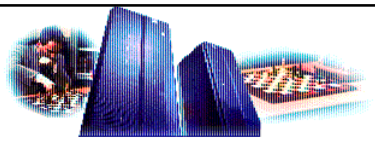
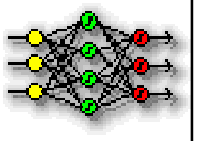




# Alberi di decisione e agenti

Alberto Borghese

Università degli Studi di Milano
Laboratorio di Sistemi Intelligenti Applicati (AIS-Lab)
Dipartimento di Informatica
Alberto.borghese@unimi.it

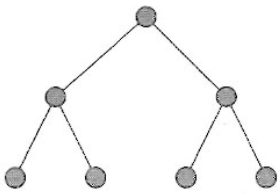


## Sommario

Alberi di decisione.

Agenti

# Decision trees



Rappresentazione mediante grafo orientato (dall'alto al basso)

Comparare con lo STG della FSM

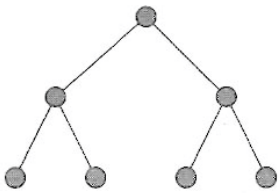
DECISION THEORY: Aim is to develop an **action plan** is to build an to achieve a **given goal (control problem)**.

COMPUTER SCIENCE (machine learning): the tree is built from the data, and each arch represents a probability or the learnt cost of the transition.

COMPUTER SCIENCE (data mining): the tree is used to **classify (classification problem).**



# How do they work?



We need a sequence of actions (control or incremental classification) to move through the tree.

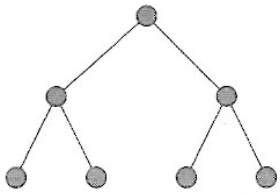
1) Definition of the tree
2) Computation of the path
3) Use of the sequence of action

Formulate-Search-Execute

# Nodes and states

A state is a (representation of) a physical configuration, that corresponds to a situation

A node is a data structure constituting part of a search tree includes state, parent node, action, path cost $g(x)$, depth





# Example: Romania

- On holiday in Romania; currently in Arad.
- Flight leaves tomorrow from Bucharest

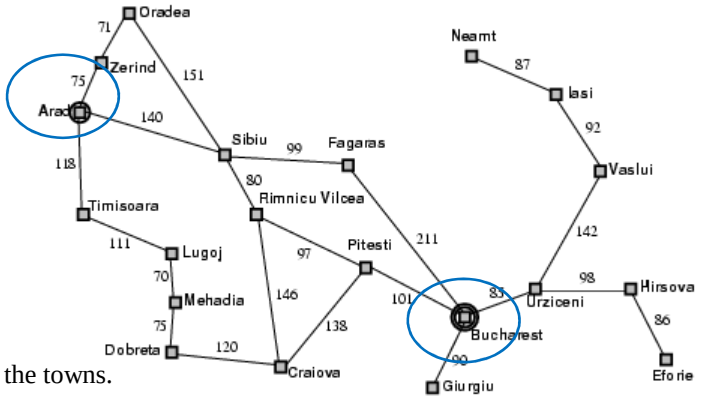
- Formulate goal:
  - be in Bucharest

- Formulate problem:
  - states: various cities
  - actions: drive between cities

- Find solution:
  - sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest

# Example: route finding - formalization

- From Arad to Bucarest




State: ensemble of the towns.

Goal: reach Bucarest.

Action: choice of a possible direction from the actual town.

Cost: length of the path.

Initial state: Arad

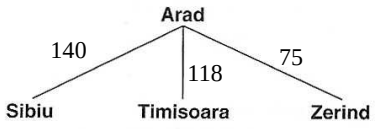


# Example: route finding – Costruzione dell'albero di ricerca

(a) The initial state

Arad

(b) After expanding Arad

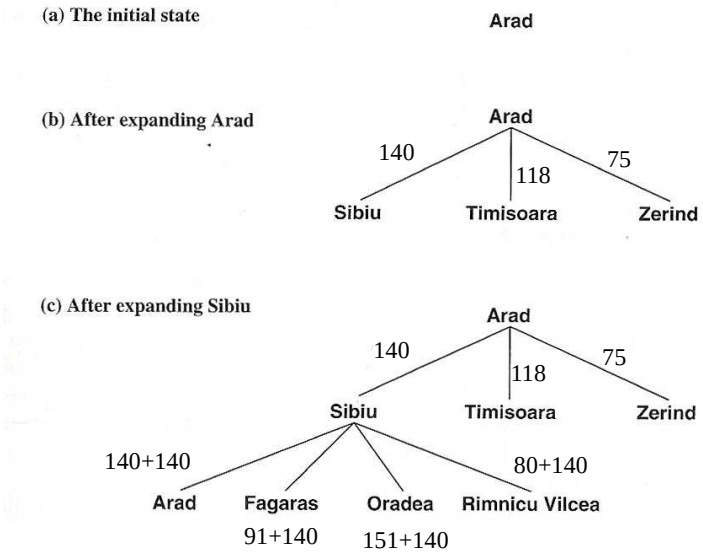



Which action is best for reaching the goal, in each state?

In Arad we have different choices, we have to decide which next town to choose. **How?**

We suppose that we have in hands a route map (known environment)

# Example: route finding – Costruzione dell'albero di ricerca



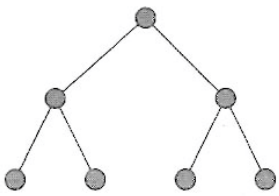



# Tecniche di search

Visito l'albero e certo la soluzione migliore.

Ricerca partendo da un nodo e analizzando i nodi figli fino a quando non si arriva ai nodi foglia (frontiera).

**Tree-search.**



- Valutazione in termini di:
  - - Completezza: la ricerca trova una soluzione, se esiste?
  - - Tempo: quanto richiede il calcolo di una soluzione?
  - - Complessità: quanta memoria viene richiesta per la ricerca?
  - - Ottimalità: la ricerca trova la soluzione ottima quango esistono più soluzioni possibili?

Techiche prive di informazioni a-priori o con informazioni a-priori.

# Search algorithm

```
function TREE-SEARCH( problem) returns a solution, or failure
  initialize the frontier using the initial state of problem
  loop do
    if the frontier is empty then return failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state then return the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier
```

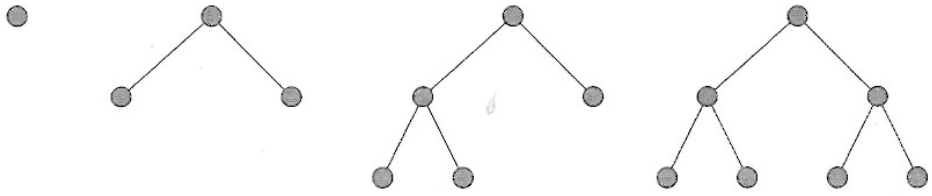
```
function GRAPH-SEARCH( problem) returns a solution, or failure
  initialize the frontier using the initial state of problem
  initialize the explored set to be empty
  loop do
    if the frontier is empty then return failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state then return the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set
```

**Figure 3.7** An informal description of the general tree-search and graph-search algorithms. The parts of GRAPH-SEARCH marked in bold italic are the additions needed to handle repeated states.



# Breadth-first search (larghezza prima)

b = branching factor



Definisco il costo di ciascun nodo figlio

Ricerca completa.
Ricerca ottima..

$N = 1 + b^1 + b^2 + b^3 + \ldots + b^{d-1} = b^d - 1$
Crescita più che polinomiale (esponenziale) in tempo e spazio

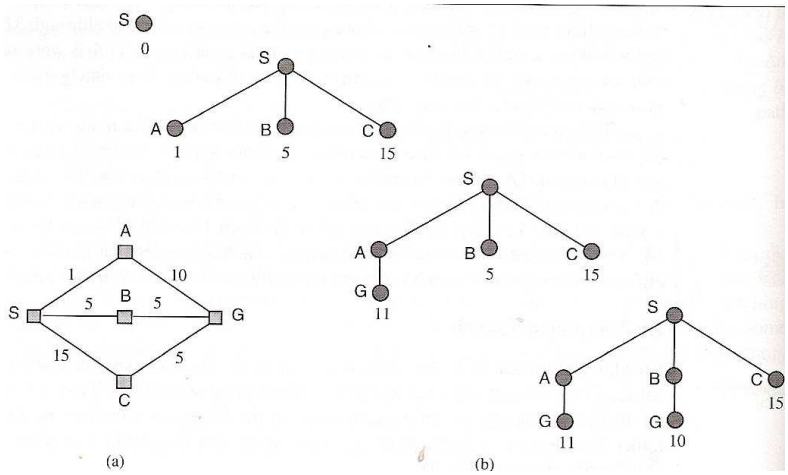
# Alcuni risultati (b=10)

| Profondità | Nodi | Tempo | Memoria |
|---|---|---|---|
| 0 | 1 | 1 ms | 100 byte |
| 2 | 111 | 0.1 s | 11 Kbyte |
| 4 | 11,111 | 11 s | 1 Mbyte |
| 6 | $10^6$ | 18 min | 111 Mbyte |
| 8 | $10^8$ | 31 h | 11 Gbyte |
| 10 | $10^{10}$ | 128 gg | 1 Tbyte |
| 12 | $10^{12}$ | 35 anni | 111 Tbyte |
| 14 | $10^{14}$ | 3500 anni | 11,111 Tbyte |

Memory requirements and computational time, make optimal search unfeasible as can be expected by a NP problem.



# Uniform cost search

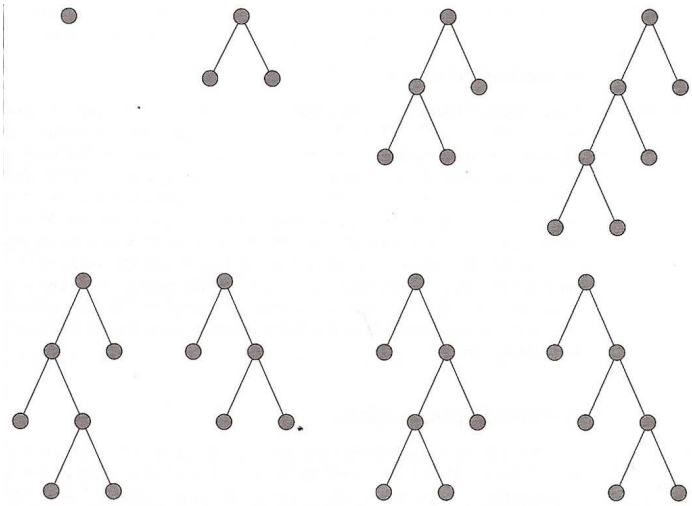



Ricerca completa.
Ricerca ottima..

g(SUCCESSOR(n)) ≥ g(n).

# Depth-first search

Come diminuire l'occupazione di memoria?



La memoria richiesta è pari a: M = b d (breadth-first: $M = b^d$)
d = 12, b = 10; 12 Kbyte versus 111 Tbyte!!



# Depth-limited search

Problemi della Depth-first search:

- Problemi per alberi profondi.
- Se prendo il ramo sbagliato, devo continuare ad esplorarlo, anche se, dopo pochi livelli mi potrei accorgere che è l'albero sbagliato.

Come evitare ciò?

Imporre una limitazione alla profondità dell'albero visitato. Questo si può fare in modo informato quando si hanno informazioni a-priori sui dati.

Questo limite richiede informazioni e non è detto che sia un buon limite.

# Sommario

Alberi di decisione

Agenti



# L'agente

- Inizialmente l'attenzione era concentrata sulla progettazione dei sistemi di "controllo". Valutazione, sintesi…
- L'intelligenza artificiale e la "computational intelligence" hanno consentito di spostare l'attenzione sull'apprendimento delle strategie di controllo e più in generale di comportamento.
- Macchine dotate di meccanismi (algoritmi, SW), per apprendere.

# Why agents are important?

**Agente** (software): essere software che svolge servizi per conto di un altro programma, solitamente in modo automatico ed invisibile. Tali software vengono anche detti agenti intelligenti

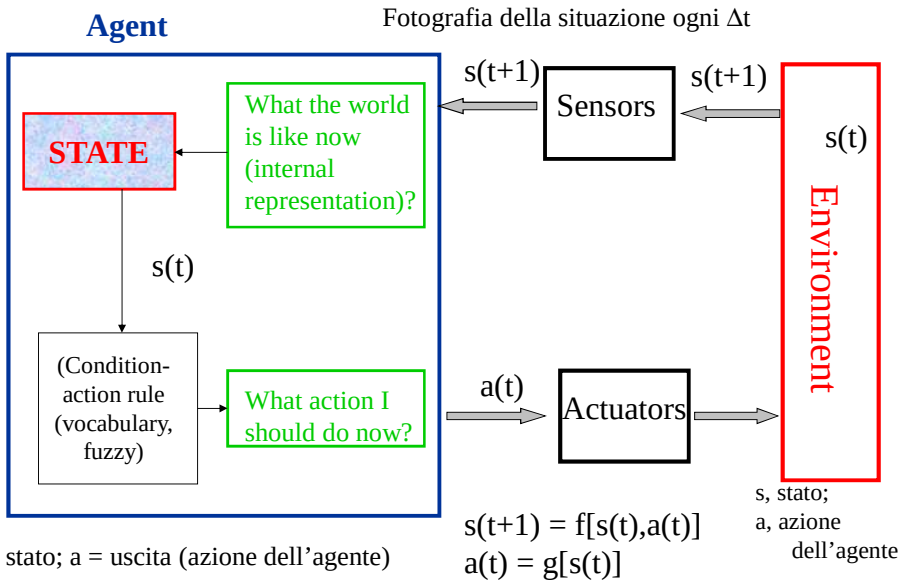
"They are seen as a natural metaphor for conceptualising and building a wide range of complex computer systems (the world contains many passive objects, but it also contains very many *active* components as well);

They cut across a wide range of different technology and application areas, including telecoms, human-computer interfaces, distributed systems, WEB and so on;

They are seen as a natural development in the search for ever-more powerful abstractions with which to build computer systems."



# Schematic diagram of an agent



# How agents solve a problem

Formulate a problem. Through analysis. State, action, indentification.

Solve the problem (by searching).

Implement the solution (execute).

Evaluate the implemented solution.

- Success or fail? Adequate or not adequate?
- How much adequate? How to measure the success or failure of the performance?
- Optimization of the performance to create better agents.

- Solve a problem = achieve a given goal (= reach a final state or avoid certain states)

- An agent can examine different sequences of actions (deterministic or stochastic response by the environment) and search the best sequence.



# Agente

- Può scegliere un'azione sull'ambiente tra un insieme continuo o discreto.

- L'azione dipende dalla situazione. La situazione è riassunta nello **stato** del sistema.

- L'agente monitora continuamente l'ambiente (input); l'ambiente modifica continuamente lo stato.

- La scelta dell'azione è non banale e richiede un certo grado di "intelligenza".

- L'agente ha una memoria "intelligente". Non può tenere in memoria tutto quanto successo nel passato.

# I vari tipi di apprendimento

$x(t+1) = f[x(t), a(t)]$ Ambiente
$a(t) = \mathbf{g}[x(t)]$ Agente

**Supervisionato** (learning with a teacher). Viene specificato per ogni pattern di input, il pattern desiderato in output.

**Semi-Supervisionato**. Viene specificato solamente per **alcuni** pattern di input, il pattern desiderato in output.

**Non-supervisionato** (learning without a teacher). Estrazione di similitudine statistiche tra pattern di input. Clustering. Mappe neurali.

**Apprendimento con rinforzo** (reinforcement learning, learning with a critic). L'ambiente fornisce un'informazione puntuale, di tipo qualitativo, ad esempio success or fail.



# I gruppi di algoritmi

Classification
Predictive regression

Clustering (data mining)

Reinforcement Learning

# Reinforcement learning

Spesso si ha a disposizione solamente un'informazione qualitativa (a volte binaria, giusto/sbagliato successo/fallimento), puntuale.

Questa è un'informazione qualitativa**.**

*L'informazione disponibile si chiama* ***segnale di rinforzo.*** *Non dà alcuna informazione su come aggiornare il comportamento dell'agente (e.g. i pesi). Non è possibile definire una funzione costo o un gradiente.*

*Obbiettivo: creare degli agenti "intelligenti" che abbiano una "machinery" per apprendere dalla loro esperienza.*



# Reinforcement Learning: caratteristiche

- Apprendimento mediante interazione con l'**ambiente**. Un agente isolato non apprende.
- L'apprendimento è funzione del raggiungimento di uno o più **obbiettivi.**
- Non è necessariamente prevista una ricompensa ad ogni istante di tempo.
- Le azioni vengono valutate mediante la ricompensa a lungo termine ad esse associata (**delayed reward**). Il meccanismo di ricerca delle azioni migliori è imparentato con la ricerca euristica: **trial-and-error**.
- **L'agente sente l'input, modifica lo stato e genera un'azione che massimizza la ricompensa a lungo termine.**

# Exploration vs Exploitation

Esplorazione (exploration) dello spazio delle azioni per scoprire le azioni migliori. Un agente che esplora solamente raramente troverà una buona soluzione.

Le azioni migliori vengono scelte ripetutamente (exploitation) perchè garantiscono ricompensa (reward). Se un agente non esplora nuove soluzioni potrebbe venire surclassato da nuovi agenti più dinamici.

Occorre non interrompere l'esplorazione.

Occorre un approccio statistico per valutare le bontà delle azioni.

Exploration ed exploitation vanno bilanciate. Come?



# Ambiente

- L'agente ha un comportamento goal-directed ma agisce in un ambiente incerto non noto a priori o parzialmente noto.
- Esempio: planning del movimento di un robot.
- Un agente impara interagendo con l'ambiente. Planning può essere sviluppato mentre si impara a conoscere l'ambiente (mediante le misure operate dall'agente stesso). La strategia è vicina al trial-and-error.
- L'agente impara facendo. Deve selezionare i comportamenti che ripetutamente risultano favorevoli a lungo termine.

# Esempi

Un giocatore di scacchi. Per ogni mossa ha informazione sulle configurazioni di pezzi che può creare e sulle possibili contro-mosse dell'avversario.

Una gazzella in 6 ore impara ad alzarsi e correre a 40km/h.

Come fa un robot veramente autonomo ad imparare a muoversi in una stanza per uscirne? (cf. competizione Robocare@home).

Come impostare i parametri di una raffineria (pressione petrolio, portata….) in tempo reale, in modo da ottenere il massimo rendimento o la massima qualità?



# Caratteristiche degli esempi

Parole chiave:

- Interazione con l'ambiente. L'agente impara dalla propria esperienza.
- Obbiettivo dell'agente.
- Incertezza o conoscenza parziale dell'ambiente.

Osservazioni:

- Le azioni modificano lo stato (la situazione), cambiano le possibilità di scelta in futuro (delayed reward).
- L'effetto di un'azione non si può prevedere completamente.
- L'agente ha a disposizione una valutazione globale del suo comportamento. Deve sfruttare questa informazione per migliorare le sue scelte. Le scelte migliorano con l'esperienza.
- I problemi possono avere orizzonte temporale finito od infinito.

# I tue tipi di rinforzo

L'agente deve scoprire quale azione (policy) fornisca la ricompensa massima provando le varie azioni (trial-and-error) sull'ambiente.

*"Learning is an adaptive change of behavior and that is indeed the reason of its existence in animals and man (K. Lorentz, 1977).*

Rinforzo puntuale istante per istante, azione per azione (condizionamento classico).

Rinforzo puntuale "una-tantum" (condizionamento operante), viene rinforzato una catena di azioni, un comportamento.



# Il Condizionamento classico

L'agente deve imparare una (o più) trasformazione tra input e output. Queste trasformazioni forniscono un comportamento che l'ambiente premia.

Il segnale di rinforzo è sempre lo stesso per ogni coppia input – output: **Reward istantaneo**.

Esempio: risposte riflesse Pavolviane. Campanello (stimolo condizionante) prelude al cibo. Questo induce una risposta (salivazione). La risposta riflessa ad uno stimolo viene evocata da uno stimolo condizionante.

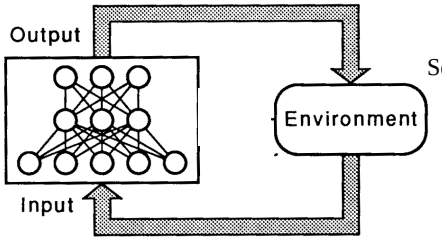
Stimolo-Risposta. Lo stimolo condizionante (campanello = input) induce la salivazione (uscita) in risposta al campanello.

# Condizionamento operante

*Reinforcement learning (operante).*

Interessa un comportamento. Una sequenza di input / output che può essere modificata agendo sui parametri che definiscono il comportamento dell'agente. Il condizionamento arriva in un certo istante di tempo (spesso una-tantum) e deve valutare tutta la sequenza temporale di azioni, anche quelle precedenti nel tempo.

$a(t) = f(s(t))$
$s(t+1) = (g(s(t), a(t))$




Comportamenti
=
Sequenza di azioni

Intreccio di azioni e reazioni dell'ambiente: $s_0$, $a_0$; $s_1$, $a_1$; $s_2$, $a_2$....



# Gli attori del RL

Policy. Descrive l'azione scelta dall'agente: mapping tra stato (input dall'ambiente) e azioni. Funzione di controllo. Le policy possono avere una componente stocastica. Viene utilizzato un modello adeguato del comportamento dell'agente (e.g. tabella, funzione continua parametrica…).

Reward function. Ricompensa **immediata**. Associata all'azione intrapresa in un certo stato. Può essere data al raggiungimento di un goal (esempio: successo / fallimento). E' uno scalare (può essere associato allo stato e/o input e/o stato prossimo). Rinforzo primario.

Value function. "Cost-to-go". Ricompensa a **lungo termine**. Somma dei reward: costi associati alle azioni scelte istante per istante + costo associato allo stato finale. Orizzonte temporale ampio. Rinforzo secondario. Ricompensa attesa. Cost-to-go per raggiungere il goal.

Ambiente. Descrive tutto quello su cui agisce la policy. Può essere non noto o parzialmente noto. L'agente deve costruirsi una rappresentazione implicita dell'ambiente attraverso la value function (rappresentazione utilitaristica).

- Quale delle due ricompense è più difficile da ottenere?
- L'agente agisce per massimizzare la funzione Value o Reward?

# Proprietà del rinforzo

L'ambiente o l'interazione può essere complessa.

Il rinforzo può avvenire solo dopo una più o meno lunga sequenza di azioni (delayed reward).

E.g. agente = giocatore di scacchi.
ambiente = avversario.

Problemi collegati:
temporal credit assignement.
structural credit assignement.

L'apprendimento non è più da esempi, ma dall'osservazione del proprio comportamento nell'ambiente.



# Meccanismo di apprendimento nel RL

Ciclo dell'agente (le tre fasi sono sequenziali):

1) Implemento una policy
2) Aggiorno la Value function
3) Aggiorno la policy.

# Il RL

•Reinforcement learning. L'agente viene modificato, rinforzando le azioni che sono risultate buone a lungo termine. E' quindi una classe di algoritmi iterativi.

•Self-discovery of a successful strategy (it does not need to be optimal!). La strategia (di movimento, di gioco) non è data a-priori ma viene appresa attraverso **trial-and-error**.

•Credit assignement (temporal and structural).

•Come possiamo procedere in modo efficiente nello scoprire una strategia di successo? Cosa vuol dire modificare l'agente?



# Sommario

Alberi di decisione

Agenti