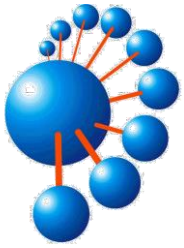
Búsqueda en línea – Ambientes desconocidos

Julio Godoy

DIICC



Hasta el momento..

- Agentes usan algoritmos de búsqueda 'offline' (Anchura, profundidad, A*, etc..):
  - Calculan una **solución completa** antes de actuar
  - **Sólo** cuando se encuentra una solución, se actúa

Hasta el momento..

- Agentes usan algoritmos de búsqueda LOCAL:
  - Aún tienen acceso a espacio de estados completo
  - Pero sólo almacenan el estado actual y sus sucesores
  - qué algoritmo usar? → decisión de diseño de la solución

Ahora..

- Búsqueda en línea:
  - Agente no conoce todo el espacio de estados
    - Restricción del problema!
      - Sólo conoce su estado actual y las posibles acciones.
  - Intercala cómputo y acción:
    - Efectúa una acción
    - Observa el ambiente
    - Calcula la siguiente acción
    - Repite

- Búsqueda en línea es buena idea cuando:
  - Ambiente es dinámico
  - Ambiente es No-determinista

- Búsqueda en línea es NECESARIA cuando:
  - Ambiente es desconocido:
    - Cúales son los estados?
    - Qué hacen mis acciones?
- Agente enfrenta un **problema de exploración**

Ejemplos

- Robót móvil
  - Exploración, encontrar salida
- Recién nacido

Problemas de búsqueda en línea

- Por ahora, asumimos un ambiente determinista y observable

- Agentes saben:
  - Acciones (en cada estado)
  - Function de costo c(s, a, s')
  - Test objetivo
- Agentes no saben:
  - Efecto de sus acciones
  - Espacio de estados

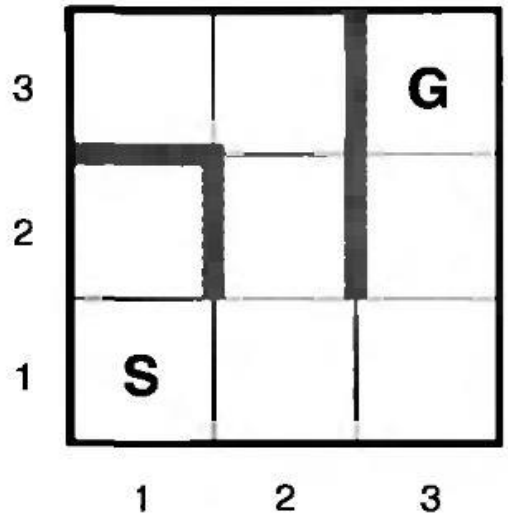


Figure 4.18 A simple maze problem. The agent starts at S and must reach G, but knows nothing of the environment.

Imagen: Russell y Norvig, 2010

Objetivo

- Típicamente:
  - Llegar al objetivo minimizando el costo
  - También puede ser: explorar el ambiente
- Desempeño: porción del espacio de estados explorado vs espacio de estados completo

Problemas

- Caminos sin salida:
  - Si las acciones no son reversibles y se han probado todas las acciones

- Por ahora, suponemos un **espacio de estados seguro de explorar**
  - Ejemplo: puzzle-8, laberinto

Agentes para búsqueda en línea

- Compara offline vs en línea
  - Piensa en A* o búsqueda uniforme
  - ¿Podemos expandir un nodo/estado en **cualquier** ubicación?
  - Ubicación física
  - ¿Podemos adaptar un algoritmo de búsqueda conocido para que funcione en línea?

Agentes para búsqueda en línea

- Compara offline vs en línea
  - Piensa en A* o búsqueda uniforme
  - ¿Podemos expandir un nodo/estado en **cualquier** ubicación?
  - Ubicación física
  - **Búsqueda en profundidad** en línea: sólo con acciones reversibles

```
function ONLINE-DFS-AGENT(s') returns an action
  inputs: s', a percept that identifies the current state
  static: result, a table, indexed by action and state, initially empty
          unexplored, a table that lists, for each visited state, the actions not yet tried
          unbacktracked, a table that lists, for each visited state, the backtracks not yet tried
          s, a, the previous state and action, initially null

  if GOAL-TEST(s') then return stop
  if s' is a new state then unexplored[s'] ← ACTIONS(s')
  if s is not null then do
      result[a, s] ← s'
      add s to the front of unbacktracked[s']
  if unexplored[s'] is empty then
      if unbacktracked[s'] is empty then return stop
      else a ← an action b such that result[b, s'] = POP(unbacktracked[s'])
  else a ← POP(unexplored[s'])
  s ← s'
  return a
```

Figure 4.20 An online search agent that uses depth-first exploration. The agent is applicable only in bidirected search spaces.

Imagen: Russell y Norvig, 2010

Búsqueda en línea y local

- Recuerda los algoritmos discutidos la clase pasada:
  - ¿Aplicables a búsqueda en línea?

Búsqueda en línea y local

- Recuerda los algoritmos discutidos la clase pasada:
  - Aplicables a búsqueda en línea?
  - Búsqueda Hill climbing

Búsqueda en línea y local

- Recuerda los algoritmos discutidos la clase pasada:
  - Aplicables a búsqueda en línea?
  - Búsqueda Hill climbing: óptimos locales…

Búsqueda en línea y local

- Recuerda los algoritmos discutidos la clase pasada:
  - Aplicables a búsqueda en línea?
  - Búsqueda Hill climbing: óptimos locales…
  - Camino aleatorio!

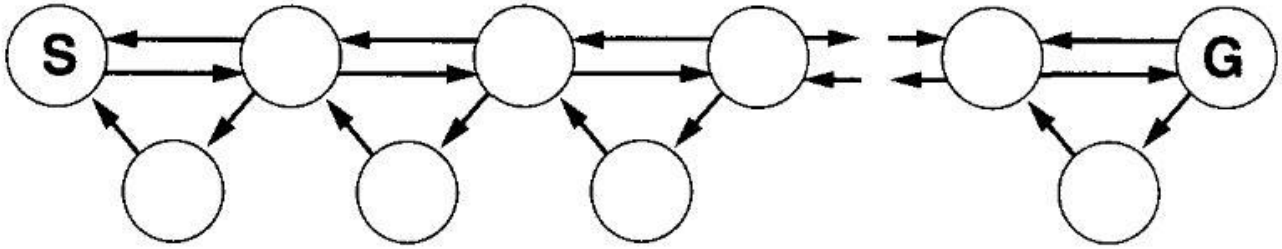


Figure 4.21 An environment in which a random walk will take exponentially many steps to find the goal.

Imagen: Russell y Norvig, 2010

Búsqueda en línea y local

- Recuerda los algoritmos discutidos la clase pasada:
  - Aplicables a búsqueda en línea?
  - Búsqueda Hill climbing: óptimos locales…
  - Camino aleatorio! → muy ineficiente
  - El agente necesita memoria

LRTA*

- Learning Real Time A*
  - Función de costo c(s, a, s')
  - ‘la mejor estimación actual’ H(s)

- Crea un 'mapa' basado en las acciones tomadas
- Asume que acciones no tomadas llevan al objetivo

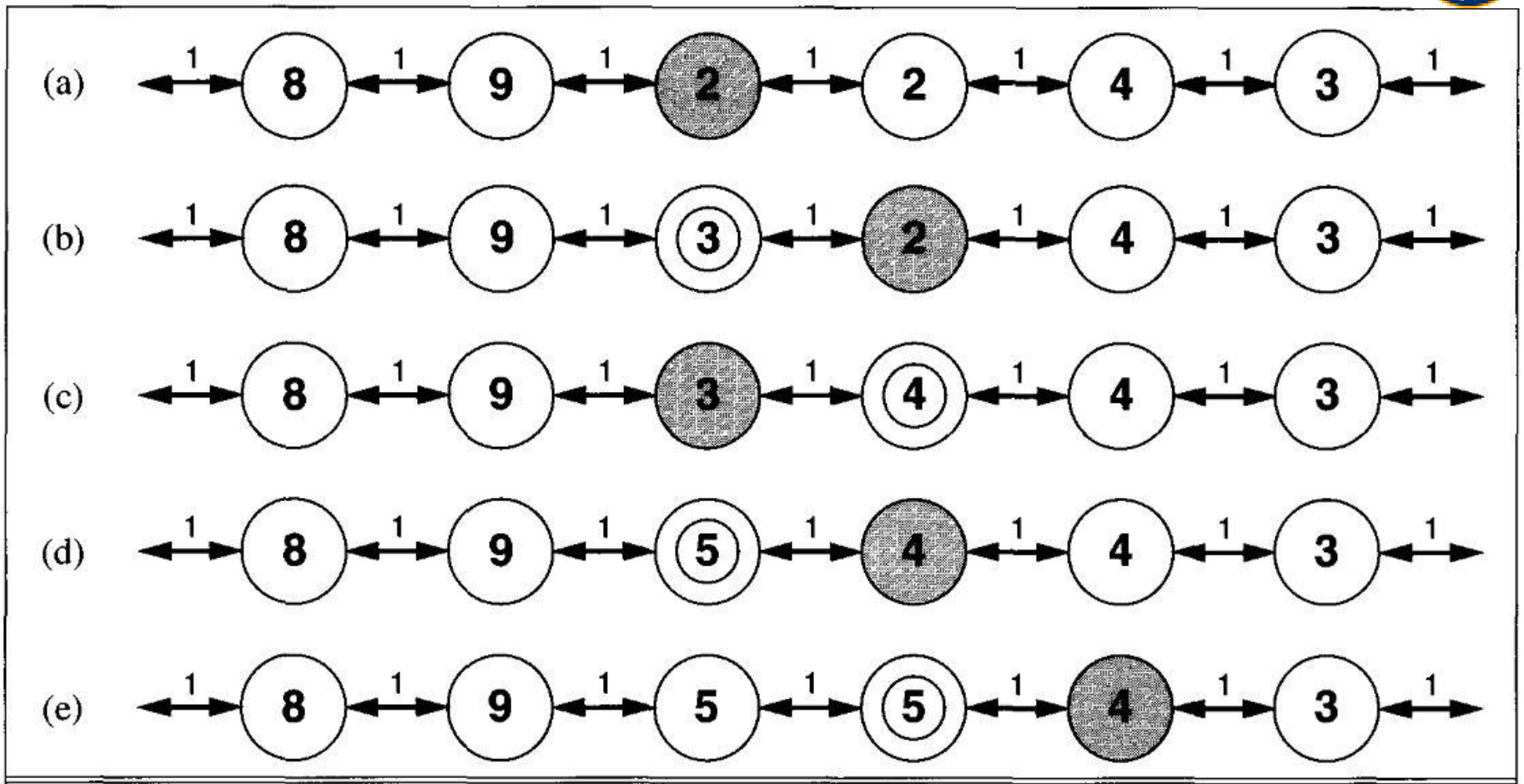


Figure 4.22 Five iterations of LRTA* on a one-dimensional state space. Each state is labeled with $H(s)$, the current cost estimate to reach a goal, and each arc is labeled with its step cost. The shaded state marks the location of the agent, and the updated values at each iteration are circled.

Imagen: Russell y Norvig, 2010

```
function LRTA*-AGENT(s') returns an action
  inputs: s', a percept that identifies the current state
  static: result, a table, indexed by action and state, initially empty
          H, a table of cost estimates indexed by state, initially empty
          s, a, the previous state and action, initially null

  if GOAL-TEST(s') then return stop
  if s' is a new state (not in H) then H[s'] ← h(s')
  unless s is null
      result[a, s] ← s'
      H[s] ← min_{b ∈ ACTIONS(s)} LRTA*-COST(s, b, result[b, s], H)
  a ← an action b in ACTIONS(s') that minimizes LRTA*-COST(s', b, result[b, s'], H)
  s ← s'
  return a

function LRTA*-COST(s, a, s', H) returns a cost estimate
  if s' is undefined then return h(s)
  else return c(s, a, s') + H[s']
```

Figure 4.23 LRTA*-AGENT selects an action according to the values of neighboring states, which are updated as the agent moves about the state space.

Imagen: Russell y Norvig, 2010

LRTA*

- Garantizado a encontrar el objetivo en ambientes:
  - Finitos
  - Seguro de explorar

Aprendizaje en búsqueda en línea

- Agente aprende un 'mapa' del ambiente
  - (s,a,?)

- Agente adquiere estimaciones más precisas del costo de un estado

Aprendizaje en búsqueda en línea

- A
  –
- A s
  d

Reinforcement Learning!