

INTELIGENCIA ARTIFICIAL (1INF24)



UNIDAD 1: Introducción a la IA. Búsqueda y optimización en IA

Tema 3: Algoritmos heurísticos, metaheurísticos y bioinspirados
(Parte 1)

Dr. Edwin Villanueva Talavera

Contenido

- Algoritmos heurísticos de búsqueda y optimización local
 - Hill-climbing
 - Simulated annealing



Búsqueda local y optimización

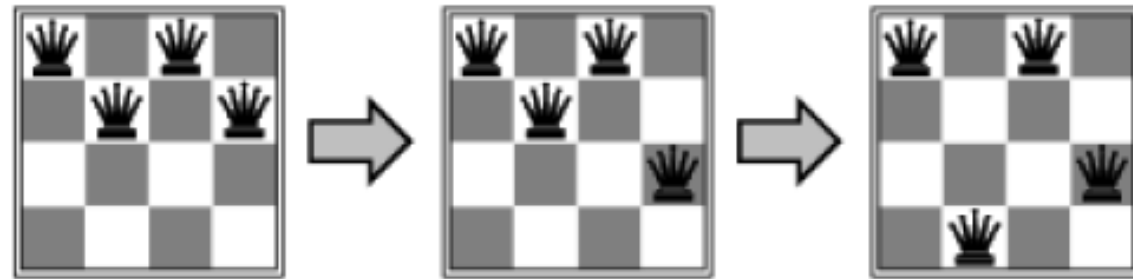
- Los algoritmos vistos hasta ahora exploran espacios de búsqueda manteniendo en memoria los caminos explorados, cuando encuentran un estado objetivo, el camino hacia él constituye una solución
- En muchos problemas de optimización el camino hasta el objetivo es irrelevante:
 - Cuando se quiere encontrar solo una configuración objetivo, sin importar la secuencia de acciones
- En ese caso se puede usar algoritmos de búsqueda local:
 - No se mantiene un árbol de búsqueda, sólo un nodo con el estado actual
 - Iterativamente se modifica el estado actual usando información de los estados vecinos



Búsqueda local y optimización

Ejemplo: Problema de las n-reinas

- Colocar n reinas en un tablero $n \times n$, tal que cada fila, columna y diagonal tenga solo una reina:



Búsqueda local y optimización

Características de métodos de búsqueda local:

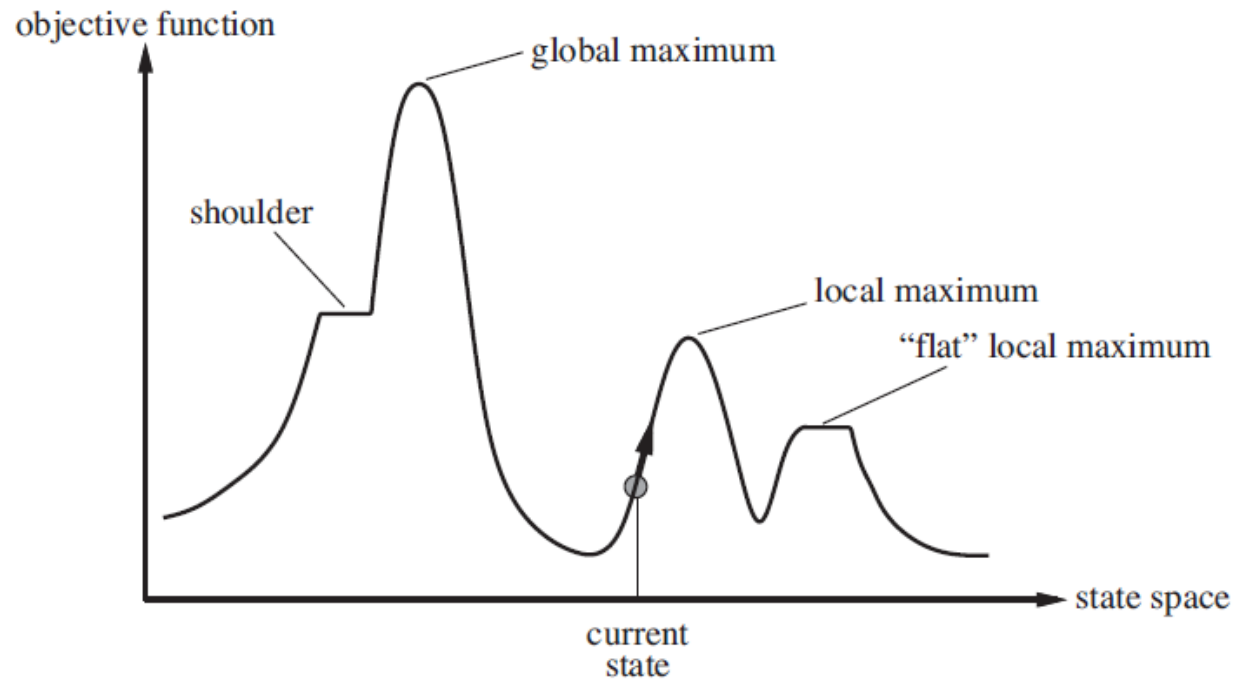
- Usan muy poca memoria, usualmente una cantidad constante
- A menudo encuentran soluciones razonables en espacios de estados enormes o infinitos (continuos), donde los métodos que exploran caminos no funcionarían
- Pueden ser útiles para resolver problemas de **optimización** puros, donde se necesita encontrar el mejor estado de acuerdo a una **función objetivo**



Búsqueda local y optimización

Superficie de espacio de estados:

- La superficie tiene localización (estado) y elevación (definido por la función objetivo o costo)



Método Hill-Climbing

- Usa un solo nodo que mantiene el estado actual y el valor de la función objetivo o función de costo
- Solo examina los valores de los estados vecinos al estado actual
- Continuamente se mueve en la dirección donde se obtiene el mayor incremento si es función objetivo o mayor decremento si es función costo
- Termina cuando se encuentra un pico (o valle) donde ningún estado vecino tiene mas alto (mas bajo) valor



Método Hill-Climbing

Pseudocódigo:

```
function HILL-CLIMBING(problem) returns a state that is a local maximum  
  current  $\leftarrow$  MAKE-NODE(problem.INITIAL-STATE)  
  loop do  
    neighbor  $\leftarrow$  a highest-valued successor of current  
    if neighbor.VALUE  $\leq$  current.VALUE then return current.STATE  
    current  $\leftarrow$  neighbor
```


Método Hill-Climbing

Limitaciones:

- Puede no encontrar la solución óptima por las siguientes razones:
 - **Máximos locales:** picos de la superficie del espacio de estados donde ningún estado vecino tiene mayor valor
 - **Crestas:** secuencias de picos que son muy difíciles de explorar
 - **Mesetas:** áreas planas del espacio de estados donde hill-climbing se quedaría perdido



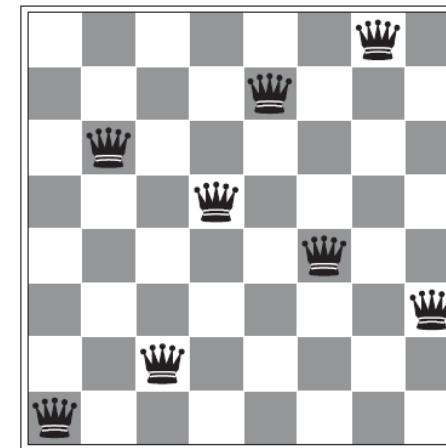
Método Hill-Climbing

Ejemplo: Problema de las 8 reinas

- Imaginemos el problema de las 8 reinas con función de costo:
 h = número de pares de reinas que se atacan mutuamente
- Sucesores:** todos los posibles estados generados de mover una simple reina de un casillero a otro en la misma columna

Valores de h de todos los sucesores posibles del estado actual ($h = 17$)

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



Un posible mínimo local con $h = 1$

Existe una probabilidad de 84% de atracarse en mínimo local!



Método Hill-Climbing

Posibles mejoras :

- Movimientos laterales para evitar mesetas
 - Sin embargo puede ocurrir repetición infinita, hay que imponer un limite para el numero de movimientos laterales.
- Stochastic hill-climbing
 - Escoje aleatoriamente el sucesor con probabilidad proporcional al aumento de la función objetivo
- Reinicios aleatorios (Random-restart Hill-climbing)
 - Ejecutar varias búsquedas a partir de varios estados iniciales escogidos aleatoriamente.
 - Es completa, pues en el peor de los casos irá acabar generando el estado objetivo como estado inicial, sin embargo es ineficiente.

Método Simulated Annealing

- Combina *hill-climbing* con caminos aleatorios (*random walk*), lo que le garantiza completitud y una mayor eficiencia
- Simula el proceso de templado de metales: al inicio se da una gran temperatura para luego ir bajando gradualmente:
 - Gran temperatura significa una alta probabilidad de que estados vecinos con pobre evaluación puedan ser escogidos como sucesores. Esto le permite escapar de óptimos locales.
 - Esa probabilidad disminuye con el pasar del tiempo

Método Simulated Annealing

Pseudocódigo:

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
           schedule, a mapping from time to “temperature”

  current  $\leftarrow$  MAKE-NODE(problem.INITIAL-STATE)
  for  $t = 1$  to  $\infty$  do
     $T \leftarrow$  schedule( $t$ )
    if  $T = 0$  then return current
    next  $\leftarrow$  a randomly selected successor of current
     $\Delta E \leftarrow$  next.VALUE – current.VALUE
    if  $\Delta E > 0$  then current  $\leftarrow$  next
    else current  $\leftarrow$  next only with probability  $e^{\Delta E/T}$ 
```

Método Simulated Annealing

Propiedades:

- Es posible probar que si T decrece lo suficientemente lento, la búsqueda puede encontrar una solución óptima global con probabilidad tendiendo a 1
- *Simulated Annealing* es muy usada en software de diseño de circuitos integrados, optimización de redes de telecomunicaciones, *planificación* de instalaciones industriales, etc.

Bibliografía



Capítulo 4.1 del libro:

Stuart Russell & Peter Norvig “[Artificial Intelligence: A modern Approach](#)”,
Prentice Hall, Third Edition, 2010

¡Gracias!

