Capitolo 3

1 Problem Solving as Search

Definizione del problema

Il problem solving può essere rappresentato matematicamente come la ricerca di un percorso in un grafo diretto dal nodo iniziale (*start node*) a un nodo obiettivo (*goal node*).
Un agente opera in un modello basato su stati, senza incertezze e con obiettivi definiti.

Concetti chiave

- **Astrazione:** Rappresentare un problema reale come un grafo facilita l'uso di algoritmi di ricerca.
- **Ricerca:**
  - La ricerca è una parte fondamentale dell'intelligenza artificiale (IA).
  - Gli agenti ricevono solo una descrizione del problema, che consente di riconoscere una soluzione ma non un algoritmo per trovarla.
- **Problemi complessi:**
  - Alcuni problemi, come la crittografia, presentano spazi di ricerca chiari, ma le soluzioni possono essere inaccessibili entro tempi realistici (es. problemi NP-completi).

2 State Spaces

Definizione

Lo spazio degli stati è una rappresentazione generale di azioni intelligenti. Include tutte le informazioni necessarie per prevedere gli effetti di un'azione e determinare se uno stato è obiettivo.

Assunzioni principali

1. L'agente ha conoscenza perfetta dello stato dello spazio (*osservabilità completa*).
2. L'agente ha un set di azioni con effetti deterministici.

Esempio: Problema del Robot Delivery

Il robot deve trovare un percorso tra stanze (*nodi*). Le azioni, come "muoversi", sono rappresentate come *archi* tra nodi.

3 Graph Searching

Definizione di grafo

Un grafo è definito da:

1. Un set di nodi N.
2. Un set di archi A, ognuno connesso da un nodo iniziale e uno terminale.

Concetti

- **Costo degli archi:** Gli archi possono avere costi o pesi per rappresentare distanze o risorse.
- **Soluzione ottimale:** Un percorso con costo minimo dal nodo iniziale al nodo obiettivo.

Esempio

Problemi di routing, come il robot che trova il percorso più efficiente per spostarsi tra punti.

4 A Generic Searching Algorithm

Descrizione

Un algoritmo generico di ricerca espande iterativamente la *frontiera* dei cammini dai nodi iniziali.

Frontiera

- Contiene tutti i cammini possibili dal nodo iniziale al goal.
- Viene espansa iterativamente con nuovi cammini.

Concetti chiave

La selezione dei cammini dalla frontiera determina l'efficienza della ricerca e varia a seconda della strategia adottata.

5 Uninformed Search Strategies

Definizione

Le strategie di ricerca non informata non considerano la posizione del nodo obiettivo. Seguono regole generiche senza conoscenze specifiche.

Algoritmi principali

1. **Depth-First Search (DFS):**
   - Utilizza uno stack (LIFO).
   - Esplora profondamente un percorso prima di tornare indietro.
   - Può rimanere intrappolato in cicli se non gestito correttamente.
2. **Breadth-First Search (BFS):**
   - Utilizza una coda FIFO.
   - Genera percorsi in ordine crescente del numero di archi.
   - Ideale per trovare percorsi più corti.
3. **Lowest-Cost-First Search (LCFS):**
   - Cerca il percorso con il costo minimo.
   - Utilizza una coda prioritaria ordinata in base al costo.

6 Heuristic Search

Definizione

La ricerca euristica introduce una funzione euristica $h(n)$ che stima il costo dal nodo corrente al goal.

Tipi di ricerca euristica

1. **Heuristic Depth-First Search:**
   - Ordina i vicini in base a $h(n)$.
   - Seleziona il cammino con il costo stimato più basso.
2. **Best-First Search:**
   - Espande il nodo con valore $h(n)$ più basso sulla frontiera.
3. **A*:**
   - Combina il costo del percorso trovato $g(n)$ con $h(n)$: $f(n) = g(n) + h(n)$.
   - Garantisce ottimalità se h(n)h(n) è ammissibile (mai sovrastima il costo reale).

7 More Sophisticated Search

Refinements

1. **Cycle Checking:**
   - Evita cicli durante la ricerca memorizzando i nodi già espansi.
2. **Multiple Path Pruning:**
   - Rimuove percorsi non ottimali che condividono lo stesso nodo finale.
   - Garantisce che solo il percorso con costo minimo venga espanso.
3. **Iterative Deepening:**
   - Combina la memoria della DFS con l'ottimalità della BFS.
   - Ricomincia la ricerca aumentando progressivamente la profondità.
4. **Branch and Bound:**
   - Mantiene il miglior costo trovato e pruna i percorsi con costi peggiori.
5. **Ricerca Bidirezionale:**
   - Avvia ricerche dal nodo iniziale e dal goal simultaneamente.
   - Riduce il tempo di ricerca ma può essere complesso da implementare.
6. **Dynamic Programming:**
   - Calcola in modo iterativo i costi ottimali dal goal a ogni nodo.
   - Costruisce una tabella dei costi per garantire l'ottimalità.

8 Progettazione della Funzione Euristica

Proprietà

Una funzione euristica deve essere:

- **Ammissibile:** Non sovrastima mai il costo reale.
- **Consistente:** Rispetta la monotonicità, cioè $h(n) \leq h(m) + costo(n, m)$.

Metodi di definizione

1. Risolvere una versione semplificata del problema.
2. Mappare stati complessi in stati equivalenti più semplici.

Cache

Memorizzare i risultati intermedi per evitare calcoli ridondanti