ciò che è programmato a fare<sup>a</sup>.

<sup>1</sup>nota aggiuntiva: non proprio, dipende dal tipo di agente e dal suo control loop.

"No Terminator, sorry :"(

### Domanda 1.5

Cosa vuol dire fare la cosa giusta?

#### Definizione 1.1.9: Funzione Deliberativa

La funzione deliberativa di un agente determina le azioni che saranno eseguite. In termini informali un agente è razionale quando “fa la cosa giusta”, cioè opera per conseguire il “successo”.

#### Note:-

Occorre quindi definire una *misura di prestazione*.

**Il comportamento razionale di un agente dipende da 4 fattori:**

1. Azioni nelle facoltà dell'agente.
2. Misura di prestazione.
3. Conoscenza dell'ambiente.
4. Percezione.

#### Corollario 1.1.1 Agente Razionale

Un agente razionale dovrebbe scegliere sempre un'azione che massimizza la misura di prestazione attesa, data la particolare sequenza percettiva in oggetto e le informazioni derivabili dalla conoscenza dell'ambiente.

**Definizioni di AI:**

- Sistemi che pensano come esseri umani:
  - Haugeland, 1985.
  - Bellman, 1978.
- Sistemi che agiscono come esseri umani:
  - Kuzweil, 1990.
  - Rich e Knight, 1991.
- Sistemi che pensano razionalmente:
  - Charniak e McDermott, 1985.
  - Winston, 1992.
- Sistemi che agiscono razionalmente:
  - Poole et al., 1998.
  - Nilsson, 1998.

### Domanda 1.6

Quali problemi per l'AI:

- Non è adatta per:
  - Modelli matematici precisi.

- Metodi algoritmici specifici.
- È utile/necessaria per:
  - Problemi non deterministici.
  - Più soluzioni.
  - Dati non numerici.
  - Grandi Knowledge Base (KB).
  - Interazione con ambiente ed esseri umani.

## 1.2 Risoluzione Automatica di Problemi

In questa parte si affronta la problematica di come definire il concetto di problema e di soluzione, di distinguere tra soluzione e soluzione ottima. Sono studiati tre approcci alla risoluzione di problemi: ricerca nello spazio degli stati, ricerca in spazi con avversario (giochi ad informazione completa), risoluzione di problemi mediante soddisfacimento di vincoli.

### 1.2.1 I Problemi

- La realtà che definisce un problema può essere astratta in un insieme di stati.
- La realtà transisce da uno stato ad un altro tramite l'esecuzione di azioni (o operazioni).

**Caratteristiche:**

- *Stati discreti* (o dentro o fuori, non ci sono stati graduali).
- Effetto *deterministico* delle azioni.
- *Dominio statico* (non cambia durante l'esecuzione delle azioni).

**Esempio non deterministico:**

- Eseguendo più volte la stessa azione si possono avere conseguenze diverse.
- Si hanno *stati continui*.

#### Definizione 1.2.1: Obiettivo

Un obiettivo (goal) è un risultato verso il quale gli sforzi sono diretti. È una condizione data in termini di:

- Situazione.
- Prestazione.

#### Note:-

L'insieme degli stati obiettivo sono tutti gli stati in cui vale la condizione che li definisce.

#### Definizione 1.2.2: Algoritmo di Ricerca

L'algoritmo di ricerca determina una soluzione che, a partire da uno stato iniziale, permette di raggiungere un dato stato obiettivo. Usa:

- Una descrizione del problema.
- Un metodo di ricerca attraverso lo spazio degli stati.

**Corollario 1.2.1 Soluzione**

Una soluzione è un percorso nello spazio degli stati.

**Un problema di ricerca può essere definito come una tupla di 4 elementi:**

1. Stato iniziale: cattura la situazione a partire dalla quale viene computata la soluzione.
2. Funzione successore: dato uno stato e un'azione legale in esso calcola lo stato a cui si transisce eseguendo quell'azione in quello stato.
3. Test obiettivo: determina se lo stato a cui è applicato è lo stato goal: può verificare una proprietà o verificare l'appartenenza dello stato all'insieme degli stati target.
4. Funzione di costo del cammino: dato un percorso possibile gli assegna un costo numerico.

**Alcune astrazioni:**

- *Stati*: occorre rappresentare solo l'informazione rilevante alla soluzione del problema.
- *Azioni*: occorre rappresentare solo gli aspetti funzionali alla soluzione del problema.
- *Toy problem*: un problema artificiale avente lo scopo di illustrare o mettere alla prova dei metodi di risoluzione. Ha una formulazione precisa e univoca. Utile per confrontare metodi diversi
- *Real-world problem*: problemi concreti, effettivi. Spesso non hanno una formulazione unica.

**Note:-**

E.g. di toy problems: problema dell'aspirapolvere, gioco dell'8, problema delle 8 regine.

**Possibili approcci:**

- *Blind*: usano esclusivamente la struttura del problema per cercare una soluzione.
- *Informati*: usano la struttura del problema e ulteriore conoscenza per guidare la ricerca.

## 1.2.2 Metodi di Ricerca non Informati (Blind Search)

**Definizione 1.2.3: Albero di Ricerca**

Un albero di ricerca è una struttura dati usata per trovare una soluzione a un problema di ricerca:

- Ogni nodo corrisponde a uno stato.
- I nodi figli sono costruiti tramite la funzione successore.
- Ogni nodo ha un riferimento al nodo padre (per ricostruire le soluzioni).
- L'albero è costruito a partire dal nodo corrispondente allo stato iniziale.
- L'albero diventa un grafo quando lo stesso nodo (NB: non lo stesso stato) può essere raggiunto tramite più percorsi.
- Un percorso che porta dal nodo iniziale a un nodo obiettivo è una soluzione.

**Note:-**

Gli alberi sono un caso specifico dei grafi.

**Formalizzando:**

- Un *grafo di ricerca*  $G = (\{n_i\}, \{e_{ij}\})$  è costituito da un insieme di nodi  $n_i$  e di archi  $e_{ij}$ .
- $e_{pq} \in \{e_{ij}\}$  rappresenta l'esistenza di un arco dal nodo  $n_p$  al nodo  $n_q$ , quindi  $n_q$  è successore di  $n_p$ .
- Ciascun arco  $e_{ij}$  ha associato un costo  $c_{ij}$ .
- L'esistenza di  $e_{ij}$  non implica l'esistenza di  $e_{ji}$ .

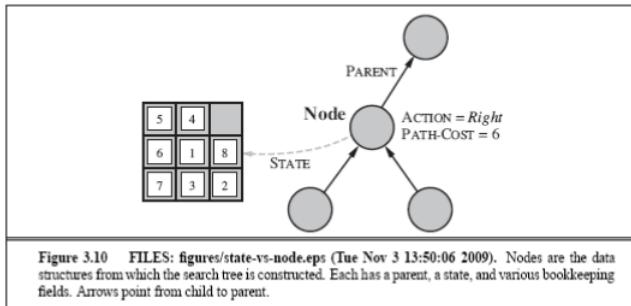


Figure 1.5: Esempio con il gioco dell'8.

**Domanda 1.7**

Se le strategie sono tante ve ne è una migliore? Come le confronto?

**Criteri di valutazione:**

- *Completezza*: garanzia di trovare una soluzione, se esiste.
- *Ottimalità*: garanzia di trovare una soluzione ottima (a costo minimo)<sup>2</sup>.
- *Complessità temporale*: quanto tempo occorre per trovare una soluzione.
- *Complessità spaziale*: quanta memoria occorre per effettuare la ricerca.

**Domanda 1.8**

Come si valuta la complessità?

- *Complessità computazionale*: dato un problema esistono infiniti algoritmi che lo risolvono.
- *Termine di paragone*:
  - Tempo.
  - Spazio.
- *Criterio di preferenza*: economicità.

**Note:-**

Per astrarre dal calcolatore utilizzato lo spazio e il tempo non sono metrici, ma parametrici. E.g. Numero di nodi creati o visitati.

È interessante vedere l'andamento del costo al variare della dimensione del problema.

<sup>2</sup>Corso di "Algoritmi e Complessità"

### 1.2.3 Lista di Strategie

#### Definizione 1.2.4: Ricerca in Ampiezza

La ricerca espande il nodo radice, poi tutti i suoi successori, poi tutti i discendenti di secondo livello, ecc. Si realizza gestendo la frontiera come una coda FIFO.



#### Valutazione:

- **Completezza:** se esiste un nodo obiettivo a una profondità finita  $d$ , la ricerca in ampiezza lo troverà a patto che il fattore di ramificazione  $b$  (cioè il numero di figli che un nodo può avere) sia finito.
- **Ottimalità:** la soluzione trovata è ottima solo se il costo del cammino è una funzione monotona crescente della profondità (es. tutte le azioni hanno lo stesso costo).
- **Complessità temporale:**  $O(b^{d+1})$ .
- **Complessità spaziale:**  $O(b^{d+1})$ , perché bisogna tenere in memoria sia la frontiera che gli antenati.

#### Definizione 1.2.5: Ricerca a Costo Uniforme

Nella ricerca a costo uniforme:

- Ogni nodo ha associato il costo del cammino con cui è stato raggiunto.
- La frontiera è mantenuta ottimale.
- A ogni iterazione espande il nodo appartenente a un cammino di costo minimo.

#### Note:-

Quando i costi sono tutti uguali diventa una ricerca in ampiezza.

#### Osservazioni 1.2.1

- Costo  $\neq$  Numero dei passi: il numero dei passi effettuati non conta, conta solo il costo dei cammini.
- Quando trova il nodo obiettivo non si ferma subito, prima controlla se vi sono cammini aperti di costo inferiore e nel caso prova a espanderli.

#### Valutazione:

- **Completezza:** garantibile solo se tutti i passi hanno costo  $\geq \epsilon > 0$ . È il costo minimo delle operazioni.
- **Ottimalità:** garantibile solo se tutti i passi hanno costo  $\geq \epsilon > 0$ . Non sa gestire la nozione di guadagno unita a quella di costo.
- **Complessità temporale e spaziale:** ordine del branching factor elevato al numero di passi del percorso ottimale se i costi dei passi fossero uniformi ( $O(b^{1+LC^*/\epsilon})$ ).

### Definizione 1.2.6: Ricerca in Profondità

La ricerca in profondità espande sempre uno dei nodi più profondi della frontiera, cioè uno dei più lontani dalla radice. L'espansione produce tutti i successori di un nodo. Quando la ricerca tenta di espandere un nodo che non ha successori, l'effetto è che il nodo viene rimosso e si "torna indietro" nell'albero per esplorare eventuali alternative.

#### Note:-

Può essere con Backtracking o senza Backtracking<sup>a</sup>.

<sup>a</sup>Meglio visto in "Intelligenza Artificiale e Laboratorio"

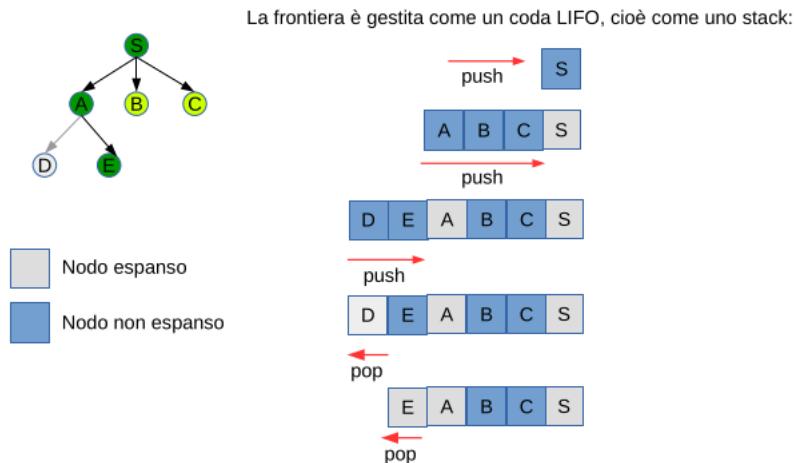


Figure 1.6: Ricerca in profondità.

#### Valutazione:

- *Completezza*: garantibile solo se tutti i cammini sono finiti.
- *Ottimalità*: in generale non è garantita.
- *Complessità temporale*: nel caso peggiore vengono percorsi tutti i nodi dell'albero ( $O(b^m)$ ), dove  $b$  è il branching factor e  $m$  è la profondità massima.
- *Complessità spaziale*:
  - Senza Backtracking:  $O(b * m)$ , perché occorre mantenere tutti i nodi del cammino esplorato più tutti i loro fratelli.
  - Con Backtracking:  $O(m)$

### Corollario 1.2.2 Ricerca in Profondità Limitata

Per evitare di entrare in branch infiniti viene introdotto un limite artificiale  $l$  (punto di taglio):

- Un nodo viene espanso solo se la sua profondità  $p \leq l$ .
- Altrimenti viene trattato come un nodo privo di successori.

#### Note:-

Tutti i cammini saranno lunghi al più  $l$ .

**Problemi:**

- Si riduce la completezza, perché una soluzione potrebbe trovarsi a profondità maggiore di  $I$ .
- Se  $l \gg d$  si perde efficienza.

**Definizione 1.2.7: Iterative Deepening**

Esegue una ricerca in profondità limitata, con iterazioni successive in cui la profondità massima viene aumentata via via.

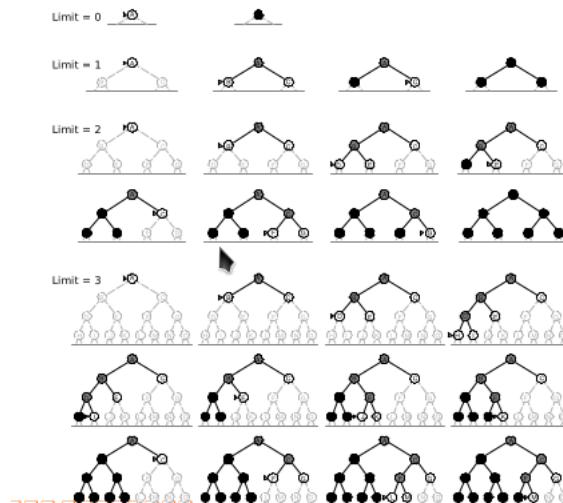


Figure 1.7: Iterative Deepening.

**Valutazione:**

- Combina ampiezza e profondità.
- È preferito quando lo spazio di ricerca è ampio e la profondità della soluzione non prevedibile.
- Ha una complessità spaziale modesta.
- La complessità temporale è alta.
- È completo se  $b$  è finito.
- È ottimo quando il costo è funzione non decrescente della profondità.

**Definizione 1.2.8: Ricerca Bidirezionale**

Composta da due ricerche:

- Forward dallo stato iniziale.
- Backward dallo stato obiettivo.

Termina quando le due ricerche si incontrano, quando le frontiere hanno intersezione non vuota.

**Note:-**

Se lo stato obiettivo non è unico si può introdurre uno stato fittizio raggiungibile da tutti gli stati obiettivo reali.

Sono possibili due andamenti:

- *A fronte d'onda*: quando non si ha informazione aggiuntiva si esplorano tutte le possibili operazioni.
- *A cono*: quando si ha informazione aggiuntiva si esplora parte delle operazioni.

## 1.3 Ricerca Informata

### Domanda 1.9

Le strategie di ricerca blind non sono efficienti. E' possibile rendere la ricerca più efficiente utilizzando della conoscenza sul problema? La conoscenza permette di focalizzare la ricerca verso le direzioni più promettenti?

#### Definizione 1.3.1: Funzione di Valutazione

Una strategia di ricerca informata ordina la frontiera sulla base di una funzione di valutazione  $f(n)$  applicata ai nodi:

- In base a  $f(n)$  si ottengono strategie differenti.
- $f(n)$  comprende una componente  $h(n)$  che restituisce una stima del minimo tra i costi che congiungono lo stato corrispondente al nodo  $n$  a uno stato goal.
- $h(n)$  è detta euristica.

#### Note:-

La strategia generale è detta *best-first search* e comprende greedy, A\* e RBFS.

#### Definizione 1.3.2: Ricerca Greedy

Viene scelto il nodo stimato più vicino a quello obiettivo ( $f(n) = h(n)$ ).

**Problemi:**

- Rischio di loop per vicoli ciechi.
- Stessi difetti della ricerca in profondità.

**Intuizione:**

- È possibile combinare la ricerca a costo uniforme con la ricerca greedy.
- *Costo uniforme*: espande per primi i nodi il cui raggiungimento dalla radice costa meno (guarda al passato).
- *Greedy*: espande per primi i nodi che promettono di raggiungere l'obiettivo spendendo meno (guarda al futuro).

### 1.3.1 A\*

#### Definizione 1.3.3: A\*

A\* considera grafi, generati a partire da un singolo nodo iniziale, in cui un sottoinsieme  $T$  di nodi è costituito da nodi obiettivo ( $T$  sta per target). Dato un qualsiasi nodo  $n$  del grafo, un obiettivo  $t$  è detto preferito per  $n$  se e solo se il costo del cammino ottimo  $h(n, t) \leq$  costo di qualsiasi altro cammino da  $n$  verso qualsiasi altro nodo di  $T$ .

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

- $g(n)$ : costo minimo di tutti i percorsi, visti fino a ora, che consentono di raggiungere il nodo  $n$  a partire dallo stato iniziale  $s$ .
- $h(n)$ : stima del costo minimo del proseguimento di percorso che consente di raggiungere un goal preferito di  $n$ .
- $f(n)$ : stima del costo minimo per raggiungere un goal preferito di  $n$  partendo da  $s$ .

$f^*(n) = g^*(n) + h^*(n)$ :

- $g^*(n)$ : costo minimo per raggiungere il nodo  $n$  a partire dallo stato iniziale  $s$  (calcolato considerando tutti i cammini possibili).
- $h^*(n)$ : costo minimo reale del proseguimento di percorso che consente di raggiungere un goal preferito a partire dal nodo  $n$ .
- $f^*(n)$ : costo minimo per raggiungere un goal preferito di  $n$  da  $s$ .

**Note:-**

Se si sono esplorate tutte le alternative  $f(x) = f^*(n)$ .

### Algoritmo di A\*:

1. Sia  $s$  il nodo iniziale.
2. Sia  $T$  l'insieme dei nodi obiettivo.
3. Segna  $s$  come aperto e calcola  $f(s)$ .
4. Seleziona il nodo aperto  $n$  avente valutazione minima.
5. Se  $n$  appartiene a  $T$ , marca  $n$  chiuso e termina.
6. Altrimenti marca  $n$  chiuso e applica l'operatore successore a  $n$  e:
  - Calcola il valore di  $f$  per tutti i successori  $n'$ .
  - Marca come aperti quei successori  $n'$  che non risultano già chiusi.
  - Rimarca come aperti quei successori  $n'$  che erano chiusi ma per cui è stato calcolato un valore  $f$  più basso di quello calcolato in precedenza (cioè sono stati raggiunti tramite un percorso migliore).

**Definizione 1.3.4: Euristica Ammissibile**

Un'euristica  $h$  è detta ammissibile quando

$$\forall n, h(n) \leq h^*(n)$$

dove  $h^*(n)$  è il costo minimo reale per raggiungere il nodo goal a partire dal nodo  $n$ .

**Note:-**

Intuitivamente un'euristica è ammissibile quando non fa mai stime per eccesso.

**Corollario 1.3.1 Ottimalità di A\***

Se:

- Un'euristica  $h$  è ammissibile.
- Tutti i passi hanno costo maggiore di 0.

Allora:

- $A^*$  termina e trova una soluzione ottima.
- In questo caso  $A^*$  è completa e ottimale.

### Osservazioni 1.3.1

Perché manchi l'ottimalità deve accadere che durante la ricerca l'algoritmo:

- Scelga un nodo obiettivo sub-ottimo.
- Al posto di un nodo.
- Che si trova su un cammino ottimo.

**Note:-**

Vedere dimostrazione su slides.

### Corollario 1.3.2 Euristica Monotona

Un'euristica ammissibile è monotona quando

$$\forall n h(n) \leq c(n, a, n') + h(n')$$

dove  $c$  è la funzione costo per andare da  $n$  a  $n'$  mediante l'azione  $a$ .

**Note:-**

Tuttavia ammissibile non vuol sempre dire informativa:  $h(n) = 0$  è ammissibile, ma non informativa.

### Valutazione:

- $A^*$  è ottimamente efficiente per qualsiasi euristica: non esiste alcun altro algoritmo ottimo che garantisca di espandere meno nodi di quelli espansi da  $A^*$ .
- Il numero di nodi espansi aumenta esponenzialmente con la profondità della soluzione ottima.
- $A^*$  mantiene in memoria tutti i nodi generati (è una ricerca in ampiezza).

### Funzioni euristiche nel gioco dell'8:

- In media occorrono 22 mosse per arrivare alla soluzione.
- Il branching factor è pari a 3.
- Albero esaustivo di ricerca: contiene  $3^{22}$  nodi.
- Grafo esaustivo di ricerca: 180000 nodi.
- Ma se si passa al problema del 15 il grafo *explode*:  $10^{23}$ .



Figure 1.8: Il grafo be like.

### 1.3.2 Approfondimento su Euristiche

Possibili euristiche per A\*:

- h1 (numero di tessere fuori posto): è ammmissible perché ogni tessera fuori posto deve essere spostata almeno una volta.
- h2 (*distanza di Manhattan*): è la somma della distanza di una tessera dalla sua posizione desiderata, contata in numero di tessere attraversate (originariamente di isolati attraversati) sulle ascisse più numero di tessere attraversate sulle ordinate. È ammmissible perché ogni mossa può spostare una tessera al più di una posizione più vicina al goal.

Qualità delle euristiche:

- La qualità di un'euristica può essere calcolata computando il branching factor effettivo  $b^*$ .
- $b^*$  = branching factor di un albero uniforme di profondità  $d$  che contiene  $N+1$  nodi.
- Le *euristiche migliori* hanno  $b^*$  bassi, vicini a 1.

#### Definizione 1.3.5: Problemi Rilassati

Un problema ne rilassa un altro quando toglie qualche vincolo. Il grafo degli stati di un problema rilassato è un supergrafo di quello del problema originario perché include transizioni che i vincoli di quest'ultimo non consentono (meno vincoli, più transizioni possibili<sup>a</sup>).

<sup>a</sup>Magari fosse così ovunque.

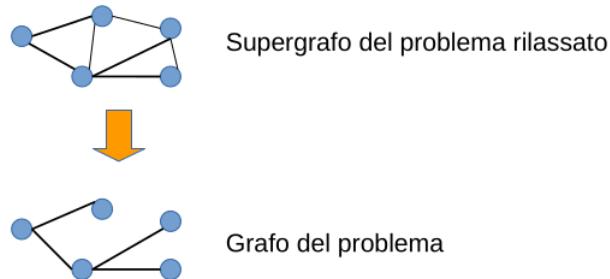


Figure 1.9: Rilassamento di un grafo.

#### Corollario 1.3.3 Absolver II

Absolver II è un esempio di programma che è in grado di generare automaticamente euristiche ammissibili per astrazione:

- Ha scoperto la prima euristica ammissibile per il cubo di Rubik.
- Ha scoperto un'euristica ammissibile per il problema dell'8 che è migliore di quelle precedentemente proposte

#### Note:-

Gli studi sulla generazione di euristiche continua oggi soprattutto nell'area di planning.

#### Definizione 1.3.6: Recursive Best First Search

RBFS trasforma la ricerca in ampiezza di A\* in una ricerca in profondità.

## 1.4 Strategie di Ricerca con Avversario

**Ambiente competitivo:**

- Multi-agente.
- Ogni agente ha *obiettivi*.
- Gli obiettivi di agenti diversi sono conflittuali, cioè il conseguimento degli obiettivi di un agente impedisce il conseguimento degli obiettivi degli altri agenti.
- I problemi di ricerca con avversario sono anche detti *giochi*.

**Tipologie di giochi:**

- *Condizioni di scelta:*
  - Informazione perfetta: gli stati del gioco sono totalmente esplicati per tutti gli agenti.
  - Informazione imperfetta: gli stati del gioco sono solo parzialmente esplicitati.
- *Effetti della scelta:*
  - *Deterministici:* gli stati sono determinati unicamente dalle azioni degli agenti.
  - *Stocastici:* gli stati sono determinati anche da fattori esterni (es: dadi).

	Informazione Perfetta	Informazione Imperfetta
Giochi deterministici	Scacchi, Go, Dama, Otello, Forza4, tris	MasterMind
Giochi stocastici	Backgammon, Monopoli	Scarabeo, Bridge, Poker... (giochi di carte) Risiko

Figure 1.10: Tipi di giochi.



2

Test2

