Búsqueda Local

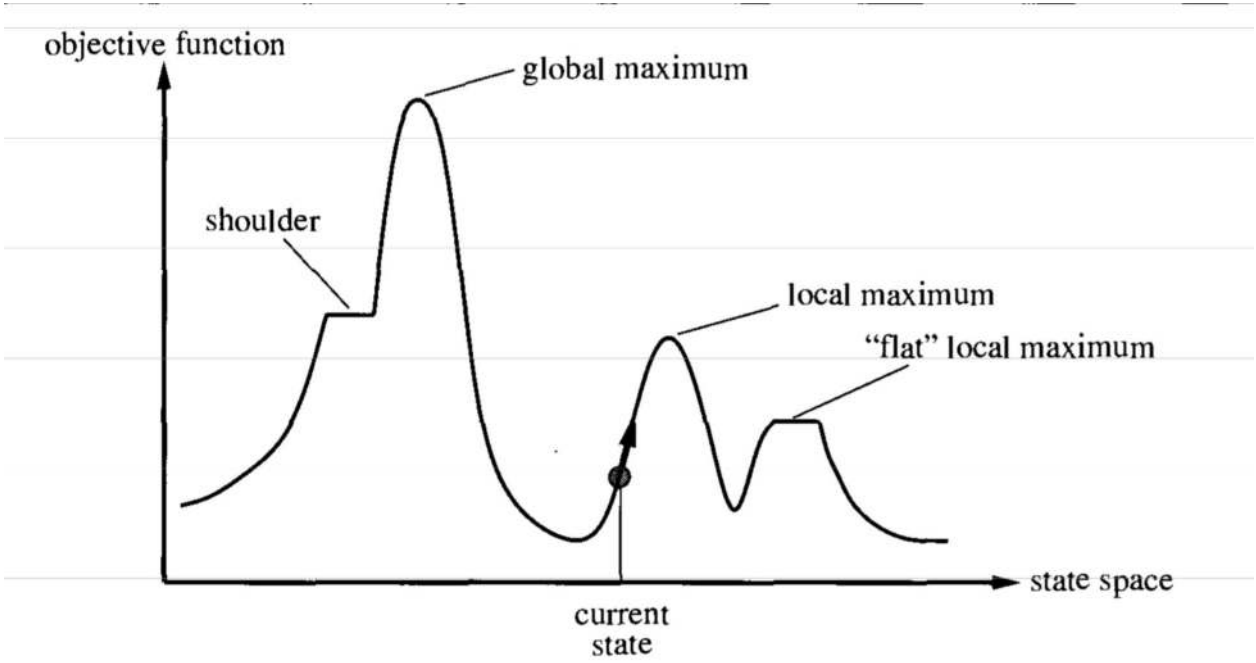
Julio Godoy
DIICC

Búsqueda local

- En muchos problemas, el camino hacia el objetivo es irrelevante
  - Lo importante es encontrar el estado objetivo
- Operan en base a un nodo único
  - Camino no es recordado
- Se mueven a nodos vecinos
- Ventajas:
  - Bajo requerimiento de memoria
  - Encuentran soluciones en espacios de estados muy grandes e incluso infinitos

Búsqueda local

- Útiles también para problemas de optimización
  - Objetivo es encontrar el mejor estado en base a una función objetivo
- Formulación de estado completo



Búsqueda Hill Climbing

- “Ascenso de colinas”
- Es un ciclo que se mueve en la dirección de valor creciente hasta alcanzar una ‘cima’
- Sólo considera los sucesores inmediatos del estado
- También llamada búsqueda local avara (greedy)
- Progresa con rapidez hacia una solución

Algoritmo Hill Climbing

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
   inputs: problem, a problem
   local variables: current, a node
                    neighbor, a node

   current ← MAKE-NODE(INITIAL-STATE[problem])
   loop do
      neighbor ← a highest-valued successor of current
      if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
      current ← neighbor
```

Figure 4.11 The hill-climbing search algorithm (**steepest ascent** version), which is the most basic local search technique. At each step the current node is replaced by the best neighbor; in this version, that means the neighbor with the highest VALUE, but if a heuristic cost estimate h is used, we would find the neighbor with the lowest h.

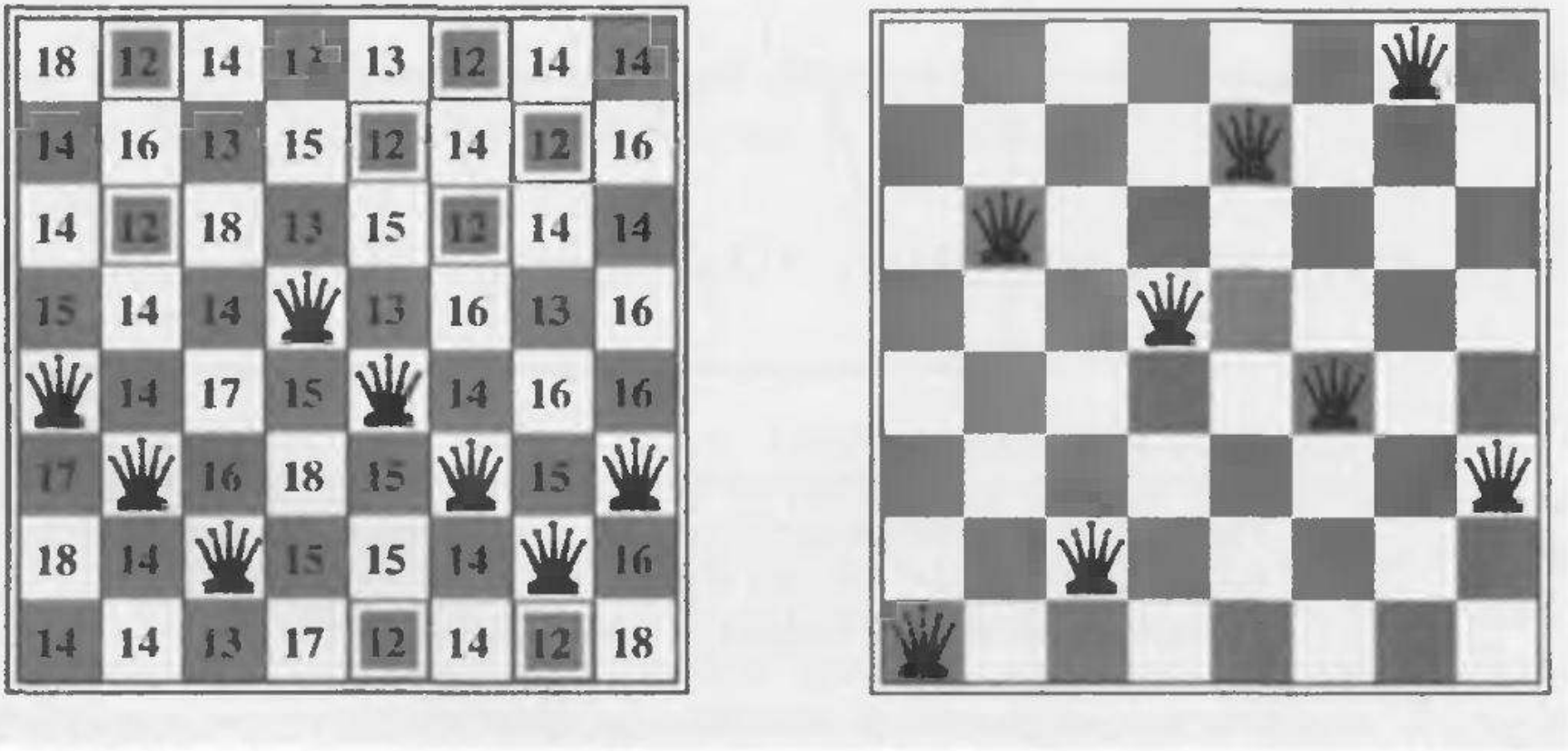


Figure 4.12 (a) An 8-queens state with heuristic cost estimate $h = 17$, showing the value of h for each possible successor obtained by moving a queen within its column. The best moves are marked. (b) A local minimum in the 8-queens state space; the state has $h = 1$ but every successor has a higher cost.

Búsqueda Hill Climbing

- Problemas:
  - Máximo local
  - Crestas (cordilleras)
  - Mesetas

- Variaciones:
  - Hill climbing estocástico
  - Hill climbing de primera opción
  - Hill climbing de reinicio aleatorio

Búsqueda de Templado simulado

- También llamada “Simulated Annealing”
  - Basado en parámetro ‘temperatura’
- Genera sucesor al azar
  - Si sucesor es mejor evaluado que actual, lo ‘acepta’
  - Si no lo es, con probabilidad < 1 lo acepta
    - Probabilidad basada en qué tan peor es el sucesor
    - Temperatura baja con cada iteración
- Aplicaciones en:
  - Programación de procesos en industrias
  - Optimización de gran escala

Algoritmo Templado simulado

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  inputs: *problem*, a problem
    schedule, a mapping from time to "temperature"
  local variables: *current*, a node
    next, a node
    T, a "temperature" controlling the probability of downward steps

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

Figure 4.14 The simulated annealing search algorithm, a version of stochastic hill climbing where some downhill moves are allowed. Downhill moves are accepted readily early in the annealing schedule and then less often as time goes on. The *schedule* input determines the value of T as a function of time.

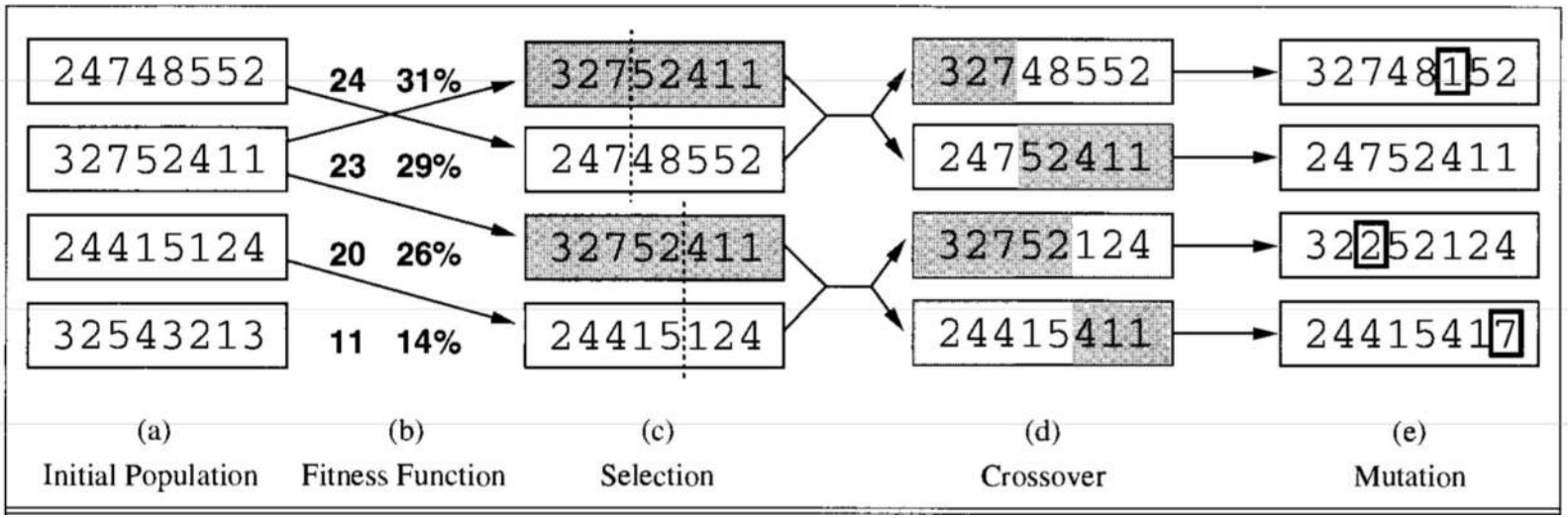
Búsqueda de ‘beam’ local

- Mantiene k estados en lugar de 1
- De todos los sucesores, escoge los k mejores
- Información es ‘compartida’ entre las búsquedas

- Problemas:
  - Puede concentrarse con rapidez en una región pequeña del espacio de estados
  - Variante estocástica:
    - Escoge k sucesores, con probabilidad basada en su valor

Algoritmos genéticos

- Variante de búsqueda ‘beam’
- Sucesores generados por combinación de dos nodos ‘padres’
  - Comienza con una ‘población’
  - Cada ‘individuo’ o estado representado como un string sobre un alfabeto finito
  - Cada estado evaluado con función de ajuste (o fitness) y elegido para ‘reproducción’ según esta.


24748552
32752411
24415124
32543213
24 31%
23 29%
20 26%
11 14%
32752411
24748552
32752411
24415124
32748552
24752411
32752124
24415411
32748152
24752411
32252124
24415417
(a)
Initial Population
(b)
Fitness Function
(c)
Selection
(d)
Crossover
(e)
Mutation