
Inteligencia Artificial

Búsqueda Informada y Exploración II


Jorge Luis Guevara Diaz

www.jorge.sistemasyservidores.com

Que veremos?

- **Busqueda Primero el mejor**
 - Busqueda Greedy primero el mejor
 - Busqueda A*
 - **Búsqueda Heurística con memoria limitada**
 - IDA*
 - Busqueda recursiva primero el mejor RBFS
 - SMA*
 - **Heurísticas**
 - **Búsqueda Local y Optimización**
 - **Busqueda Local en espacios Continuos**
-

Que veremos?

- **Búsqueda Local y Optimización**
 - Hill Climbing
 - Simulated Annealing (Tarea)
 - Local Beam
 - Algoritmos Genéticos
 - **Busqueda Local en espacios Continuos**
 - **Búsqueda Online en Entornos desconocidos**
 - Problemas de búsqueda online
 - Agentes de búsqueda online
 - Búsqueda local online
 - Aprendizaje en búsqueda local online
- 
-

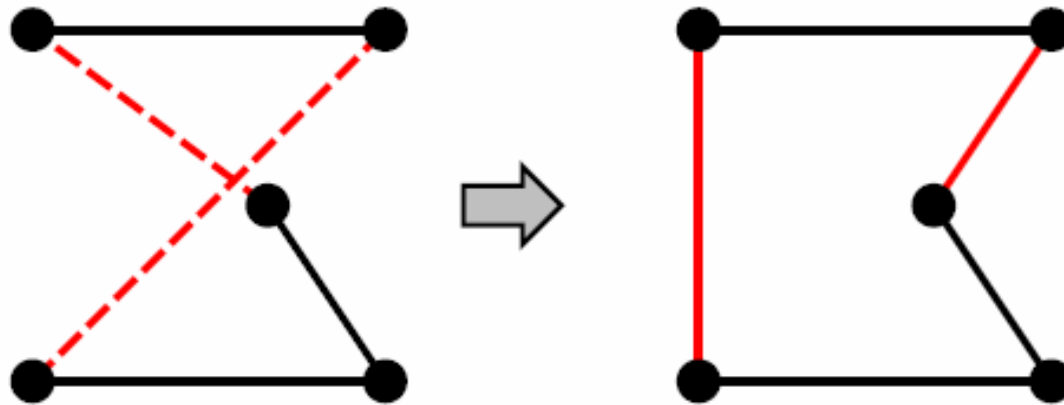
Búsqueda Local y Optimización

- En muchos problemas de optimización, el **camino** a la meta es irrelevante; el estado meta en si mismo es la solución
 - Estado de espacios = conjunto de configuraciones "completas"
Ejemplo buscar **configuración óptima** del TSP, n-reynas,etc
 - En estos casos se utilizan algoritmos de **mejora iterativa**, es decir guardar el estado actual y tratar de mejorarlo
 - Los algoritmos de búsqueda local son útiles tanto para búsqueda offline y búsqueda online
-

Búsqueda Local y Optimización

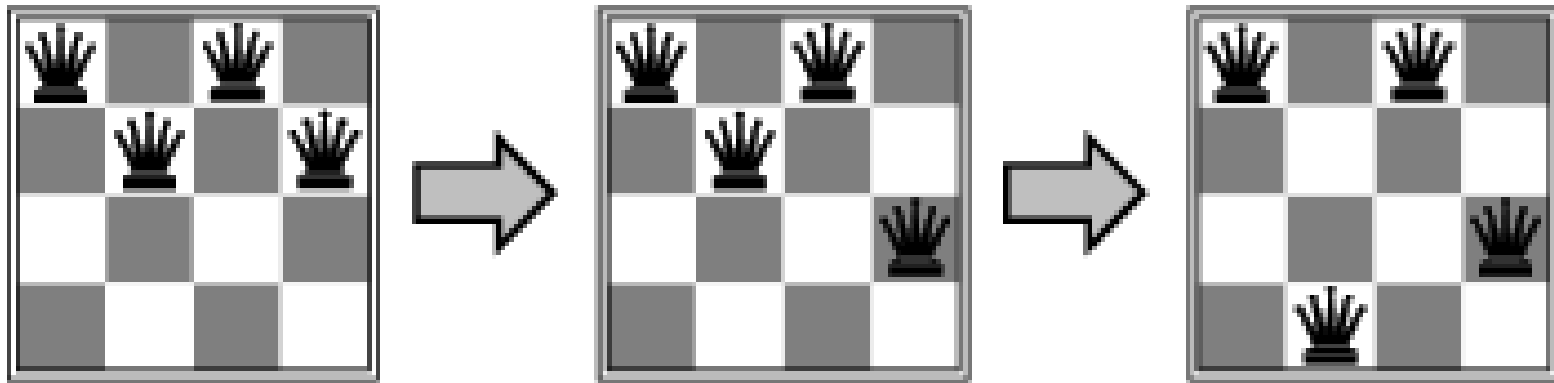
- Ejemplo TSP

Empezar con cualquier tour completo y realizar intercambios



Búsqueda Local y Optimización Ejemplo: *n*-reynas

- Colocar *n* reynas en un tablero *n* × *n* sin dos reynas en la misma fila, columna, o diagonal



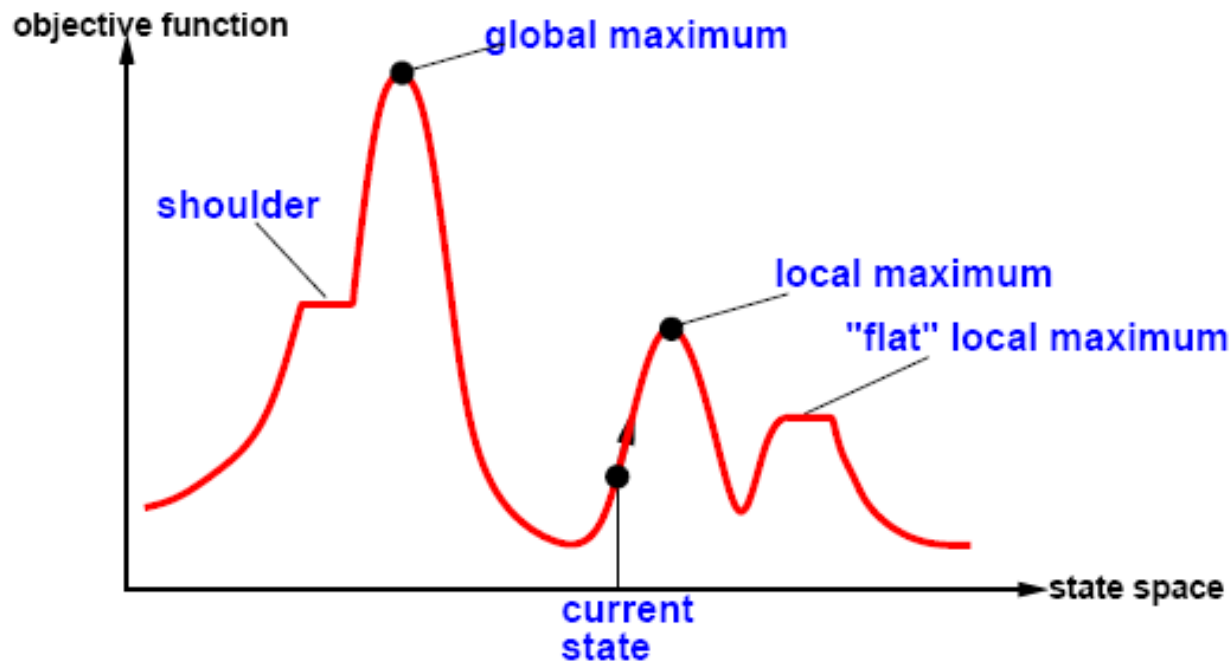
Búsqueda Local y Optimización :

Búsqueda Hill-climbing









- “Como escalar el Everest en espesa niebla con amnesia“
 - “Es un loop que continuamente se mueve en dirección de valor más creciente”
 - Termina cuando un pico es encontrado.
 - Hill climbing no ve mas allá que sus vecinos inmediatos de su estado actual.
 - Hill-climbing escoge aleatoriamente entre el conjunto de mejores sucesores, si es que hay mas de uno.
 - Hill-climbing es una búsqueda local *greedy*.
-

Búsqueda Local y Optimización : Búsqueda Hill-climbing

- Problema: dependiendo del estado inicial, podemos obtener un máximo local.

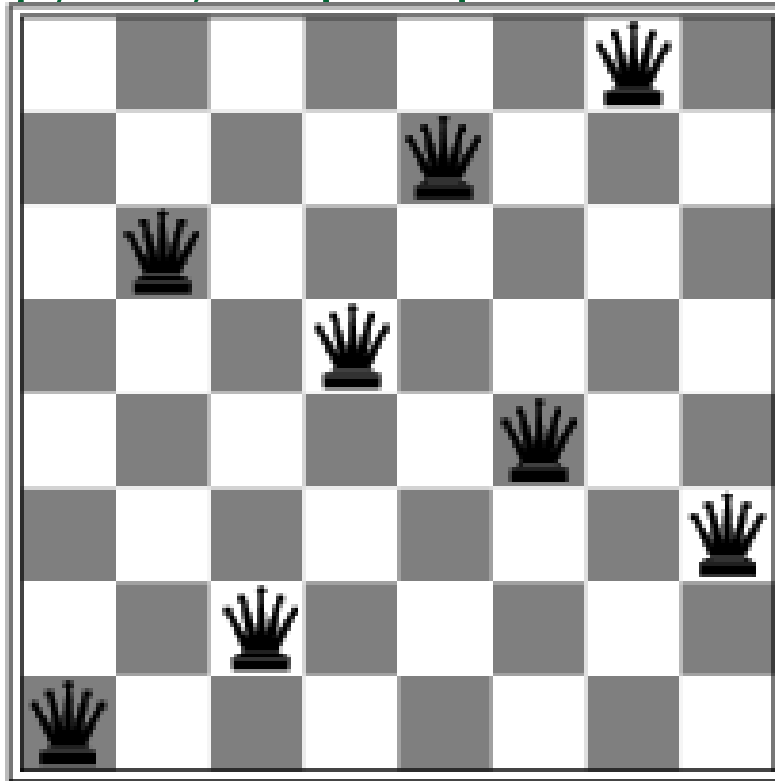


Búsqueda Local y Optimización : Búsqueda Hill-climbing : Ejemplo problema 8-reynas

18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14		13	16	13	16
	14	17	15		14	16	16
17		16	18	15		15	
18	14		15	15	14		16
14	14	13	17	12	14	12	18

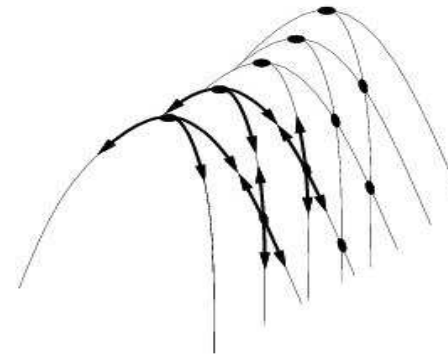
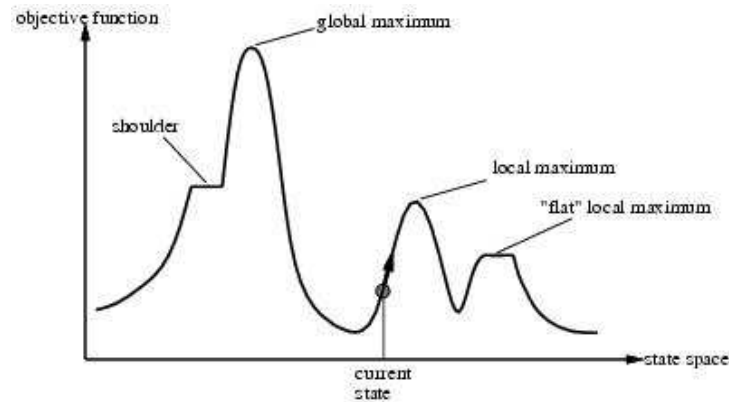
- h = numero de pares de reynas que se están atacando una a otra, directa o indirectamente
- $h = 17$ par el estado actual

Búsqueda Local y Optimización : Búsqueda Hill-climbing : Ejemplo problema 8-reynas



- Un mínimo local con $h = 1$ □
-

Búsqueda Local y Optimización : Búsqueda Hill-climbing



Algunos Inconvenientes

- **Máximos Locales** = es un pico que es mas alto que sus vecinos pero más bajo que el máximo local
- **Colinas** = secuencia de máximos locales que son difíciles de navegar para los algoritmos greedy
- **Mesetas** = una area en el espacio de estados donde la función de evaluación es una superficie llana.

Búsqueda Local y Optimización : Búsqueda Hill-climbing

Se atasca el 86% del tiempo, resolviendo solamente el 14% de instancias de problemas para estados generados aleatoriamente de 8-reynas

- La buena noticia es que en 4 pasos como promedio en encontrar soluciones, y 3 pasos en atascarse, no es una mala noticia para un espacio de estados de 8^8 que es igual a aproximadamente 17 millones de estados
-

Búsqueda Local y Optimización : Búsqueda Simulated annealing (Investigación)

- Escapar de los máximos locales permitiendo algunos “malos” movimientos, pero gradualmente decrementado su frecuencia
 - Se puede demostrar lo siguiente: Si T decremента lo suficientemente lento, entonces la búsqueda simulated annealing podrá encontrar un óptimo local con una aproximación de probabilidad de 1
 - Usando ampliamente en diseño VLSI layout, planificación de aerolíneas, etc□
-

Búsqueda Local y Optimización :

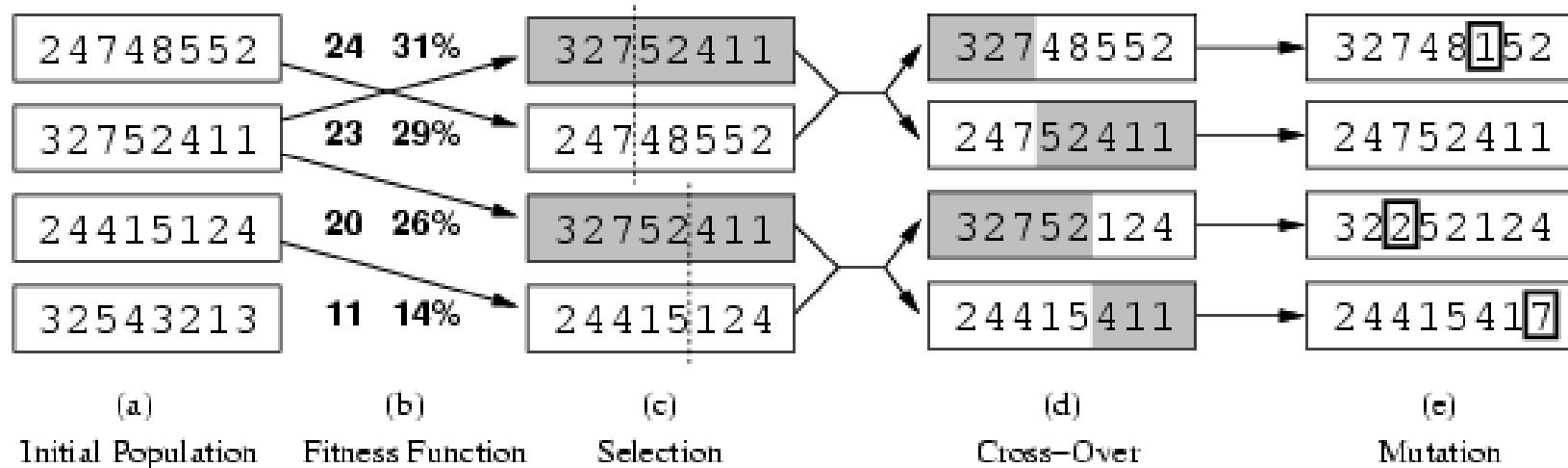
Búsqueda Local Beam

- Idea: Guardar k estados en lugar de 1, luego escoger los mejores k de sus descendientes
 - Empezar con k estados generados aleatoriamente
 - En cada iteración todos los sucesores de cada estado son generados
 - Si alguno es el estado meta parar, si no seleccionar los mejores k sucesores de la lista completa de estados generados y repetir
-

Búsqueda Local y Optimización : Algoritmos Genéticos

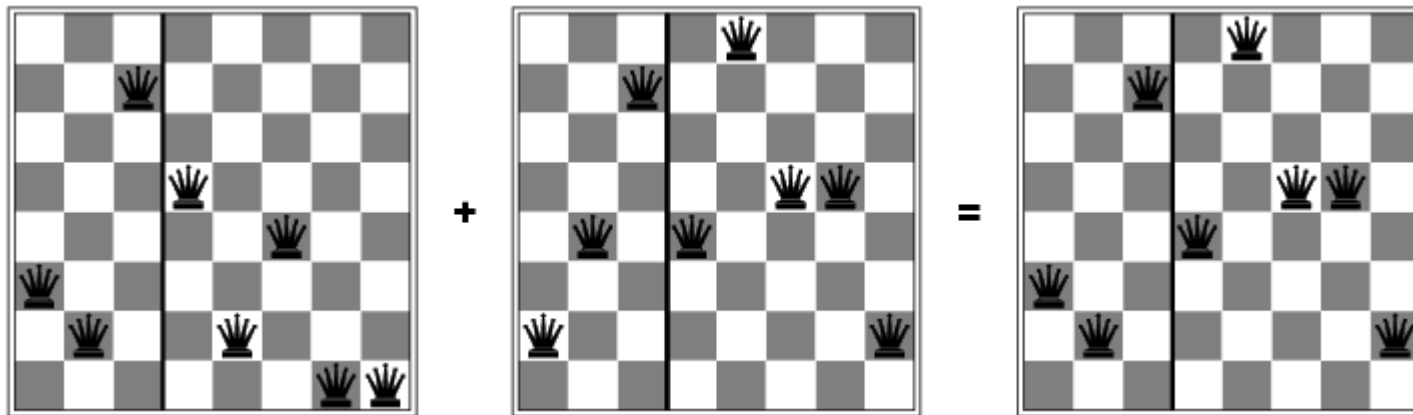
- Un estado sucesor es generado por combinar dos estados padres
 - Empezar con k estados generados aleatoriamente (**población**)
 - Un estado es representado como una cadena sobre un alfabeto finito (usualmente una cadena de 0s y 1s) □
 - Función de evaluación (**fitness**). Altos valores para mejores estados
 - Producir la siguiente generación de estados por selección, cruzamiento, y mutación
-

Búsqueda Local y Optimización : Algoritmos Genéticos



- Fitness : número de pares de reinas que no se atacan (min = 0, max = $8 \times 7/2 = 28$) □
- $24/(24+23+20+11) = 31\%$ □
- $23/(24+23+20+11) = 29\%$ etc □

Búsqueda Local y Optimización : Algoritmos Genéticos



Los estados de las 8-reynas que corresponden a los primeros dos padres de la diapositiva anterior (c). Las columnas sombreadas son perdidas en el paso de cruzamiento, y las columnas iluminadas son retenidas

Que veremos?

- **Búsqueda Local y Optimización**

- Hill Climbing
- Simulated Annealing (Tarea)
- Local Beam
- Algoritmos Genéticos

- **Busqueda Local en espacios Continuos**

- **Búsqueda Online en Entornos desconocidos**

- Problemas de búsqueda online
 - Agentes de búsqueda online
 - Búsqueda local online
 - Aprendizaje en búsqueda local online
-



Busqueda Local en Espacios Continuos

- La mayoría de entornos en el mundo real son continuos
 - La funcion sucesor retornará infinitos estados
 - Métodos que usan la informacion del gradiente son utilizados
-

Busqueda Local en Espacios Continuos

- Suponga que deseamos ubicar tres aeropuertos en Rumania
 - Espacio de 6 estados definidos como $(x_1, y_1), (x_2, y_2), (x_3, y_3)$
 - Función objetivo, definida como $f(x_1, y_1, x_2, y_2, x_3, y_3) =$ suma del cuadrado de las distancias de cada ciudad al aeropuerto mas cercano

Métodos de **Discretización** convierten espacios continuos a espacios discretos, ejemplo, gradiente empírica considera un valor de cambio discreto $\pm\delta$ en las coordenadas

Busqueda Local en Espacios Continuos

- Métodos que utilizan gradiente realizan el siguiente cálculo para incrementar o reducir la función f

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3} \right)$$

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f(\mathbf{x})$$

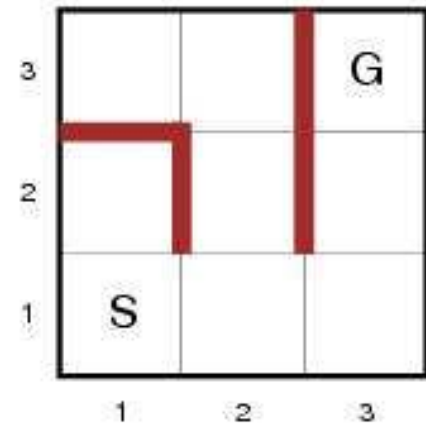
- Algunas veces se puede solucionar $\nabla f(\mathbf{x}) = 0$ exactamente
 - Newton{Raphson (1664, 1690) utilizan $\mathbf{x} \leftarrow \mathbf{x} - \mathbf{H}_f^{-1}(\mathbf{x}) \nabla f(\mathbf{x})$ para solucionar $\nabla f(\mathbf{x}) = 0$, donde $\mathbf{H}_{ij} = \partial^2 f / \partial x_i \partial x_j$
-

Búsqueda Online en Entornos Desconocidos

- Hasta ahora nuestros algoritmos fueron **offline**.
 - **Offline**= la solución es determinada antes de ejecutarla.
 - **Online** = intercambia computación y acción
 - Búsqueda **Online** es necesaria para entornos dinámicos y semi-dinámicos
 - Es imposible tomar en cuenta todas las posibles contingencias.
 - Usados para *problemas de exploración*:
 - Estados y acciones desconocidas.
 - ejemplo. Cualquier robot en un entorno nuevo, un bebé recién nacido,...
-

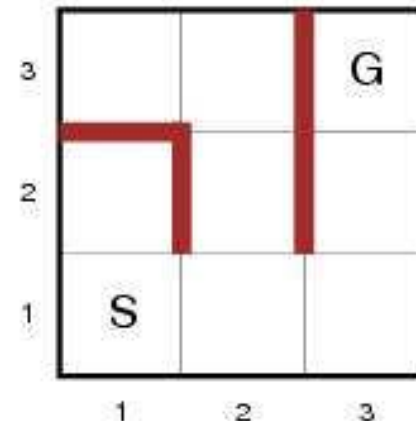
Búsqueda Online en Entornos Desconocidos : Problemas de Búsqueda Online

- Un problema de búsqueda online puede ser resuelto únicamente por la ejecución de acciones de un agente en vez de un proceso computacional puro, se asume que el agente tiene conocimiento de lo siguiente:
 - **ACCIONES(s)**: lista de acciones permitidas en el estado s
 - **$C(s,a,s')$** : función de costo de paso (! Después de que s' es determinado)
 - **TEST-META(s)**
- Un agente puede reconocer estados previos.
- Las acciones son determinísticas.
- Tienen acceso a una **heurística admisible $h(s)$**
ejemplo. distancia de manhattan

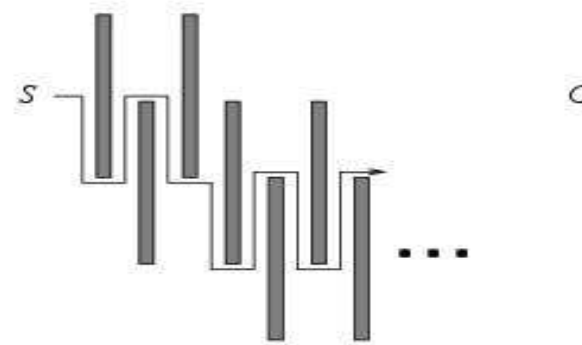
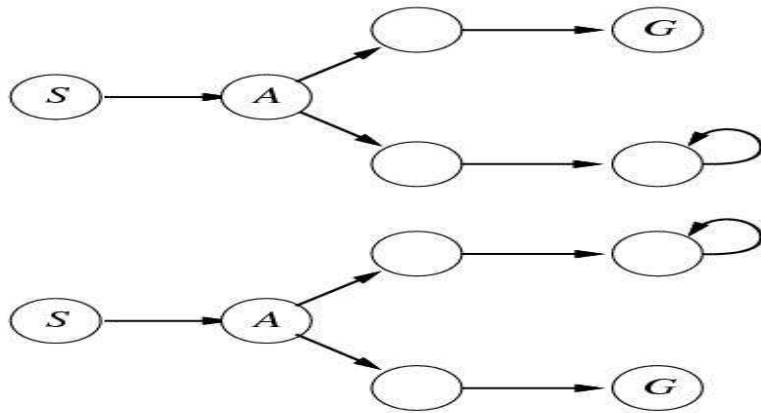


Búsqueda Online en Entornos Desconocidos : Problemas de Búsqueda Online

- Objetivo: alcanzar la meta con costo mínimo
 - Costo = costo total del camino recorrido
 - Radio competitivo=comparación del costo con el costo del camino de la solución si el espacio de búsqueda es conocido.
 - Puede ser infinito en el caso del que el agente accidentalmente caiga en situaciones que no tienen salida



Búsqueda Online en Entornos Desconocidos : Problemas de Búsqueda Online



- **El argumento del adversario**
- Asumir un adversario quien puede construir el espacio de estados mientras el agente lo explora
 - Estados visitados S y A. Que sigue?
 - Falla en cualquiera de los espacios de estados
- No existe algoritmo que pueda evitar situaciones que no tengan salida en todos los espacios de estados.

Búsqueda Online en Entornos Desconocidos : Agentes de Búsqueda Online

- El agente mantiene un **mapa del entorno**.
 - Actualizado basado en la entrada de percepciones.
 - Este mapa es usado para decidir la siguiente acción.
-

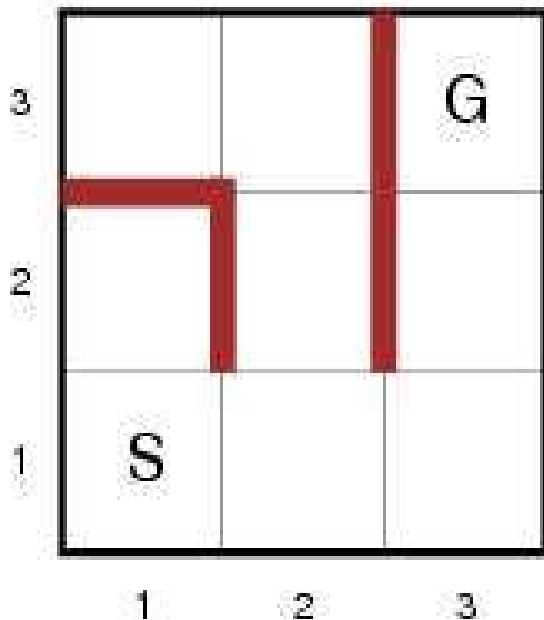
Búsqueda Online en Entornos Desconocidos : Agentes de Búsqueda Online

```
function AGENTE_DFS_ONLINE (s') return una acción
  input: s', una percepción identificando el estado actual
  static: resultado, una tabla indexada por la acción y el estado, inicialmente vacía
           no_explorada, una tabla que lista para cada estado visitado, la acción todavía
           no probada
           no_vuelto_atrás, una tabla que lista para cada estado visitado, la vuelta atrás
           no probada
           s,a, el estado y la acción previa, inicialmente nula

  if TEST-META(s') then return parar
  if s' es un nuevo estado then no_explorada[s'] ← ACCIONES(s')
  if s no es nulo then do
    resultado[a,s] ← s'
    agregar s al inicio de no_vuelto_atrás[s']
  if no_explorada[s'] es vacía then
    if no_vuelto_atrás[s'] es vacía then return parar
    else a ← una acción b tal que resultado[b, s']=POP(no_vuelto_atrás[s'])
  else a ← POP(no_explorada[s'])
  s ← s'
  return a
```

Búsqueda Online en Entornos Desconocidos :

Agentes de Búsqueda Online, Ejemplo



Resultado = tabla indexada por la acción y el estado

no_explorado=tabla que lista para cada estado visitado las acciones que todavía no han sido probadas

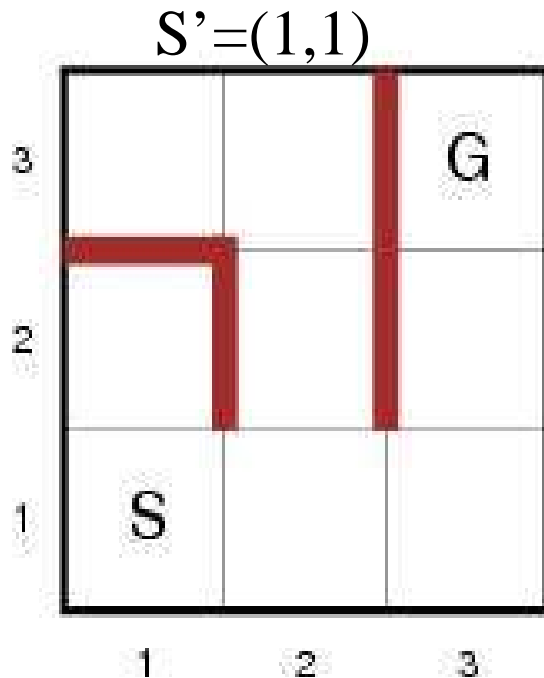
no_vuelto_atras=tabla que lista para cada estado visitado la vuelta atrás que no ha sido probada

S, a = el estado previo y la acción

s' = una percepción que identifica el estado actual

- Asumir el problema del laberinto en una malla de 3x3.
- $s' = (1,1)$ es el estado inicial
- Resultado, no_explorado (UX), no_vuelto_atras (UB), ...
están vacíos
- S,a también están vacíos

Búsqueda Online en Entornos Desconocidos : Agentes de Búsqueda Online, Ejemplo



TEST-META((1,1))?

□ $S \neq G$ falso

(1,1) un nuevo estado?

□ Verdadero

□ ACCION((1,1)) \rightarrow UX[(1,1)]

■ {DERECHA,ARRIBA}

s es nulo?

□ Verdadero (inicialmente)

UX[(1,1)] vacio?

□ Falso

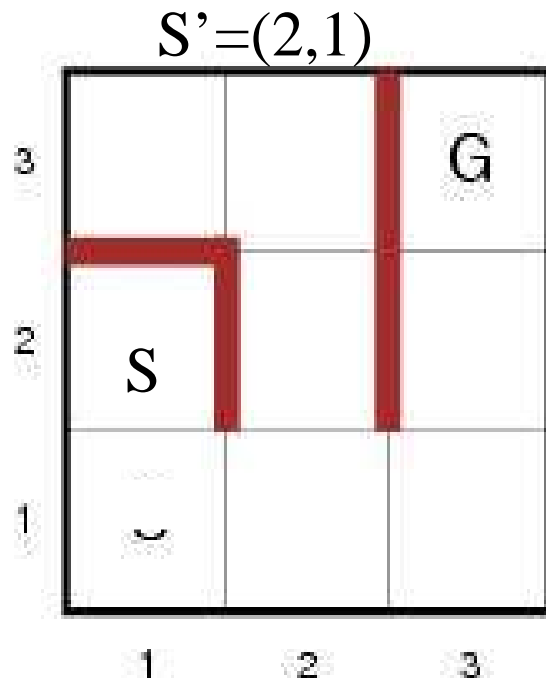
POP(UX[(1,1)]) \rightarrow a

□ A=ARRIBA

s = (1,1)

Return a

Búsqueda Online en Entornos Desconocidos : Agentes de Búsqueda Online, Ejemplo



TEST-META((2,1))?

□ S no! = G Falso

(2,1) es un nuevo estado?

□ Verdadero

□ ACCION((2,1)) -> UX[(2,1)]

■ {ABAJO}

s es nulo?

□ Falso (s=(1,1))

□ resultado[ARRIBA,(1,1)] <- (2,1)

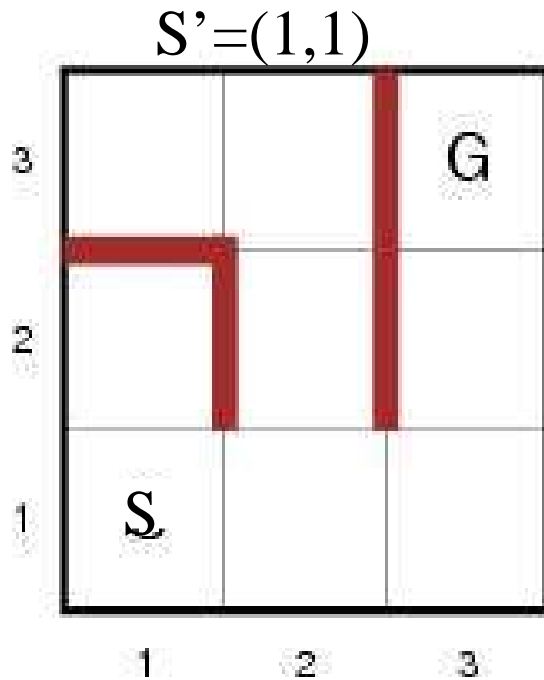
□ UB[(2,1)]={ (1,1) }

UX[(2,1)] vacio?

□ Falso

A=BAJAR, s=(2,1) return A

Búsqueda Online en Entornos Desconocidos : Agentes de Búsqueda Online, Ejemplo



TEST-META((1,1))?

❑ $S \neq G$ Falso

(1,1) un nuevo estado?

❑ Falso

s es nulo?

❑ Falso ($s=(2,1)$)

❑ resultado[ABAJO,(2,1)] \leftarrow (1,1)

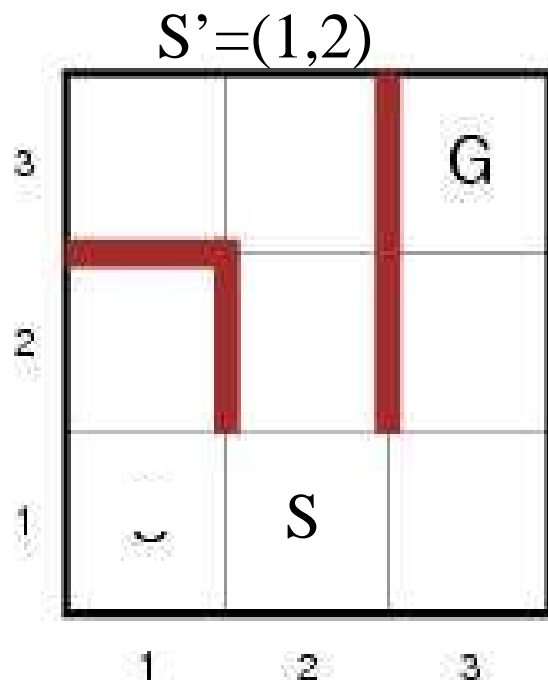
❑ $UB[(1,1)] = \{(2,1)\}$

UX[(1,1)] vacío?

❑ Falso

A=DERECHA, $s=(1,1)$ return A

Búsqueda Online en Entornos Desconocidos : Agentes de Búsqueda Online, Ejemplo



TEST-META((1,2))?

- $S \neq G$ Falso

(1,2) un nuevo estado?

- Verdadero,
 $UX[(1,2)] = \{DERECHA, ARRIBA, IZQUIERDA\}$

s es nulo?

- Falso ($s=(1,1)$)

- $resultado[DERECHA, (1,1)] \leftarrow (1,2)$

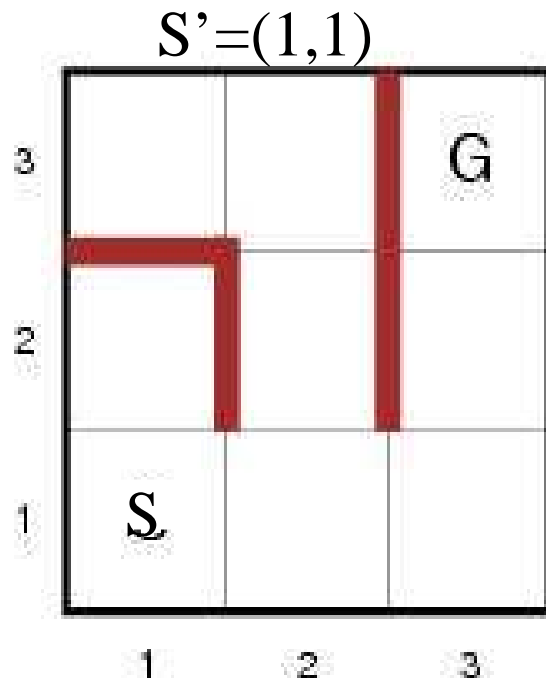
- $UB[(1,2)] = \{(1,1)\}$

$UX[(1,2)]$ vacio?

- Falso

$A=IZQUIERDA, s=(1,2)$ return A

Búsqueda Online en Entornos Desconocidos : Agentes de Búsqueda Online, Ejemplo



TEST-META((1,1))?

□ $S \neq G$ Falso

(1,1) un nuevo estado?

□ Falso

s es nulo?

□ Falso ($s=(1,2)$)

□ resultado[IZQUIERDA,(1,2)] \leftarrow (1,1)

□ $UB[(1,1)] = \{(1,2), (2,1)\}$

UX[(1,1)] vacio?

□ Verdadero

□ $UB[(1,1)]$ vacio? Falso

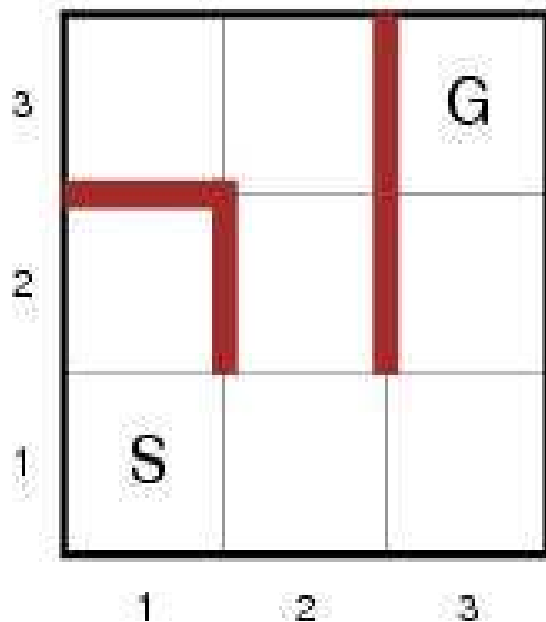
A= b para b en

resultado[b,(1,1)]=(1,2)

□ B=DERECHA

A=DERECHA, $s=(1,1)$...

Búsqueda Online en Entornos Desconocidos : Agentes de Búsqueda Online, Ejemplo

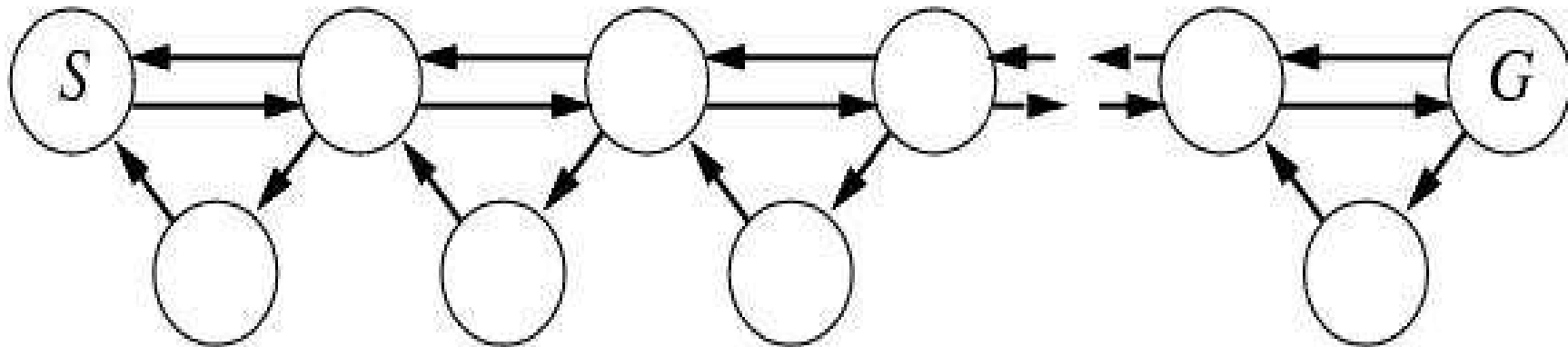


- Peor caso cada nodo es visitado dos veces.
- Un agente puede ir en un largo camino incluso si está cerca a la solución.
- Una aproximación online iterativa en profundidad soluciona este problema.
- Búsqueda DF Online
DF trabaja solo cuando las ACCIONES son reversibles.

Búsqueda Online en Entornos

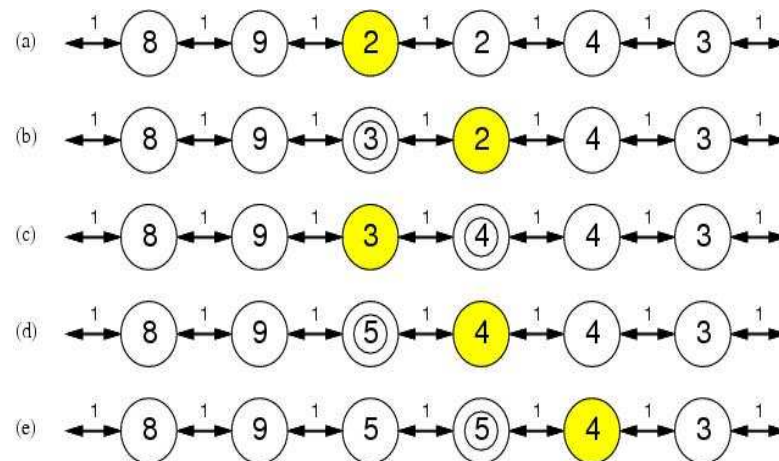
Desconocidos : Búsqueda local Online

- Hill-climbing es online
 - Un estado es almacenado.
- Mala performance debido a máximos locales
 - Reinicios aleatorios imposible.
- Solución: Caminos aleatorios introducidos en la exploración (puede producir exponencialmente muchos pasos)



Búsqueda Online en Entornos Desconocidos : Aprendizaje en Búsqueda Online

- Solución 2: agregar memoria al algoritmo hill climbing
 - Almacenar la mejor estimación actual $H(s)$ del costo de llegar a la meta
 - $H(s)$ es inicialmente la heurística estimada $h(s)$
 - Luego actualizada con experiencia (mirar debajo)
- Aprendizaje en tiempo real A* (LRTA*)



Referencias Bibliográficas

- Capitulo 4 Artificial Intelligence: A Modern Approach, Russell and Norvig
 - <http://www.aaai.org/AITopics/pmwiki/pmwiki.php/AITopics/Search>
-