

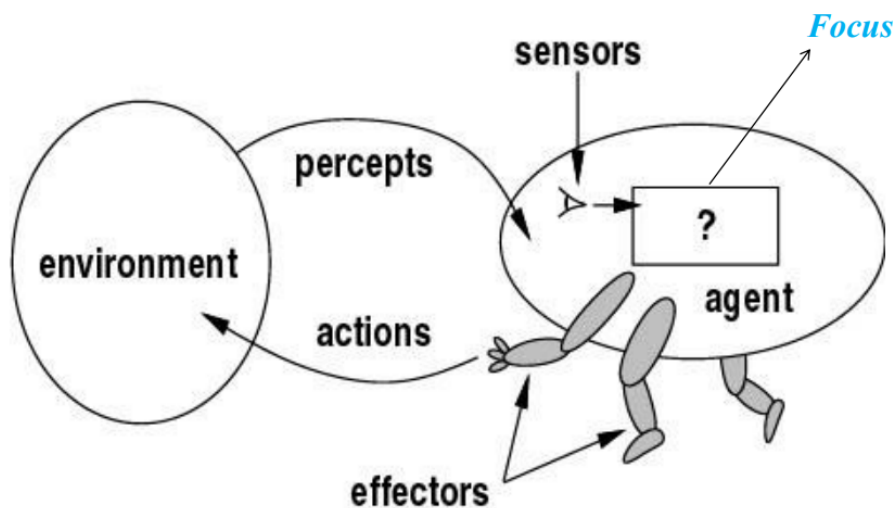
Introduzione all'Intelligenza Artificiale 2022

Ambra Manattini

February 2022

1 Agenti Intelligenti

Primo obiettivo: agenti per la risoluzione di problemi vista come ricerca in uno spazio di stati (**problem solving**)



Ciclo *percezione- azione*

L'agente esegue il ciclo percezione-azione. Il focus è il programma dell'agente

1.1 Caratteristiche degli agenti

- **Situati:** ricevono percezioni dall'ambiente e agiscono mediante *azioni*
- **Abilità sociale:** capaci di comunicare, collaborare, difendersi da altri agenti
- Hanno **credenze, obiettivi, intenzioni**
- **Embodied:** hanno un corpo fisico, fino a considerare i meccanismi delle emozioni



Figure 1: funzione agente: definisce l'azione da compiere per ogni sequenza percettiva

La scelta dell'agente è una funzione della **sequenza percettiva**, definisce l'azione da compiere in base a ogni sequenza percettiva, implementata da un programma agente

1.2 Agenti Razionali

Un agente razionale interagisce col suo ambiente in maniera efficace, perché serve un *criterio di valutazione* oggettivo dell'effetto delle azioni dell'agente (es costo minimo di un cammino alla soluzione)

Valutazione della prestazione

Misura di prestazione

- Esterna (come vogliamo che il mondo evolva)
- Scelta dal progettista a seconda del problema
- Possibilmente valutata su ambienti diversi

La razionalità dipende da

- Misura di prestazione
- Conoscenza pregressa dell'ambiente
- Percezioni presenti e passate
- Capacità dell'agente

Un **agente razionale** per ogni sequenza di percezioni compie l'azione che massimizza il valore atteso della misura delle prestazioni, considerando le sue percezioni passate e la sua conoscenza pregressa.

Non è perfetto e non conosce il futuro, potrebbero essere necessarie azioni di acquisizione di informazioni o esplorative. Le capacità dell'agente sono limitate. Raramente tutta la conoscenza dell'ambiente può essere fornita a priori, l'agente razionale deve essere in grado di modificare il suo comportamento con l'esperienza (*si adatta*)

Agente autonomo

È autonomo nella misura in cui il suo comportamento dipende dalla sua capacità di ottenere esperienza

Prestazione	Ambiente	Attuatori	Sensori
Arrivare alla destinazione, sicuro, veloce, ligio alla legge, viaggio confortevole, minimo consumo di benzina, profitti massimi	Strada, altri veicoli, pedoni, clienti	Sterzo, acceleratore, freni, frecce, clacson, schermo di interfaccia o sintesi vocale	Telecamere, sensori a infrarossi e sonar, tachimetro, GPS, contachilometri, accelerometro, sensori sullo stato del motore, tastiera o microfono

Figure 2: es. agente guidatore di taxi

1.3 Ambienti

Definire un problema per un agente significa caratterizzare l'ambiente in cui l'agente opera (*ambiente operativo*)

1.3.1 PEAS

Descrizione PEAS dei problemi

- **P**erformance — prestazione
- **E**nvironment — ambiente
- **A**ctuators — attuatori
- **S**ensors — sensori

1.4 Osservabilità

- Completamente osservabile: l'apparato percettivo è in grado di dare una conoscenza completa dell'ambiente o almeno tutto quello che serve a decidere l'azione, non c'è bisogno di mantenere uno stato del mondo esterno
- Parzialmente osservabile: sono presenti limiti o inaccuratezza dell'apparato sensoriale

1.4.1 Ambiente singolo/multi agente

- Agente/non agente: il mondo può cambiare per eventi, non necessariamente per azioni di agenti
- Multi agente competitivo (scacchi)
- Multi agente cooperativo: comunicazione, stesso obiettivo

1.4.2 Predicibilità

- Deterministico: stato successivo completamente determinato dallo stato corrente dell'azione
- Stocastico: esistono elementi di incertezza con associata probabilità
- Non deterministico: si tiene traccia di più stati possibili risultato dell'azione

1.4.3 Episodico/sequenziale

- Episodico: l'esperienza dell'agente è divisa in episodi atomici indipendenti
- Sequenziale: ogni decisione influenza le successive

1.4.4 Statico/dinamico

- Statico: il mondo non cambia mentre l'agente decide l'azione
- Dinamico: cambia nel tempo, va osservata la contingenza
- Semi dinamico: l'ambiente non cambia, ma la valutazione dell'agente sì

1.4.5 Discreto/continuo

- Possono assumere valori discreti o continui:
 - Lo stato
 - Il tempo
 - Le percezioni
 - Le azioni
- Combinatoria vs infinito

1.4.6 Noto/ignoto

Si riferisce allo stato di conoscenza dell'agente sulle leggi fisiche dell'ambiente; se l'agente conosce l'ambiente o deve compiere azioni esplorative.

Noto != Osservabile

1.5 Struttura di un agente

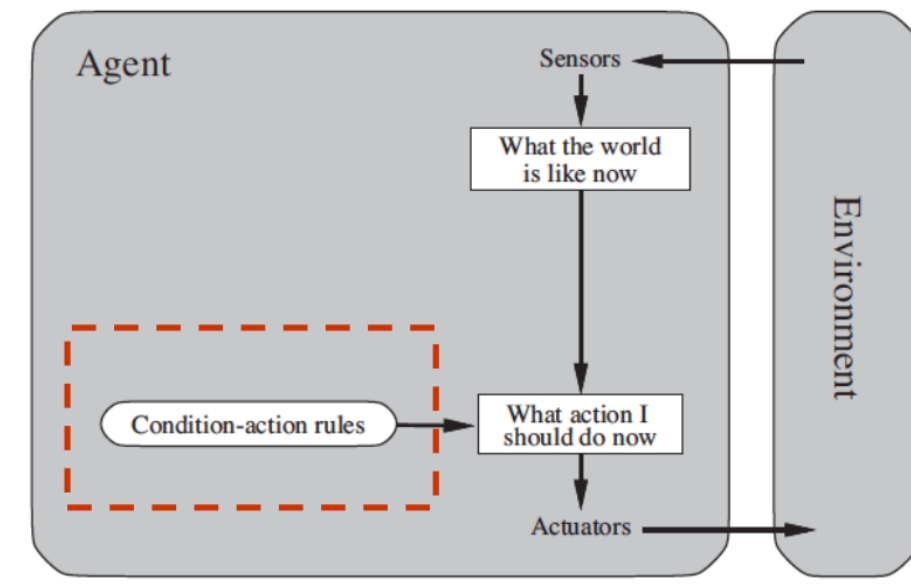
Un agente è formato da *architettura* e *programma*. Il programma dell'agente implementa la funzione $Ag : P \rightarrow Az$

1. Prende in input delle percezioni e ritorna delle azioni
2. Aggiorna la memoria in base alle percezioni
3. Sceglie la migliore azione possibile in base alla memoria
4. Aggiorna la memoria a seconda dell'azione che abbiamo fatto

1.5.1 Agente basato su tabella

La scelta dell'azione è un accesso a una "tabella" che associa un'azione a ogni possibile sequenza di percezioni. L'implementazione tramite actual tabella è ingestibile per vari motivi

1.6 Agenti reattivi semplici



Non c'è storia in memoria, non tiene conto delle cose precedenti.

function *Agente – Reattivo – Semplice*(percezione)

returns azione

persistent : regole, un insieme di regole condizione-azione (if-then)

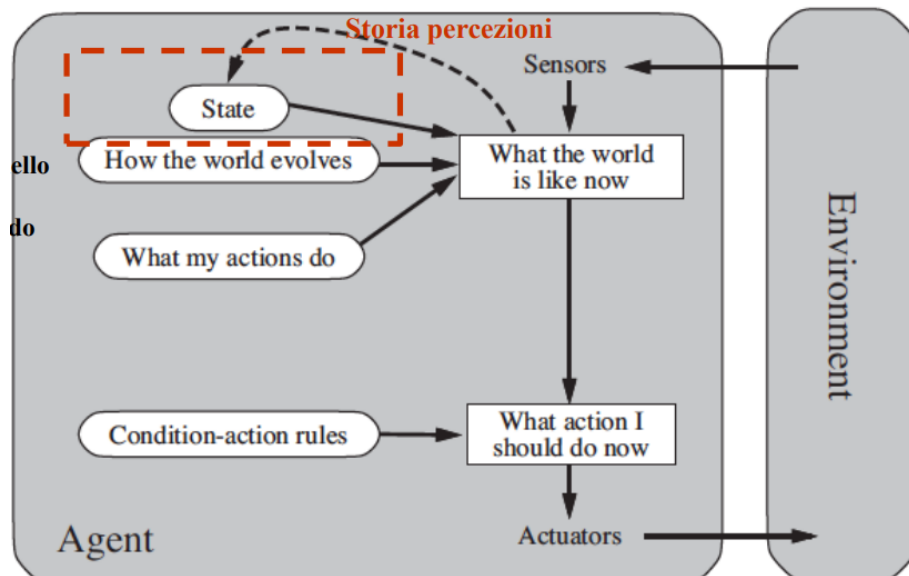
stato \leftarrow Interpreta – Input(percezione)

regola \leftarrow Regola – Corrispondente(stato, regole)

azione \leftarrow regola.Azione

return azione

1.7 Agenti basati su modello



Si introduce uno stato storia delle percezioni, abbiamo di nuovo lo stato e il modello del mondo

```
function Agente – Basato – su – Modello(percezione)
return azione
```

persistent: stato, una descrizione dello stato corrente
 modello, conoscenza del mondo
 regole, un insieme di regole condizione – azione
 azione, l'azione più recente

```
stato ← Aggiorna – Stato(stato, azione, percez., modello)
```

```
regola ← Regola – Corrispondente(stato, regole)
```

```
azione ← regola.Azione
```

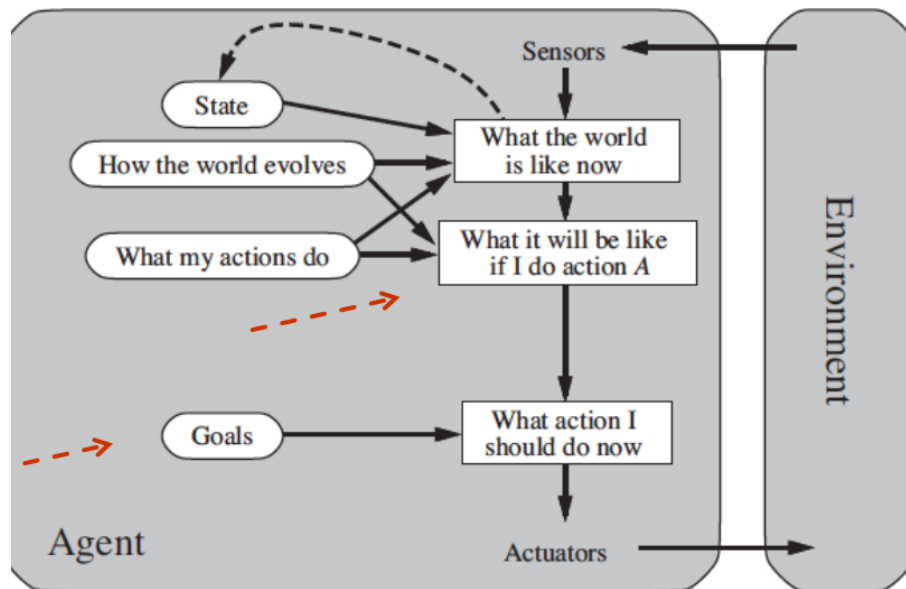
```
return azione
```

Lo stato si aggiorna in base alla sequenza di azioni fatte e il modello del mondo.

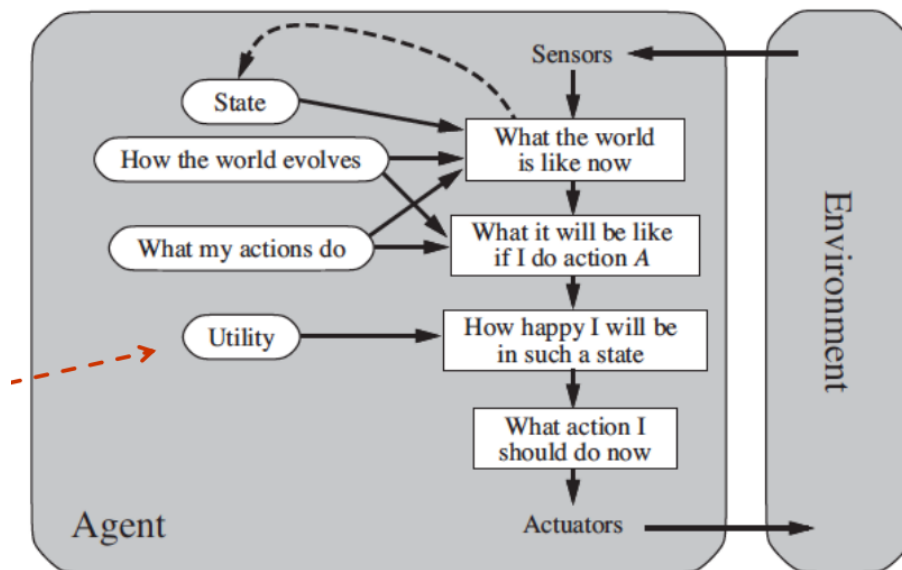
1.8 Agente con obiettivo

Guidati da un obiettivo nella scelta dell'azione, l'azione migliore dipende dall'obiettivo che devo raggiungere, devo pianificare una sequenza di azioni per raggiungere questo obiettivo.

Meno efficienti ma più flessibili (può cambiare l'obiettivo)



1.9 Agenti con valutazione di utilità



Obiettivi alternativi (o più modi per raggiungerlo) l'agente decide verso quali di questi muoversi, necessita di una funzione di *utilità* che associa ad uno stato obiettivo un numero reale. La funzione di utilità tiene conto anche della probabilità di successo: *utilità attesa*.

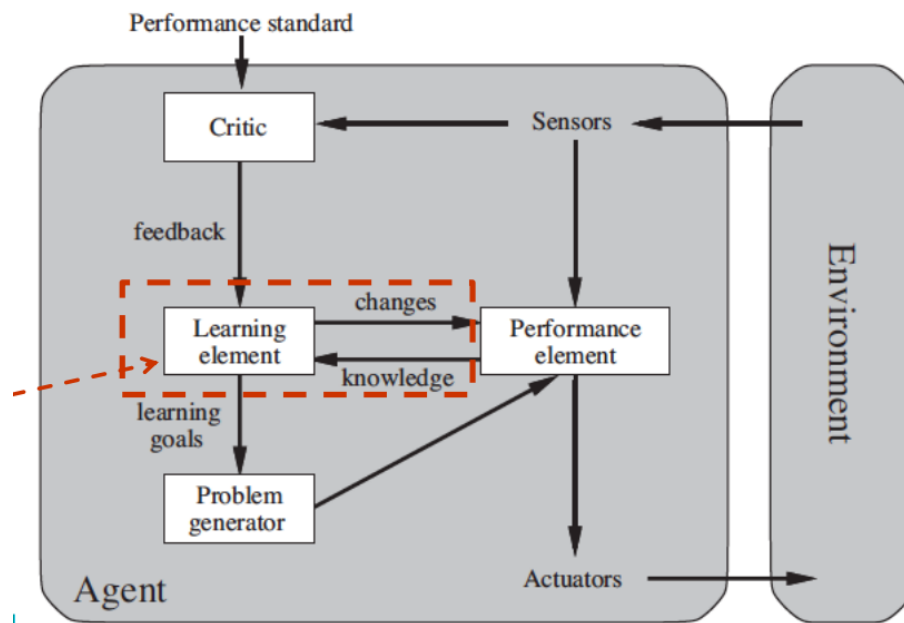


Figure 3: Agenti che apprendono

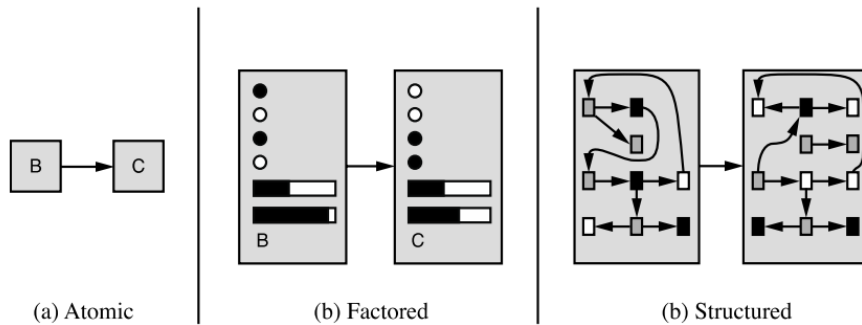
1.10 Agenti che apprendono

- Componente di apprendimento: produce cambiamenti al programma agente, apprendendo dall'ambiente
- Elemento esecutivo
- Elemento critico
- Generatore di problemi

1.10.1 Tipi di rappresentazione

Tipi di rappresentazione

Stati e transizioni



- Rappresentazione atomica (stati)
- Rappresentazione fattorizzata (piu variabili e attributi)
- Rappresentazione strutturata (aggiunge relazioni)

2 Agenti risolutori di problemi

Adottano il paradigma della risoluzione di problemi come ricerca in uno spazio di stati (problem solving). Agenti con modello che adottano una rappresentazione atomica dello stato. Sono particolari agenti con obiettivo che pianificano l'intera sequenza di mosse prima di agire

2.0.1 Processo di risoluzione

Passi da seguire:

1. Determinazione obiettivo (un insieme di stati in cui l'obiettivo è soddisfatto)
2. Formulazione del problema *Design (qui ancora umano)*
 - rappresentazione degli stati
 - rappresentazione delle azioni
3. Determinazione della soluzione mediante **ricerca** (un piano)
4. Esecuzione del piano *Qui soluzione algoritmica*

Assunzioni

- Ambiente statico
- Osservabile
- Discreto
- Deterministico

2.0.2 Formulazione del problema

Un problema può essere definito formalmente mediante cinque componenti:

1. Stato iniziale
2. Azioni possibili in s : $Azioni(s)$
3. Modello di transizione:
Risultato: $stato \times azione \rightarrow stato$
Risultato(s, a) = s' , uno stato **successore**
1, 2 e 3 definiscono implicitamente lo **spazio degli stati**
4. Test obiettivo:
 - Un insieme di stati obiettivo
 - Goal-Test: $stato \rightarrow true, false$
5. Costo del cammino:
 - somma dei costi delle azioni (costo dei passi)
 - costo di passo: $c(s, a, s')$
 - Il costo di un'azione/passaggio non è mai negativo

3 Algoritmi di ricerca

Gli algoritmi di ricerca prendono in input un problema e restituiscono un **cammino soluzione**

3.1 Problema itinerario

3.1.1 Formulazione

Stati: le città

1. Stato iniziale: città di partenza $In(Arad)$
2. Azioni: spostarsi su una città vicina collegata $Azioni(In(Arad)) = Go(Sibiu), Go(Zerind)...$
3. Modello di transizione $Risultato(In(Arad), Go(Sibiu)) = In(Sibiu)$
4. Test Obiettivo: $In(Bucarest)$

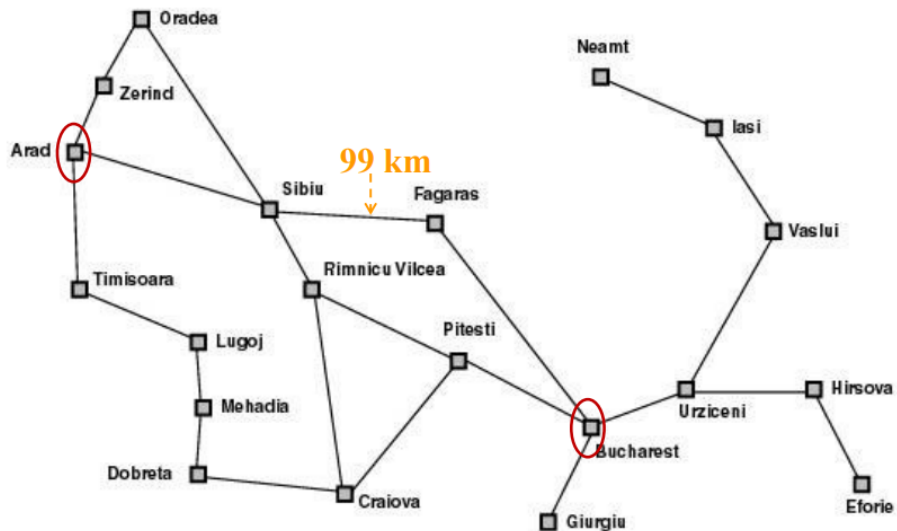


Figure 4: caso: trovare il percorso più breve da una città di partenza a una di arrivo

5. Costo del cammino: somma delle lunghezze delle strade

Lo spazio degli stati coincide con la rete di collegamenti tra città

Misura delle prestazioni

Costo totale = costo della ricerca + costo del cammino soluzione

3.1.2 Ricerca della soluzione

Generazione di un **albero di ricerca** sovrapposto allo **spazio degli stati**

Ricerca: approfondire un'opzione, da parte le altre e riprenderle se non trova soluzione

3.2 Ricerca ad albero

3.3 I nodi dell'albero di ricerca

Un nodo n è una struttura dati con 4 componenti

- Stato: $n.stato$
- Nodo padre: $n.padre$
- Azione effettuata per generarlo: $n.azione$
- Costo del cammino dal nodo iniziale al nodo n : $n.costo - cammino$ indicata come $g(n) (= padre.costo - cammino + costo - passo ultimo)$

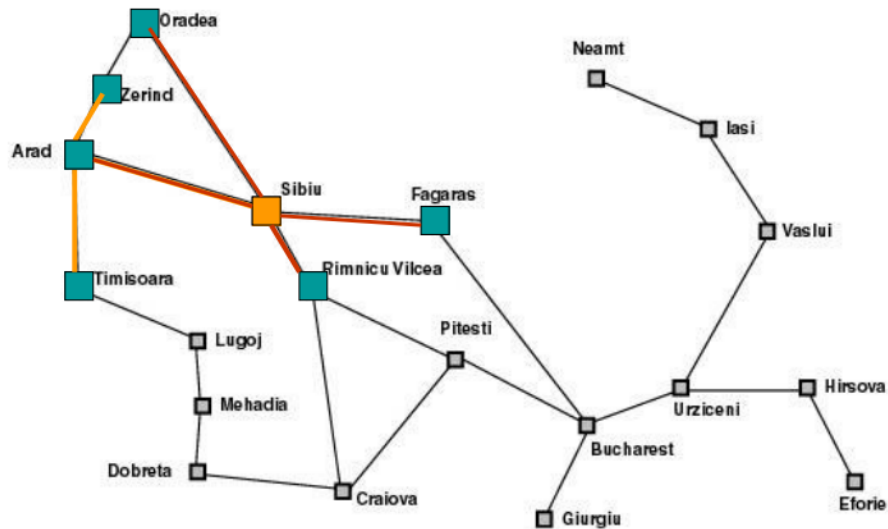


Figure 5: albero di ricerca sullo spazio degli stati

(a) The Initial state

Arad

(b) After expanding Arad

Il nodo è espanso

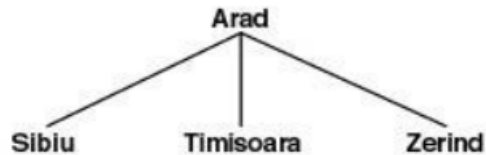


Figure 6: Stato iniziale e nodo espanso

Algorithm 1 Ricerca ad albero

function Ricerca-Albero (*problema*)

returns soluzione oppure **fallimento**

Inizializza la frontiera con stato iniziale del problema

loop do

if la frontiera è vuota **then return** fallimento

Scegli un nodo foglia da espandere e rimuovilo dalla frontiera

if il nodo contiene uno stato obiettivo

then return la soluzione corrispondente

Espandi il nodo e aggiungi i successori alla frontiera

3.4 Struttura dati per la frontiera

Frontiera: Lista dei nodi in attesa di essere espansi (foglie dell'albero di ricerca). E' implementata come una cosa con operazioni (empty, POP, Inert)

3.4.1 Strategie di ricerca

- FIFO \rightarrow BF(Breadth-first): viene estratto l'elemento più vecchio
- LIFO \rightarrow DF(Depth-first): viene estratto il più recentemente inserito
- Coda con priorità \rightarrow UC: viene estratto quello con priorità più alta in base a una funzione di ordinamento, si riordina ad ogni inserimento di nodi.

3.4.2 Valutazione di una strategia

- Completez

4 Ricerca euristica

La ricerca esaustiva è difficilmente applicabile in problemi di complessità esponenziale. Vogliamo usare la conoscenza euristica per fare scelte "oculate" (conoscenza del problema ed esperienza). La conoscenza euristica consente di trovare una buona soluzione in tempi accettabili

4.1 Funzioni di valutazione euristica

Conoscenza del problema data tramite una *funzione di valutazione* f

4.2 Algoritmo di ricerca Best-First

4.2.1 Greedy Best-First

4.3 Algoritmo di tipo A

Un algoritmo A è un algo