

Ricerca in ambienti più complessi

- ▶ Azioni non deterministiche e ambiente parzialmente osservabile
 - ▶ Piani condizionali, ricerca AND-OR, stati credenza
- ▶ Ambienti sconosciuti e problemi di esplorazione (percezioni forniscono nuove informazioni dopo l'azione)
 - ▶ Ricerca online

Ricerca in ambienti più complessi

- ▶ Gli agenti risolutori di problemi “classici” assumono:
 - ▶ Ambienti completamente osservabili
 - ▶ Azioni/ambienti deterministici
- ▶ Il piano generato è una sequenza di azioni che può essere generato *offline* e eseguito senza imprevisti
- ▶ Le percezioni non servono se non nello stato iniziale

Ricerca in ambienti più complessi

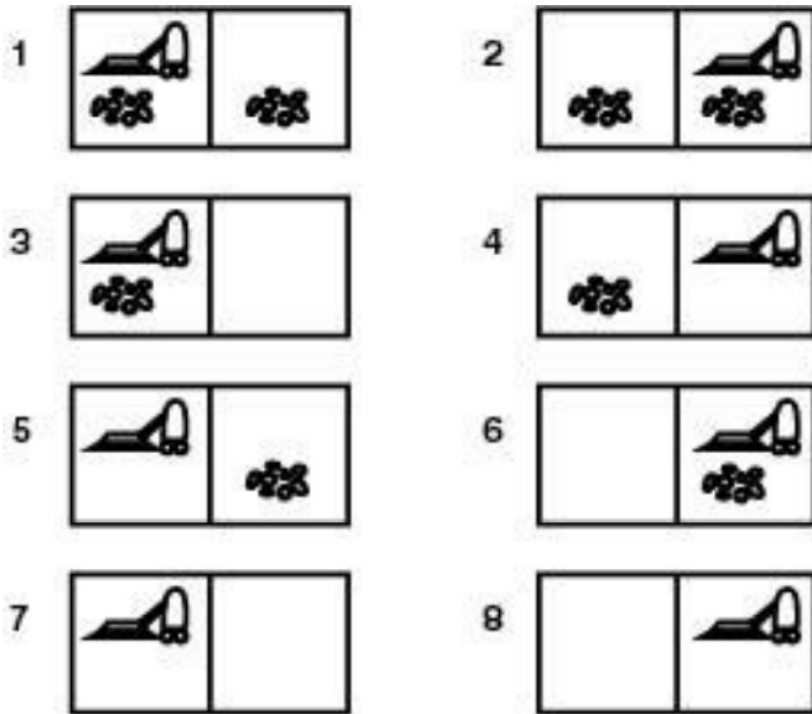
- ▶ In un ambiente parzialmente osservabile e non deterministico le **percezioni** sono importanti
 - ▶ restringono gli stati possibili
 - ▶ informano sull'effetto dell'azione
- ▶ Più che un piano l'agente può elaborare una “strategia”, che tiene conto delle diverse eventualità:
un piano con contingenza
- ▶ Esempio: l'aspirapolvere con assunzioni diverse
 - ▶ Vediamo prima il non determinismo

Azioni non deterministiche

L'aspirapolvere imprevedibile

- ▶ Comportamento:
 - ▶ Se aspira in una stanza sporca, la pulisce ... ma talvolta pulisce anche una stanza adiacente
 - ▶ Se aspira in una stanza pulita, a volte rilascia sporco
- ▶ Variazioni necessarie al modello
 - ▶ Il modello di transizione restituisce un **insieme di stati**: l'agente non sa in quale si troverà
 - ▶ Il piano **di contingenza** sarà un piano condizionale e magari con cicli

Esempio



▶ Esempio

▶ Risultati(Aspira, 1) = {5, 7}

▶ Piano possibile

[Aspira,
if stato=5

then [Destra, Aspira]

else []

]

▶ Da sequenza di azioni a piano (albero)

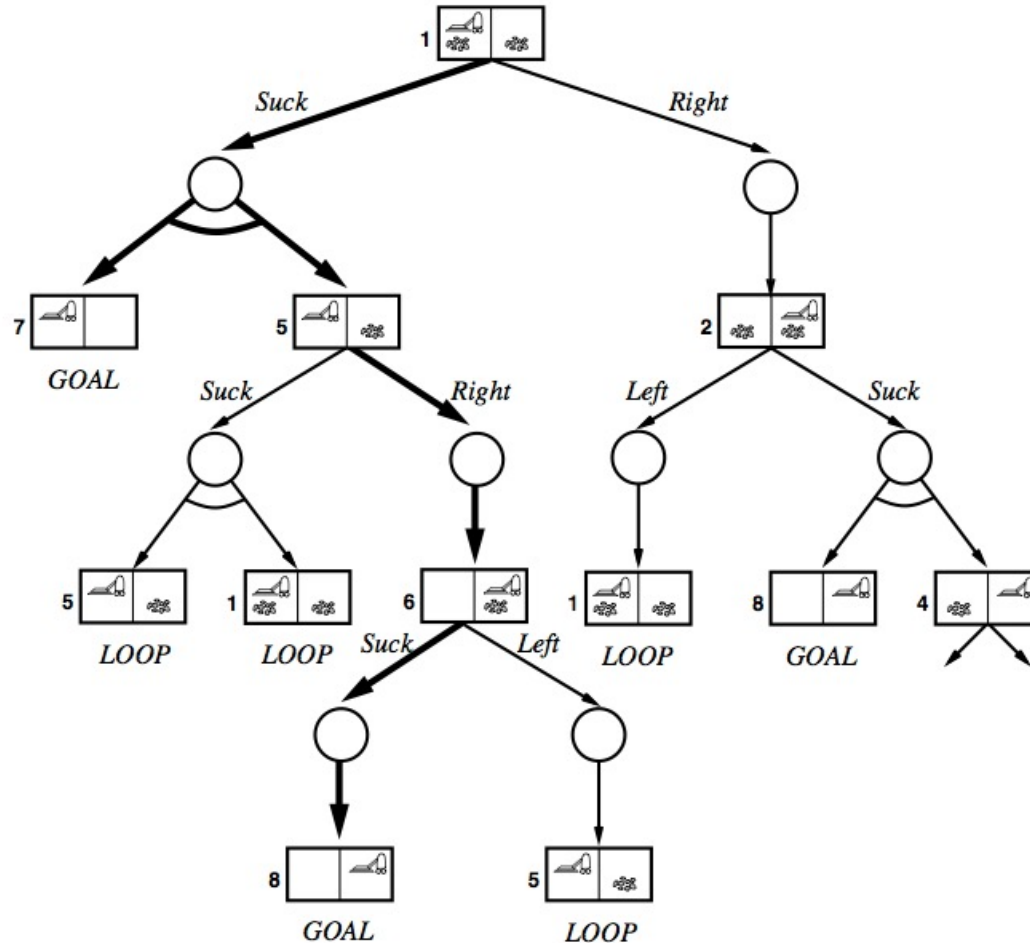
Alberi di ricerca And-Or

- ▶ Fino a questo momento abbiamo discusso strategie di ricerca per grafi OR nei quali vogliamo trovare un singolo cammino verso la soluzione.
- ▶ Il grafo (o albero) AND/OR risulta appropriato quando si vogliono rappresentare problemi che si possono risolvere scomponendoli in un insieme di problemi più piccoli che andranno poi tutti risolti.
- ▶ \Rightarrow arco AND che deve puntare a un qualunque numero di nodi successori che si devono tutti risolvere per risolvere il nodo AND stesso.
- ▶ Dal nodo AND possono anche partire rami OR che indicano soluzioni alternative.

Alberi di ricerca And-Or

- ▶ Nodi OR le scelte dell'agente [**1 sola azione**]
- ▶ Nodi AND le diverse contingenze (le scelte dell'ambiente), da considerare **tutte**
- ▶ Una soluzione a un problema di ricerca AND-OR è un **albero** che:
 - ▶ ha un nodo obiettivo in ogni foglia
 - ▶ specifica un'unica azione nei nodi OR
 - ▶ include tutti gli archi uscenti da nodi AND

Alberi di ricerca And-Or



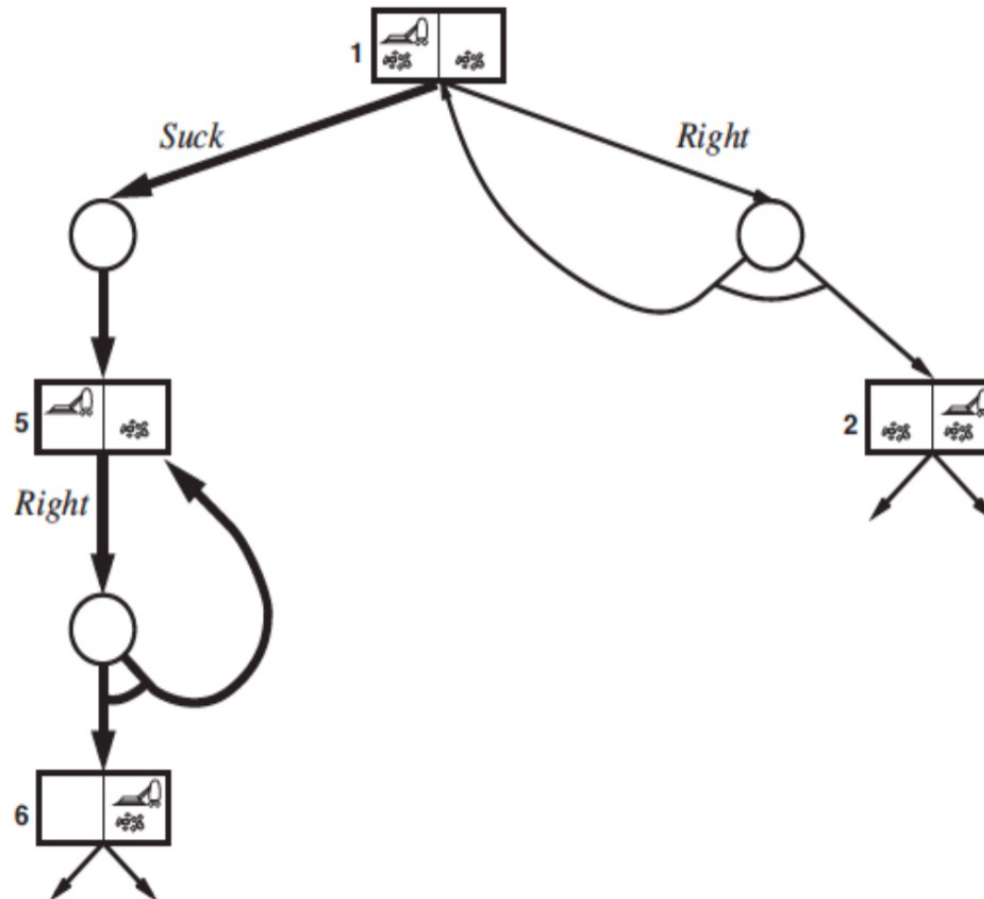
Piano: [Aspira, if Stato=5 then [Destra, Aspira] else []]

Ancora azioni non deterministiche

L'aspirapolvere slittante

- ▶ Comportamento:
 - ▶ Quando si sposta può scivolare e rimanere nella stessa stanza
 - ▶ Es. Risultati(Destra, 1) = {1, 2}
- ▶ Variazioni necessarie
 - ▶ Continuare a provare ... [finche' riesce ad andare a destra]
 - ▶ Il **piano di contingenza** potrà avere dei cicli

Aspirapolvere slittante: soluzione



Piano: [Aspira, L_1 : Destra, if Stato=5 then L_1 else Aspira]

Ricerca con assenza di osservazioni o con osservazioni parziali

- ▶ Le percezioni non sono sufficienti a determinare lo stato esatto, anche se l'ambiente è deterministico.
- ▶ **Stato credenza**: un insieme di stati possibili in base alle conoscenze dell'agente
- ▶ **Problemi senza sensori** (*sensorless o conformanti*)
- ▶ Si possono trovare soluzioni anche senza affidarsi ai sensori utilizzando stati-credenza

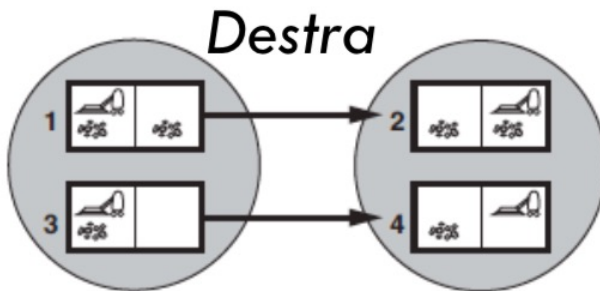
Ambiente non osservabile: *Aspirapolvere senza sensori*

- ▶ L'aspirapolvere:
 - ▶ non percepisce la sua locazione, né se la stanza è sporca o pulita
 - ▶ conosce la geografia del suo mondo e l'effetto delle azioni
- ▶ Inizialmente tutti gli stati sono possibili
 - ▶ Stato iniziale = {1, 2, 3, 4, 5, 6, 7, 8}
- ▶ Le azioni riducono gli stati credenza
- ▶ Nota: nello spazio degli stati credenza l'ambiente è **osservabile** (l'agente conosce le sue credenze)

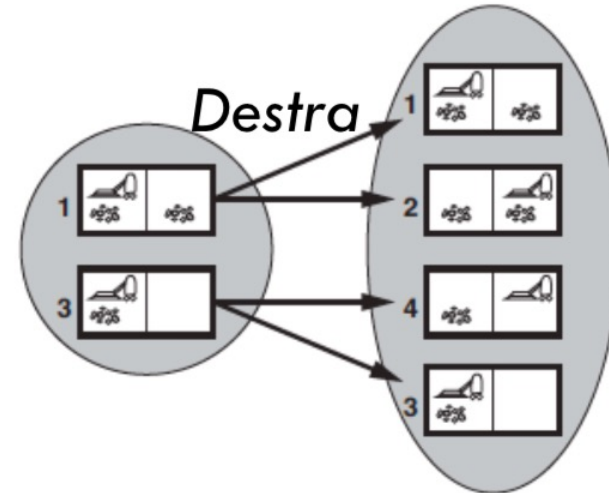
Formulazione di problemi con stati-credenza

- ▶ Se N numero stati, 2^N sono i possibili stati credenza
- ▶ **Stato-credenza iniziale** $SC_0 \subseteq$ insieme di tutti gli N stati
- ▶ **Azioni**(b) = unione delle azioni **lecite** negli stati in b (ma se azioni illecite in uno stato hanno effetti dannosi meglio intersezione)
- ▶ **Modello di transizione**: gli stati risultanti sono quelli ottenibili applicando le azioni a uno stato qualsiasi (l'unione degli stati ottenibili dai diversi stati possibili con le azioni eseguibili)

Problemi con stati-credenza (cnt.)



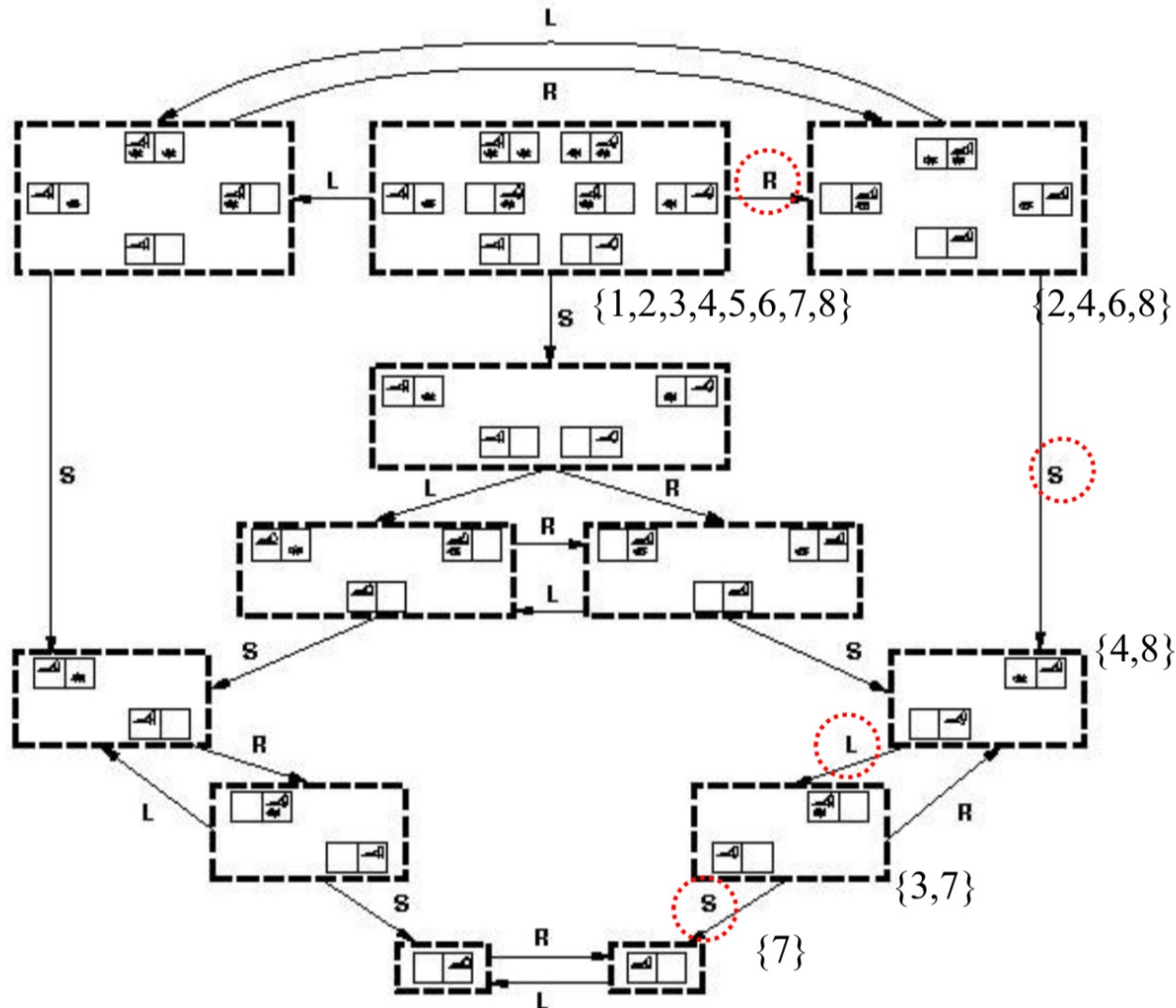
Senza sensori deterministico



Senza sensori e slittante (non det.)

- ▶ **Test obiettivo**: tutti gli stati nello stato credenza devono soddisfarlo
- ▶ **Costo di cammino**: il costo di eseguire un'azione potrebbe dipendere dallo stato, ma assumiamo di no

Il mondo dell'aspirapolvere senza sensori (determ.)



Ricerca della soluzione

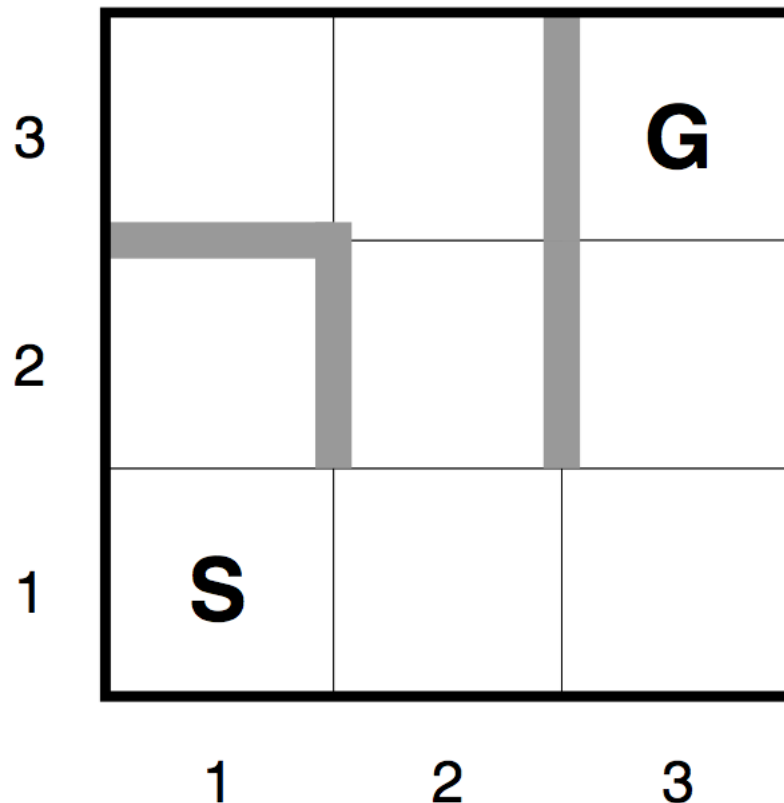
- ▶ Gli stati credenza possibili sono $2^8=256$ ma solo 12 sono raggiungibili
- ▶ In generale lo spazio di ogni stato può essere molto più grande con gli “stati credenza”
- ▶ La rappresentazione atomica obbliga a elencare tutti gli stati. Non è molto “compatta”.

Ricerca online

- ▶ Ricerca offline e ricerca online
- ▶ L'agente alterna pianificazione e azione
- 1. Utile in ambienti dinamici o semi-dinamici
 - ▶ Non c'è troppo tempo per pianificare
- 2. Utile in ambienti non deterministici
 - ▶ Pianificare vs agire
- 3. Necessaria per **ambienti ignoti** tipici dei **problemi di esplorazione**

Esempio: Labirinto con mappa e senza

- ▶ Con mappa
 - ▶ applicabili tutti gli algoritmi di ricerca visti
- ▶ Senza mappa
 - ▶ l'agente non può pianificare, può solo esplorare nel modo più razionale possibile
 - ▶ Ricerca online



Assunzioni

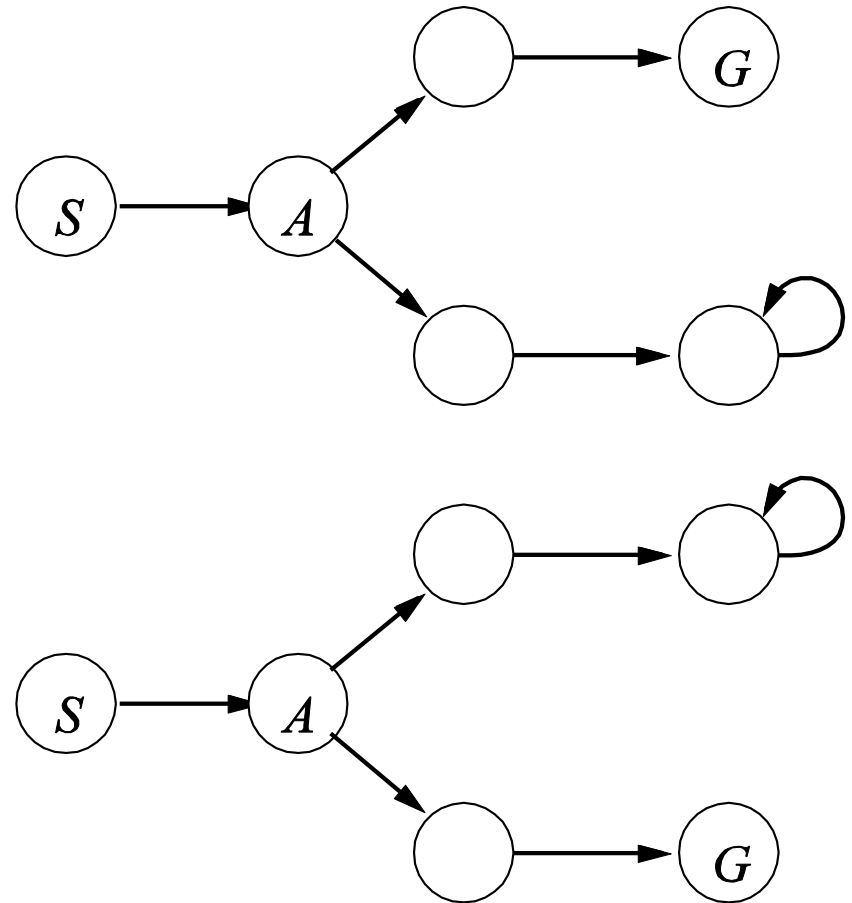
- ▶ Un problema di ricerca online può essere risolto da un agente che esegue azioni
- ▶ Conoscenza:
 - ▶ $AZIONI(s)$: lista delle azioni permesse nello stato s
 - ▶ $c(s,a,s')$: funzione di costo che può essere usata solo se s' è il risultato dell'azione
 - ▶ $TEST-OBIETTIVO(s)$
- ▶ $RISULTATO(s,a)$ non è noto
- ▶ L'agente potrebbe avere accesso ad una funzione euristica ammissibile $h(s)$

Costo della soluzione

- ▶ Il costo del cammino è quello effettivamente percorso
- ▶ Il rapporto tra questo costo e quello ideale (conoscendo l'ambiente) è chiamato **rapporto di competitività**
- ▶ Tale rapporto può essere infinito
- ▶ Le prestazioni sono in funzione dello spazio degli stati

Assunzioni

- ▶ Ambienti esplorabili in maniera sicura
 - ▶ E' sempre possibile arrivare ad uno stato obiettivo
 - ▶ Non esistono azioni irreversibili
- ▶ Nessun algoritmo può evitare vicoli ciechi in tutti gli spazi degli stati



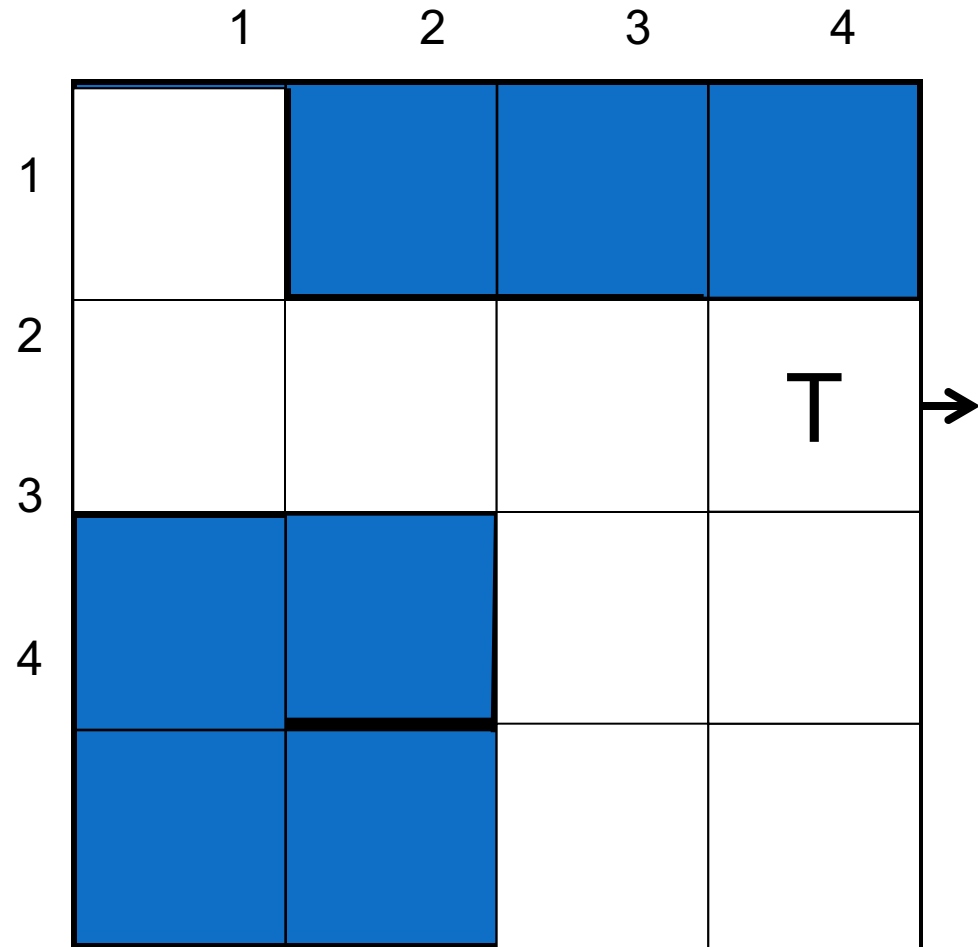
Ricerca in profondità *online*

- ▶ Gli agenti *online* ad ogni passo decidono l'azione da fare (non il piano) e la eseguono. Non possono espandere più stati. Costruiscono mappa dell'ambiente.
- ▶ Ricerca in profondità *online*
 - ▶ Un metodo locale
 - ▶ Il backtracking significa tornare sui propri passi
 - ▶ È necessario ricordarsi ciò che si è scoperto
 - ▶ Esplorazione sistematica delle alternative
- ▶ Funziona negli spazi degli stati dove le azioni sono reversibili

Esempio

- ▶ Sceglie il primo tra $(1,1)$ e $(2,1)$
- ▶ In $(1,1)$ deve spostarsi in $(2,1)$

	1	2	3	4
1				
2				T
3				
4				

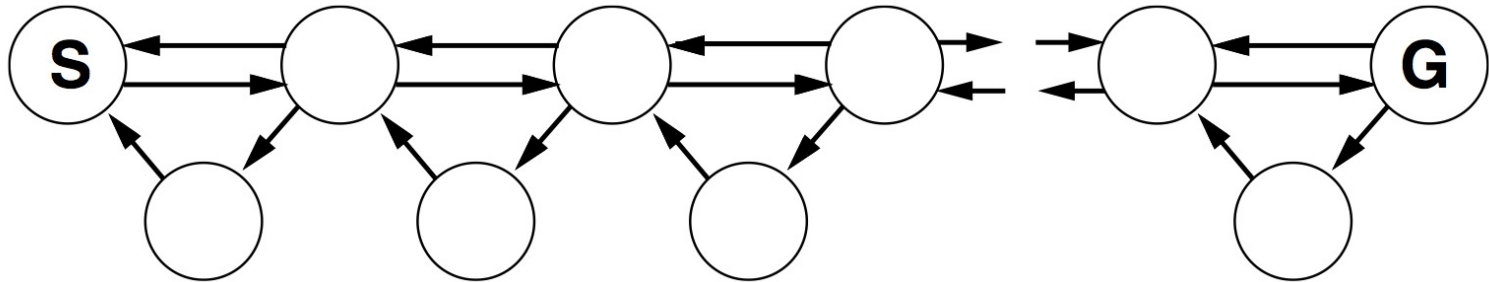


Ricerca locale *online*

- ▶ La ricerca Hill Climbing è già un algoritmo online
- ▶ Sfortunatamente si potrebbe bloccare in un massimo/minimo locale.
- ▶ Random-restart non praticabile
- ▶ Come sfuggire ai massimi/minimi locali?

Due soluzioni

- ▶ Random-walk



- ▶ Apprendimento Real-Time: esplorando si aggiustano i valori dell'euristica per renderli più realistici
 - ▶ Se s è uno stato con f. euristica $h(s)$, i successori sono valutati $h(s)$ se inesplorati, altrimenti si prende la loro stima H + costo azione per raggiungerli.
 - ▶ $\text{Costo-LRTA}^*(s, a, s', H) = \begin{cases} h(s) & \text{se } s' \text{ non esplorato} \\ H(s') + \text{costo}(s, a, s') & \text{altrimenti} \end{cases}$