

# Tema 6: Búsqueda de Soluciones en IA

## 6.1. El problema de la búsqueda. Métodos Heurísticos y Metaheurísticos.

Búsqueda en IA. Heurística. Tipos. Búsqueda sistemática (global), Búsqueda local.

Variaciones Algoritmo A: IDA, SMA

Metaheurísticas. Características. Tipos.

¿Elección de una metaheurística?

## 6.2. Metaheurísticas Poblacionales (Evolutivas). Algoritmos Genéticos.

Esquema General.

Codificación y Decodificación.

Función Fitness.

Operaciones: Cruce, Mutación, Reemplazo.

Ejemplos.

## Bibliografía

- Monografía: Metaheurísticas. Inteligencia Artificial, Vol 7, No 19 (ed. B. Melián, J.A. Moreno Pérez, J. Marcos Moreno-Vega) (2003). <http://journal.iberamia.org/>
- **Inteligencia Artificial. Técnicas, Métodos y Aplicaciones.** Palma, Marín, 2008 (Mc Graw Hill). Cap. 9, 11
- Computational Intelligence: An introduction. Andries P. Engelbrecht, Wiley Ed. 2<sup>a</sup> ed. 2007.
- **Handbook of Metaheuristics**, Fred Glover and Gary Kochenberger (1<sup>a</sup> ed, 2003), M. Gendreau, J.Y.Potvin (2<sup>a</sup> ed, 2010) Springer International Series in Operations Research & Management Science.
- **Essentials of Metaheuristics**. Sean Luke. **Online Version**  
<http://cs.gmu.edu/~sean/book/metaheuristics/>
- **Metaheuristics Network web site** (<http://www.metaheuristics.org/>)
- **How to Solve It: Modern Heuristics.** Z. Michalewicz, D. Fogel. 2ed. 2004 (Springer)
- Experimental Evaluation of Heuristic Optimization Algorithms: A Tutorial, Ronald L. Rardin and Reha Uzsoy, Journal of heuristics, 7(3), 2001.

# Tema 6.1.- El problema de la búsqueda de soluciones.

## Métodos Heurísticos y Metaheurísticos

### Búsqueda de soluciones.

- Factibilidad y Optimalidad. Explosión combinatoria.

### Tipología de la Búsqueda:

- Búsqueda Sistemática (global), Búsqueda Local.
- Métodos Constructivos, de Mejora, Poblacionales.

### Búsqueda heurística en IA: Algoritmos A y A\*. Variantes.

### Métodos Metaheurísticos. Tipos. Características.

### Elección de una metaheurística. No free lunch.

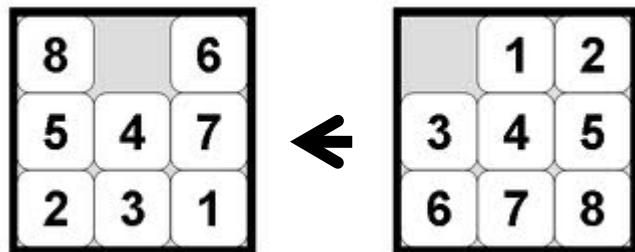
# El problema de la búsqueda de soluciones: Obtención de soluciones mediante búsqueda

El problema de la “*búsqueda de soluciones*” consiste en encontrar soluciones que:

- ✓ Satisfagan las restricciones del problema: **Factibilidad**
- ✓ Maximicen unos determinados criterios (función objetivo): **Optimalidad**

## Problemas de Factibilidad (y Optimalidad)

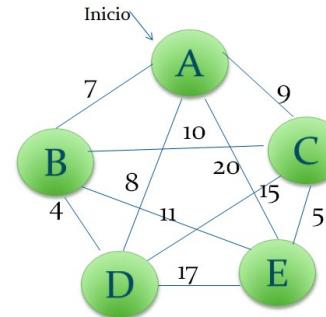
8-puzzle



Difícil encontrar una solución,  
aunque también existen soluciones más óptimas que otras

## Problemas Optimalidad

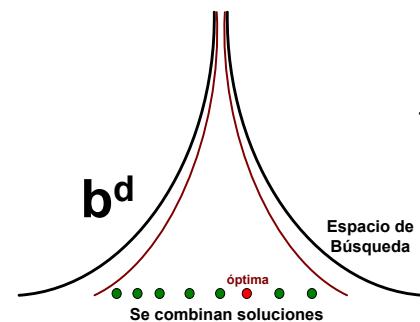
Problema del Viajante



Es trivial encontrar soluciones factibles,  
el problema es encontrar soluciones optimizadas

Problemas de Planificación, Recubrimiento (uni, bi y multidimensional), Diseño y Configuración,  
Asignación de recursos (scheduling), Generación de Horarios, Generación de Rutas,  
Problemas de Distribución, Respuesta a Preguntas, Juegos, Sistemas de Aprendizaje...

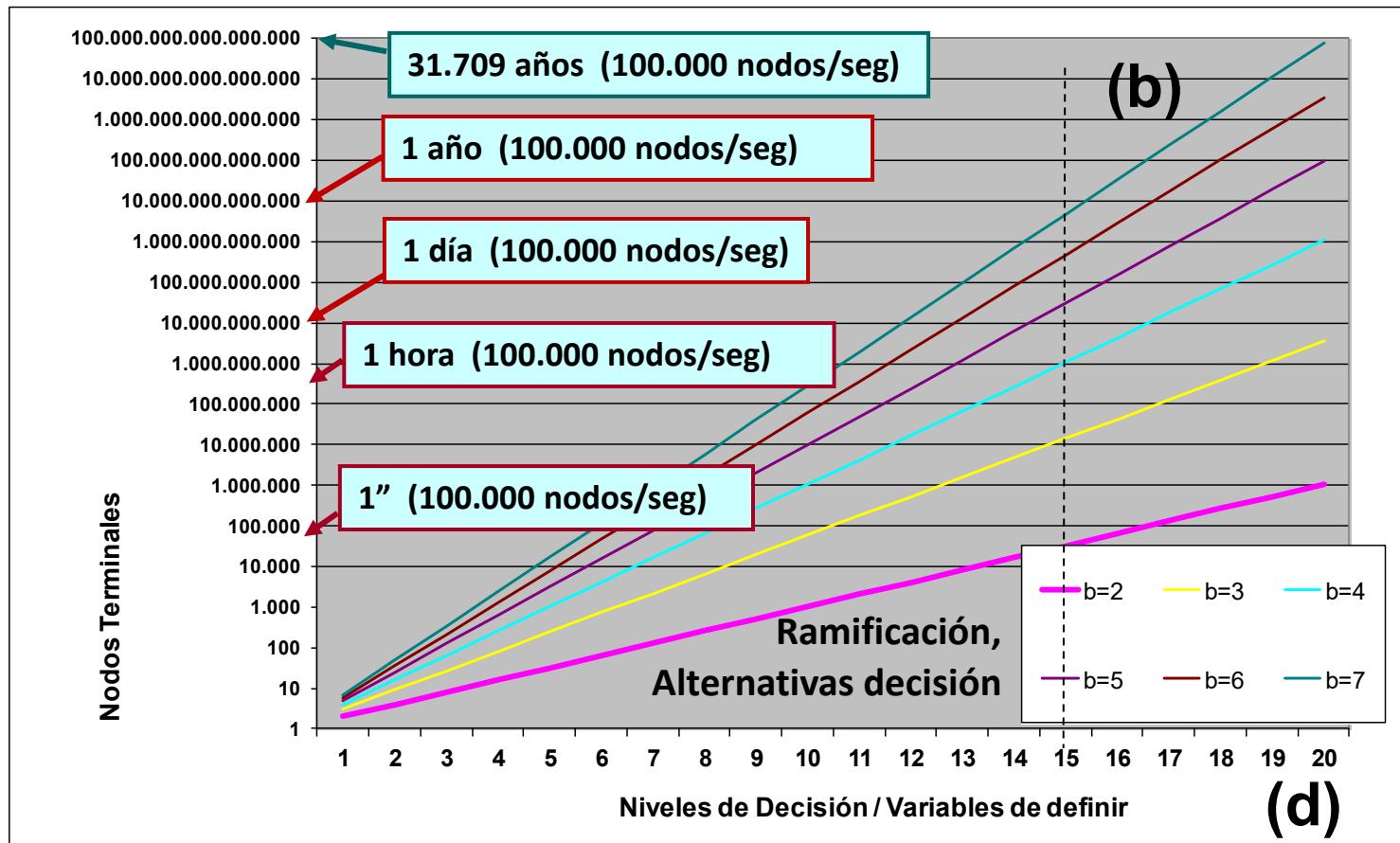
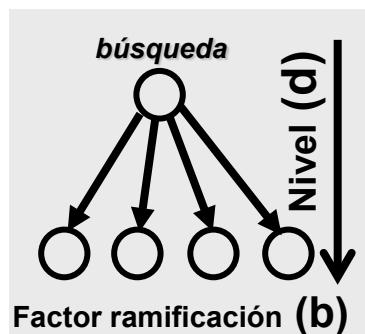
# BÚSQUEDA $\Rightarrow$ explosión combinatoria: $b^d$



En dominios finitos (optimización combinatoria), el conjunto de soluciones es finito y numerable (pero muy grande)



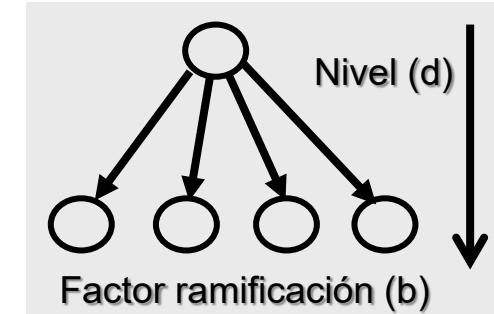
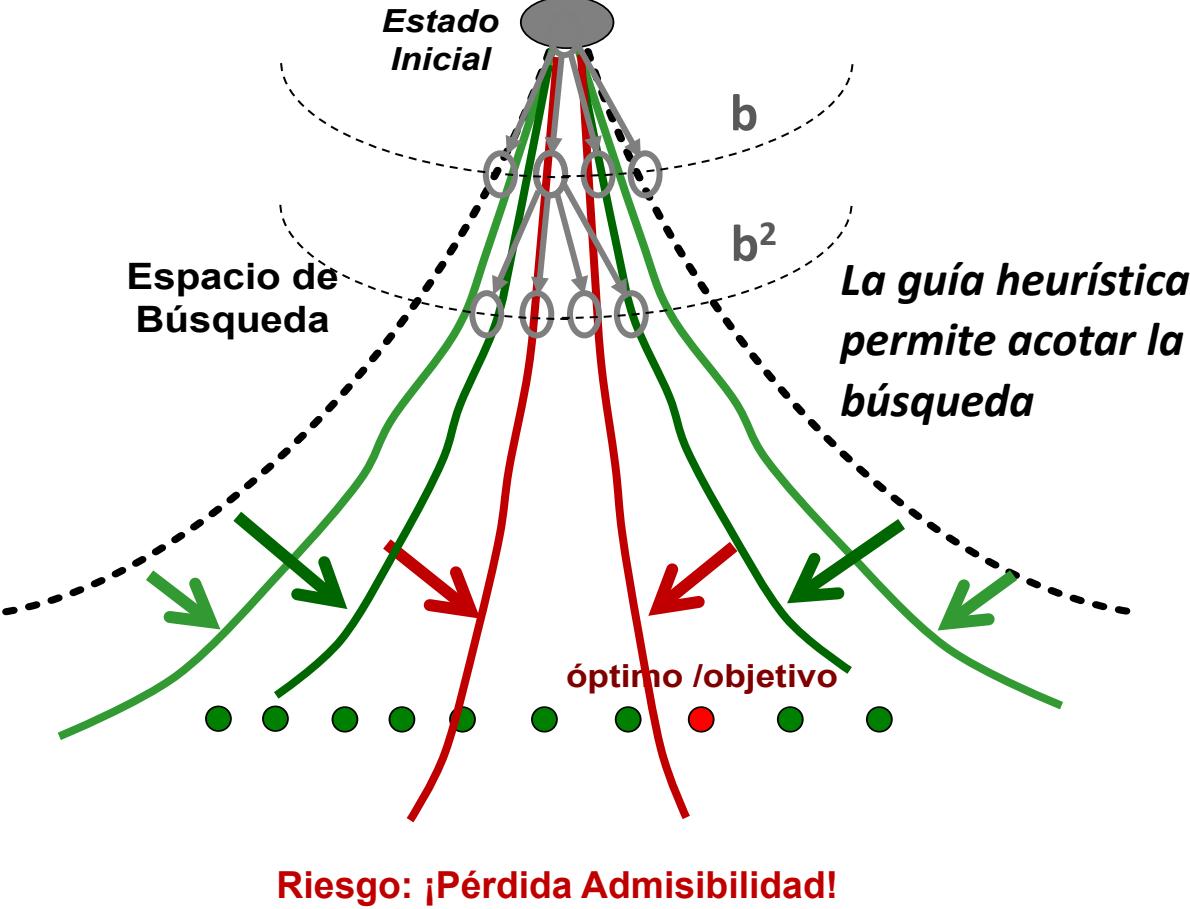
Una fina hoja de papel (0.01 mm) doblada 45 veces, alcanza un espesor de... y ¿46 veces?, ¿54 veces?, ¿103 veces?



# Problemas de complejidad Exponencial:

En general, bastará con una solución **razonablemente buena** (*factible y optimizada, pero no la óptima*)

- Métodos aproximados (como alternativa a los métodos exactos)  $\Rightarrow$  Heurística.
- La solución es obtenida (*generada/mejorada*), mediante un proceso de búsqueda en un amplio espacio de estados/soluciones.



La heurística disminuye el factor efectivo de ramificación ( $b$ ).

Idealmente,  $b \rightarrow 1$ .

Estados no expandidos,  
Alternativas no exploradas, etc.

# Tipología de la búsqueda

## Estrategias de Búsqueda

(Criterio para determinar el siguiente estado en la búsqueda: local vs. global)

### Búsqueda Local:

busca solo en el *entorno local actual.*  
(*irrevocable, escalada...*)

### Búsqueda Sistemática (Global):

busca en *todas* las alternativas posibles.  
(*tentativa, backtracking, algoritmos A y variantes...*)

## Métodos (procesos)

(búsqueda en un espacio de estados/soluciones)

### Métodos Constructivos:

construyen una solución paso a paso.

*Espacio de búsqueda ⇒ espacio de estados.*  
*Búsqueda global/local.*

### Métodos de Mejora:

mejora iterativa de soluciones.

*Espacio de búsqueda ⇒ espacio de soluciones.*  
*Búsqueda local.*

### Métodos Evolutivos:

combinan soluciones previas  
información implícita en la sociedad/población.

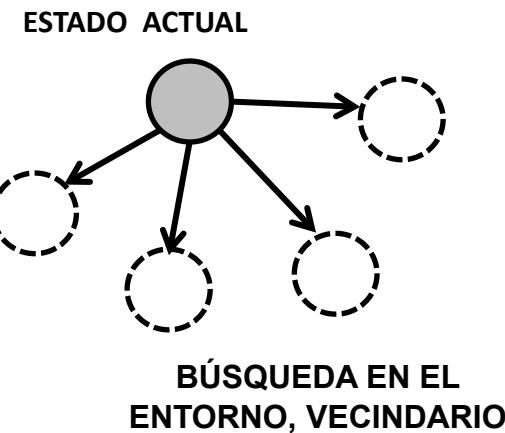
# Estrategia: Criterio para determinar el siguiente estado en la búsqueda

## Búsqueda Local (local search):

En cada iteración, se busca solo en el *entorno del estado actual*.

- *Eficientes en memoria (solo estado actual y sucesores)*
- *Coste temporal sigue siendo exponencial.*
- *Funciones de evaluación* para elegir el mejor estado sucesor.
- *Inconvenientes:*

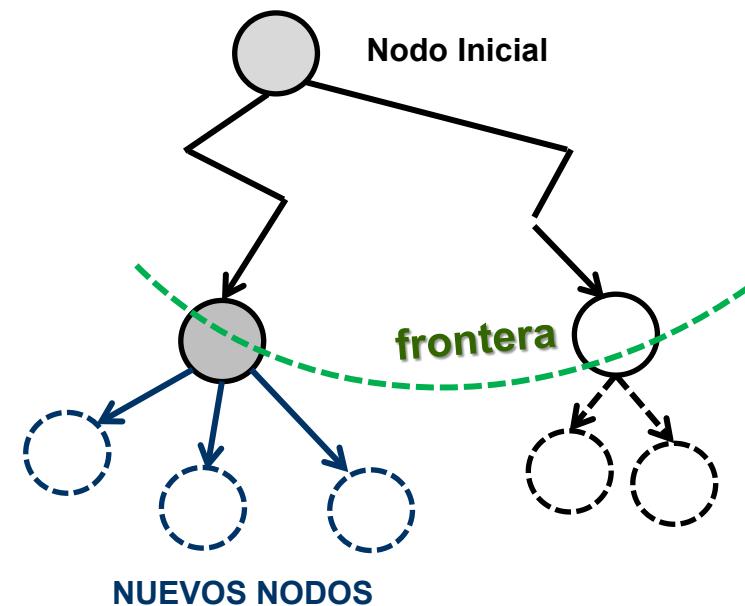
Pueden quedar atrapadas en soluciones que no admiten mejoras en su entorno (óptimos locales, valles, mesetas), no garantizan óptima, problemas de incompletitud, ciclos, etc.



## Búsqueda Global (global search):

*Búsqueda sistemática* en todo el espacio de búsqueda.

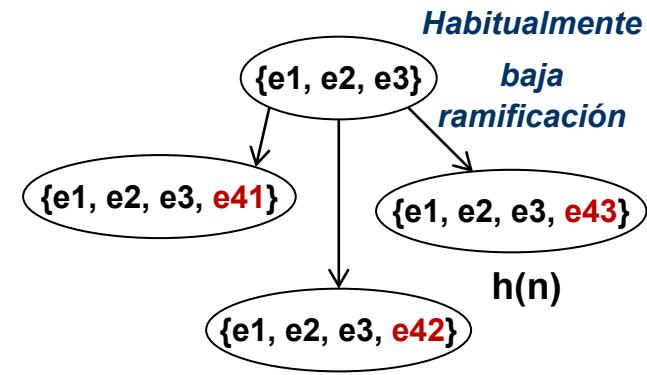
- *Alto coste en memoria (explosión combinatoria).*
- Puede ser completa y admisible (garantizar solución óptima)
- Suelen hacer uso de *funciones heurísticas*.



# Métodos para la obtención de una buena solución

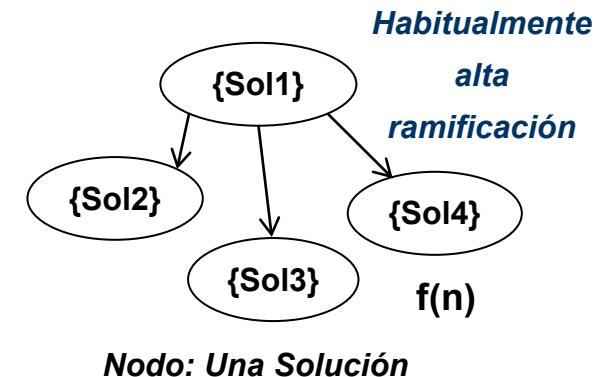
## Métodos Constructivos

- **Construyen** una solución al problema: cada iteración añade un elemento.
- **Heurística** para **estimar** la bondad de cada estado (solución parcial).
- El factor de ramificación (**b**) depende del número alternativo de elementos a añadir en cada paso.
- El nivel (**d**) depende de los elementos de la solución.
- **No requieren solución inicial** (aplicación en problemas en donde lo difícil es obtener una solución): *Problemas de factibilidad*.



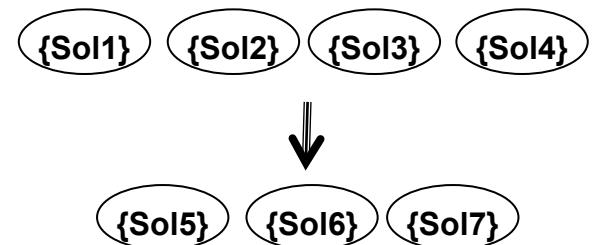
## Métodos de Mejora (habitualmente llamados Búsqueda Local)

- Requieren **solución inicial** (constructiva / aleatoria).
- Cada iteración intenta **mejorar** la solución actual.
- Nivel de expansión depende de las **soluciones vecinas** (alternativas) a la actual (generalmente muy alto), por lo que suele aplicarse búsqueda local.
- Utilizan una **función para la evaluación** de soluciones.
- Comportamiento **any-time (ventaja)**.

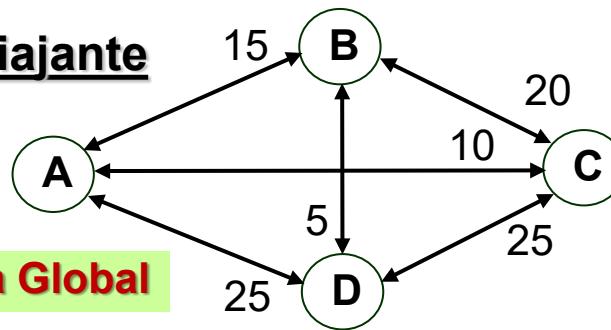


## Métodos Evolutivos

- A partir de un conjunto (**población**) de soluciones, se seleccionan, recombinan y modifican *individuos* para obtener un nuevo conjunto de soluciones.
- Utilizan una **función para la evaluación** (*fitness*) de soluciones.
- Comportamiento **any-time (ventaja)**.

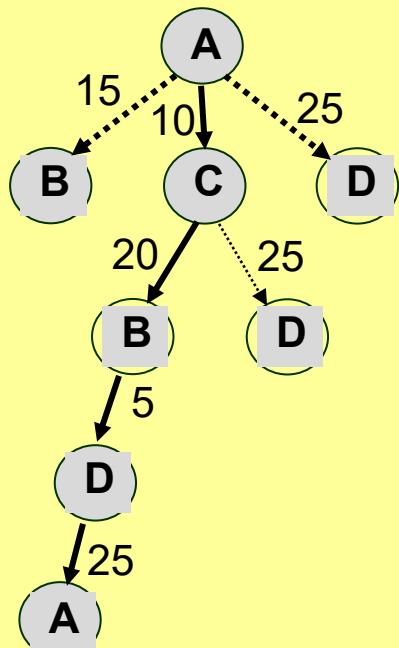


## Ejemplo: Problema del Viajante

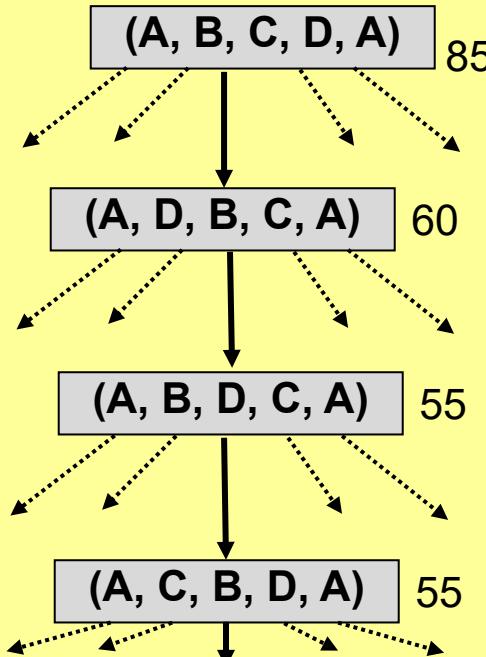


Búsqueda Local / Búsqueda Global

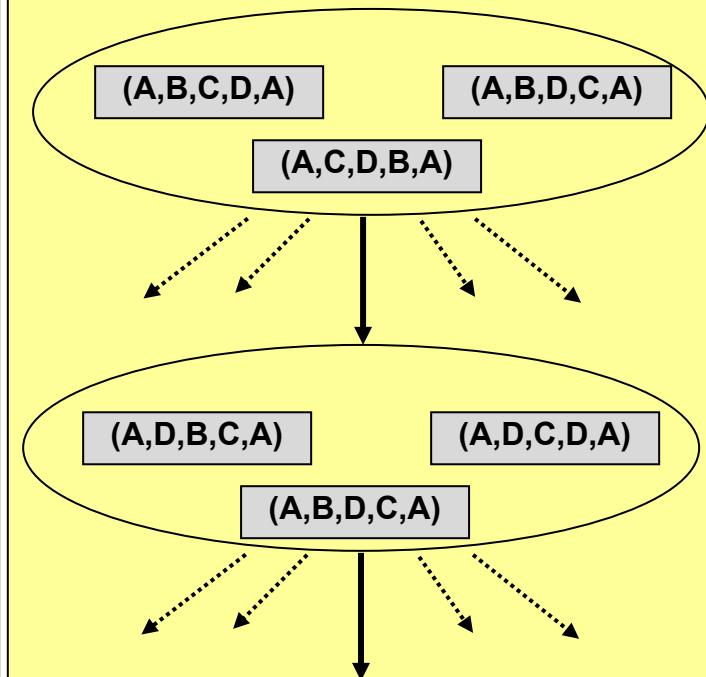
### Método Constructivo



### Método de Mejora

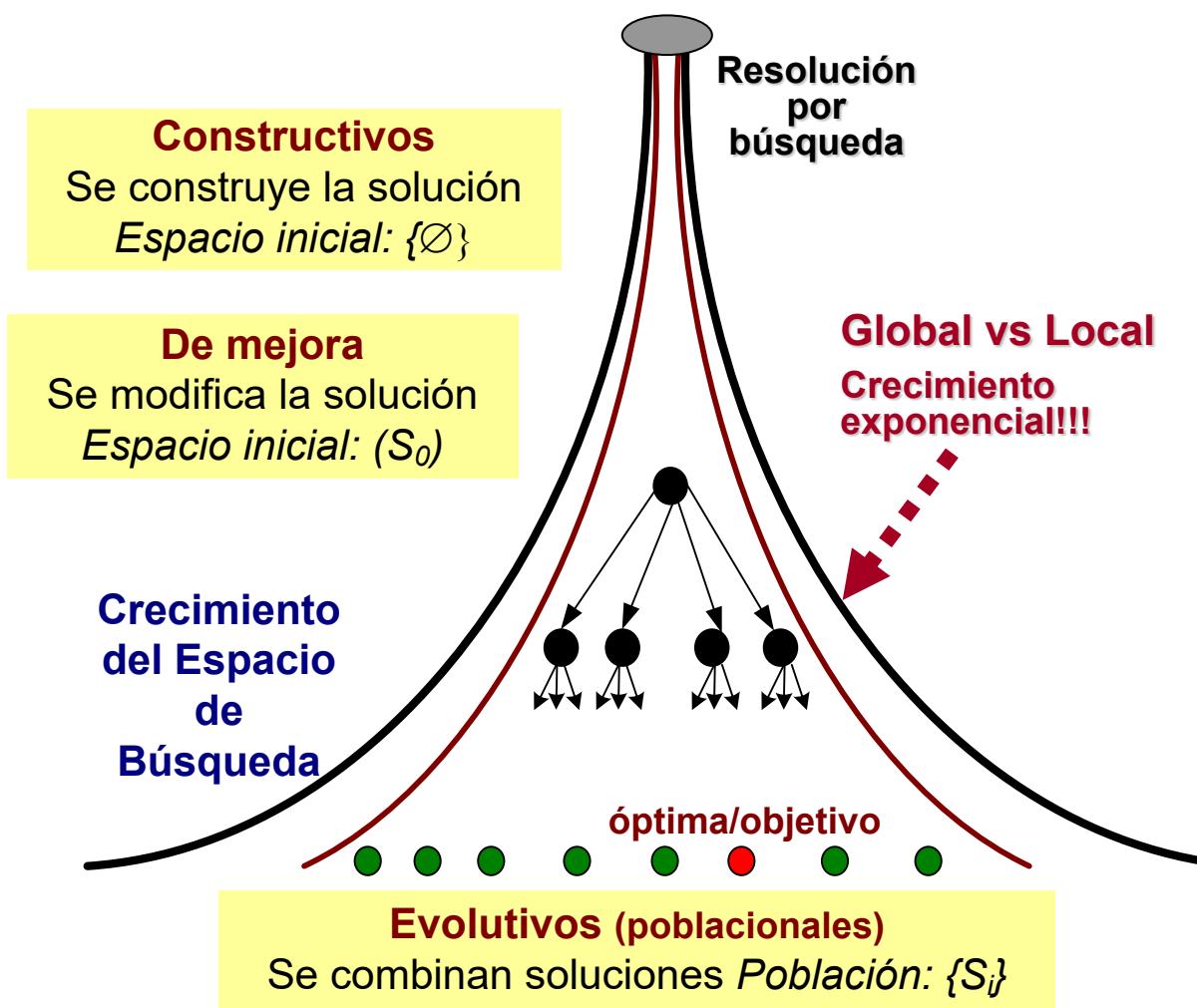


### Método Evolutivo

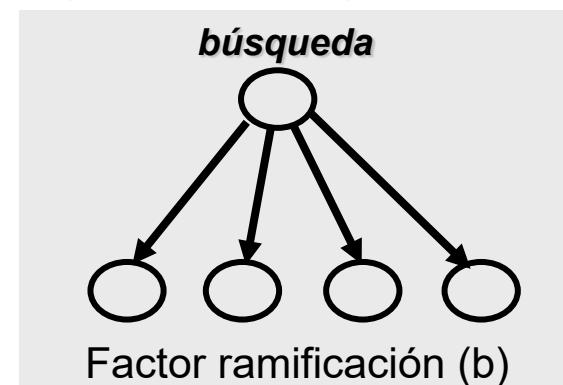


Alternativa:  
Soluciones alternativas

# En resumen...



Crecimiento Exponencial  
(Memoria / Tiempo)

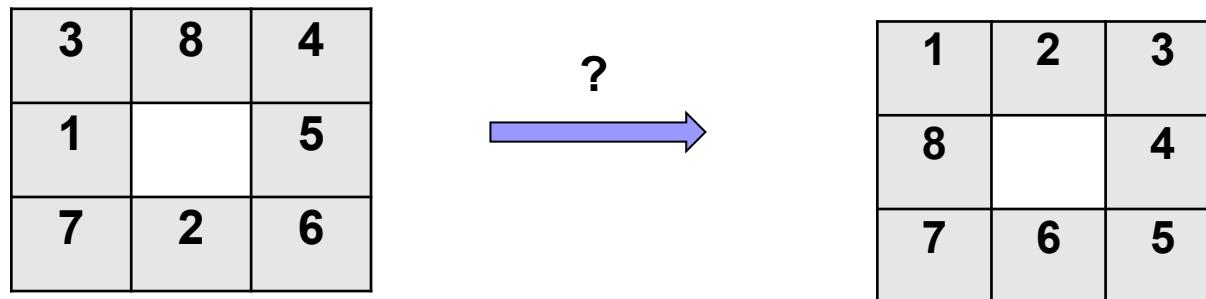


Los métodos se pueden aplicar de forma híbrida:

- **De mejora**: partiendo de sucesivas soluciones iniciales
- **Constructivos**: repitiendo la generación de una solución
- **Poblacionales**: generando nuevas soluciones

# Ejercicios. ¿Cómo abordarías estos problemas usando un método constructivo vs. uno de mejora?

**Ejercicio 1 (juego del n-puzzle).** Queremos pasar de una configuración inicial a una final con el menor número de movimientos.



**Ejercicio 2 (lectores y periódicos).** Existen 3 periódicos (P1..P3) y 4 lectores (L1..L4), que desean leer los periódicos en el mismo orden. Todos pueden empezar a partir de un mismo tiempo y no pueden compartir los periódicos mientras lo leen.

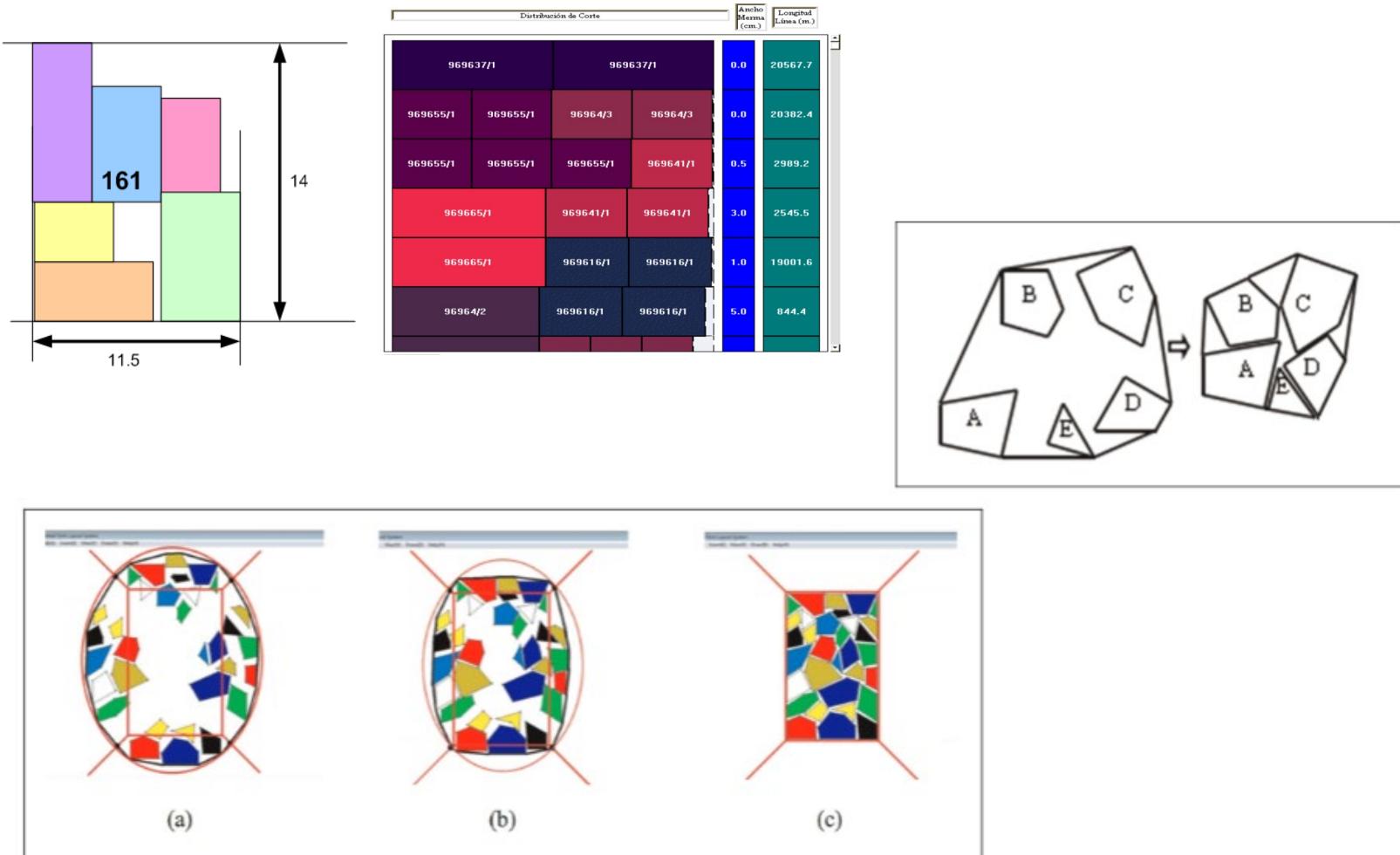
El objetivo es obtener la asignación de periódicos a lectores, tal que se minimice el tiempo total en el que todos los lectores han leído todos los periódicos

	P1	P2	P3
L1	10	15	25
L2	18	20	15
L3	20	30	10
L4	45	45	44



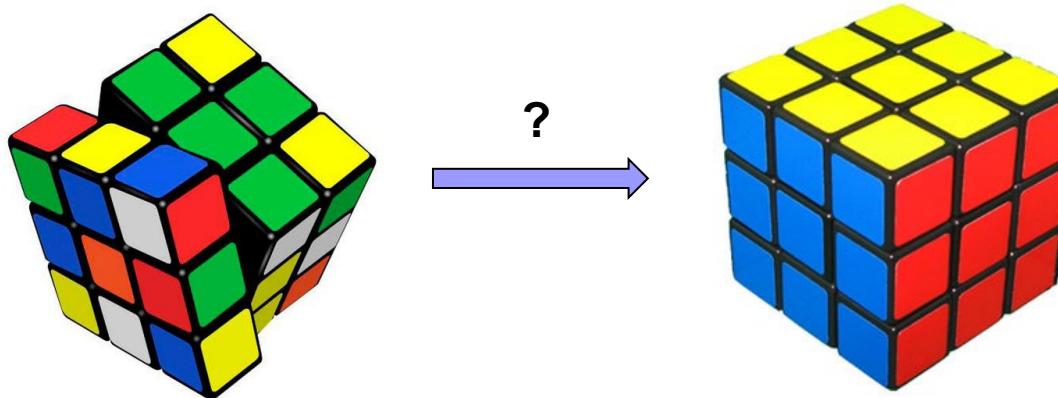
# Ejercicios. ¿Cómo abordarías estos problemas usando un método constructivo vs. uno de mejora?

**Ejercicio 3 (cutting stock problem).** Queremos satisfacer la demanda de corte de todos los productos, pero minimizando la cantidad de merma o material desperdiciado.



# Ejercicios. ¿Cómo abordarías estos problemas usando un método constructivo vs. uno de mejora?

**Ejercicio 4 (cubo de Rubik).** Queremos pasar de una configuración inicial de colores a una objetivo, donde cada cara es de un único color, con el menor número posible de movimientos.



**Ejercicio 5 (invitados a un evento).** Existe un conjunto de personas que hay que distribuir en una mesa en base a sus preferencias de sentarse junto a un amig@. Queremos maximizar las preferencias de todos los invitados tanto como sea posible.



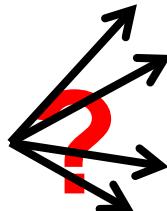
# Búsqueda Heurística. (*heuriskein: encontrar o descubrir*)

## Técnicas Heurísticas: Guía en la búsqueda de soluciones

Técnicas e ideas aplicadas a la resolución de problemas complejos que se espera obtengan una buena solución (no necesariamente óptima) de un modo sencillo y rápido.

- Obtienen soluciones aceptables en tiempos razonables.
- Métodos aproximados, contrapuestos a los métodos exactos (métodos numéricos de optimización).
- Suelen ser específicas para cada problema, aunque sus bases pueden ser trasladadas a otros problemas.
  - Aplica el “**conocimiento sobre el problema**” al método de búsqueda.
  - No siempre es el resultado de un riguroso análisis formal, sino de un **conocimiento intuitivo**, experimental, práctico, de experto, etc. sobre el problema o de su método de resolución.

3	8	4
1		5
7	2	6



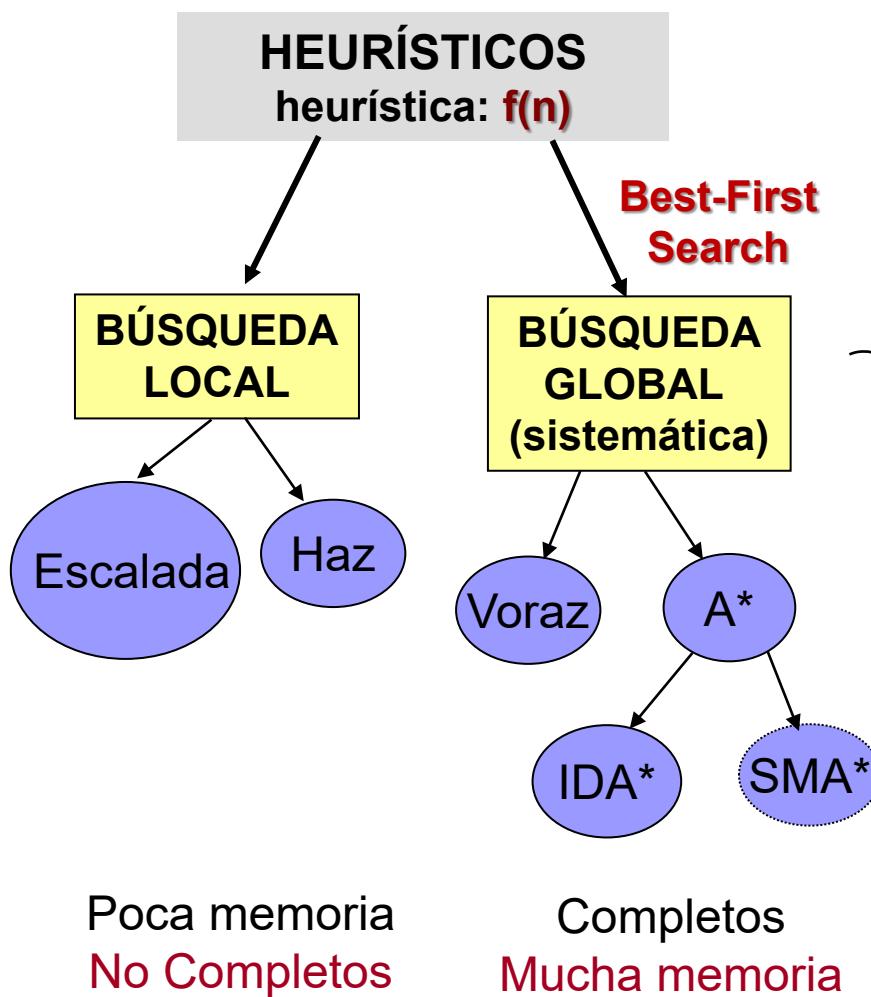
3		4
1	8	5
7	2	6

3	8	4
	1	5
7	2	6

3	8	4
1	2	5
7		6

3	8	4
1	5	
7	2	6

# Búsqueda Heurística en IA



## Búsqueda Heurística

Aplica una función de evaluación  $f(n)$  que mide la bondad del nodo.

- Se busca (expande) a partir del nodo con **mejor  $f(n)$**
- Típicamente,  $f(n)$  tiene una componente **heurística**  
 $h(n)$ = estimación del coste al objetivo  
 $\forall n, h(n) \geq 0, h(\text{meta}) = 0$

### Estrategias:

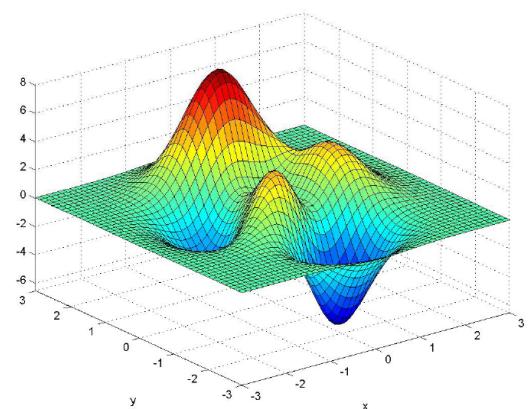
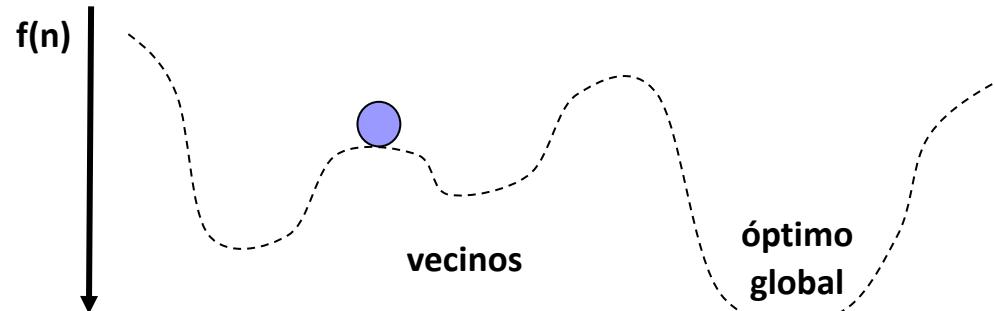
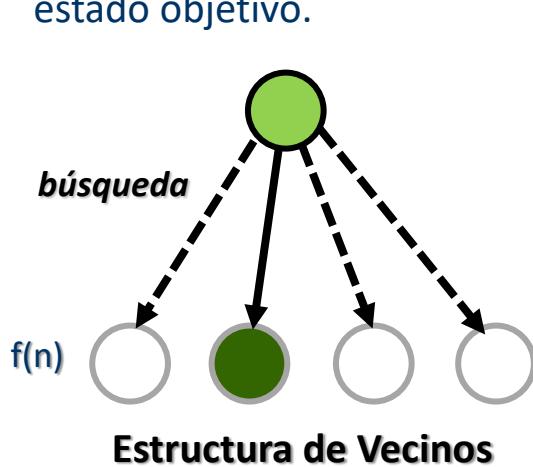
- Búsqueda Global (Best-First Search): permite soluciones óptimas, pero suele requerir mucha memoria.
- Búsqueda Local:  
requieren muy poca memoria;  
pero tienen **problemas** de valles, ciclos, etc. y **no garantizan la solución óptima**.

*Aplicados en los métodos ‘constructivos’/ ‘mejora’ de la solución*

# Búsqueda Heurística en IA: Búsqueda Local, No sistemática

## Función de Escalada (*hill-climbing*), Descenso por Gradiente, Voraz local, etc.

- No mantiene un árbol de búsqueda, sino solo el estado actual (y sucesores inmediatos).
- En cada iteración, busca en el **entorno** (vecindario) del nodo actual, eligiendo el sucesor que **mejora** su  $f(n)$ .
- Realiza unos bucles de búsqueda dirigido hacia el crecimiento de  $f(n)$  (colina arriba, máximo gradiente) en los estados vecinos/sucesores. El proceso acaba cuando no hay mejora posible en el conjunto de soluciones vecinas (valles, llanuras, ciclos, etc.)
- Aplicables cuando el camino hasta el objetivo es irrelevante y solamente interesa encontrar el mejor estado objetivo.



**Ventaja:** Memoria limitada ( $\approx$  constante), ya que no necesitan mantener caminos/alternativas a la solución.

**Inconvenientes:** óptimos locales (no garantiza óptimo global, valles, mesetas, ciclos, etc.).

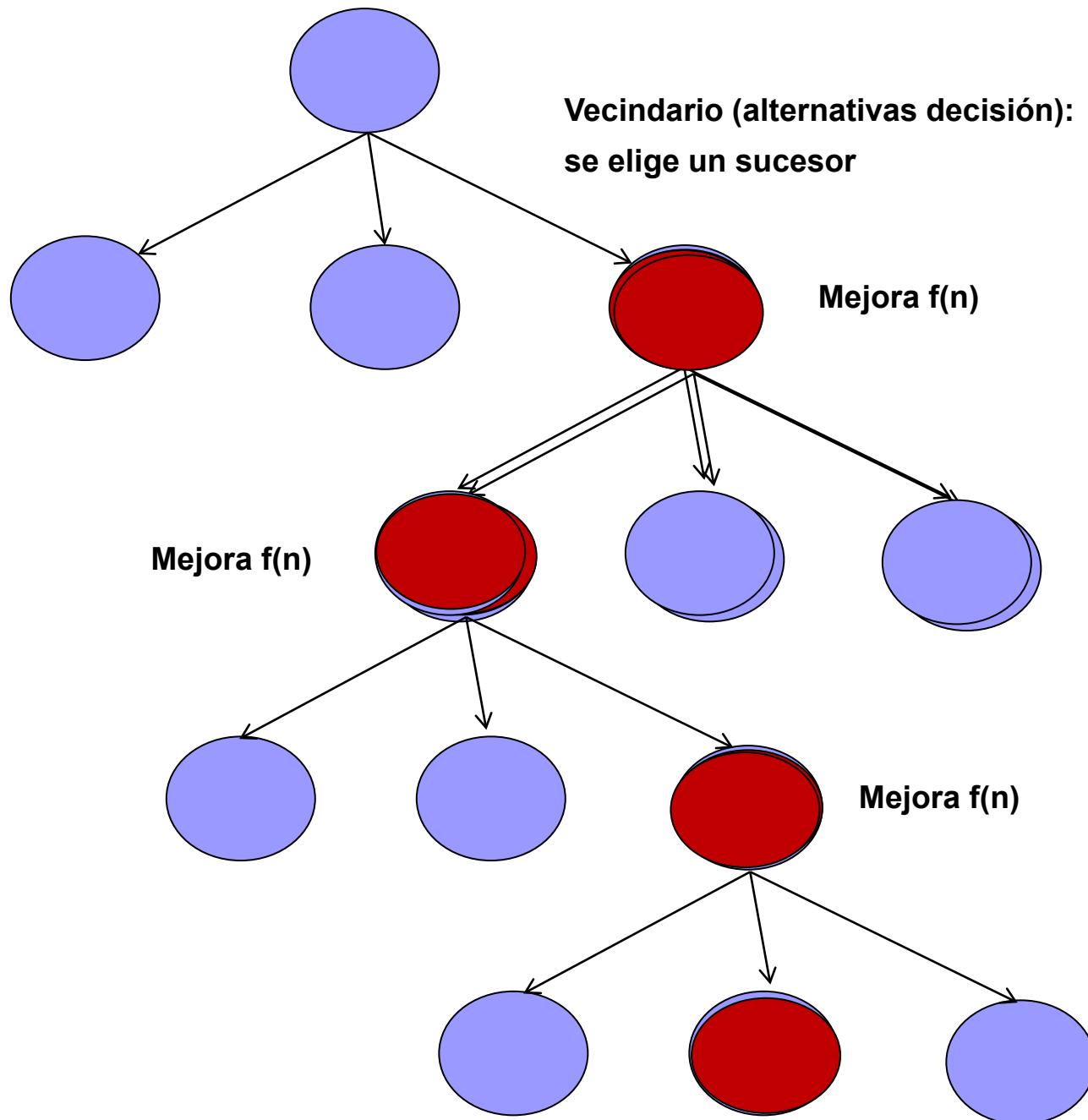
# Búsqueda Local

Un nodo puede representar

- a) Solución parcial (*método constructivo*)
- b) Solución total (*método de mejora*)

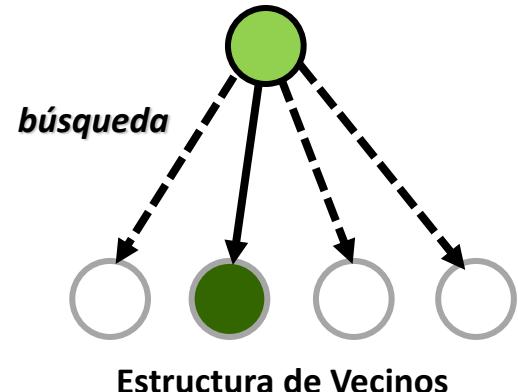
En cada iteración, se busca solo en el *entorno del estado actual*.

- *No conocemos las elecciones ya realizadas*
- *Eficientes en memoria* (solo *estado actual* y *sucesores*), pero el *coste temporal sigue siendo exponencial*
- $f(n)$  para elegir el mejor *estado sucesor*



# Variantes del Método de Escalada (hill-climbing)

Se selecciona un sucesor que suponga una mejora respecto al padre: *Máximos locales función  $f(n)$*



## a) Escalada por máxima pendiente (Best-improvement):

Se generan todos los sucesores y se elige el mejor. No es lo habitual.

Si se puede elegir más de un sucesor con mismo valor de evaluación, se elige al azar.

## b) Escalada estocástica (variante de máxima pendiente):

Se generan todos los sucesores y se escoge aleatoriamente entre los sucesores con mejor valoración que el estado actual.

## c) Escalada Simple (First-improvement): primer sucesor que mejore el actual:

Escalada estocástica de primera opción, donde se generan aleatoriamente sucesores, uno a uno, hasta que uno de ellos mejore el nodo predecesor (solución actual).

## d) Escalada lateral:

Permite movimientos que igualen la función (permiten escapar de llanuras, ciclos).

## Algoritmo General: Hill Climbing (máxima pendiente)

```
Actual ← Estado_inicial  
fin ← falso  
mientras no_fin hacer  
    Hijos ← Sucesores(Actual)  
    si no-vacio (Hijos)  
        entonces  
            Actual ← Escoger_mejor(Hijos)  
        sino  
            fin ← cierto  
    fin  
fin
```

Es 'Máxima Pendiente'.

Si 'Escalada simple': *Generar hijos hasta que mejore estado actual.*

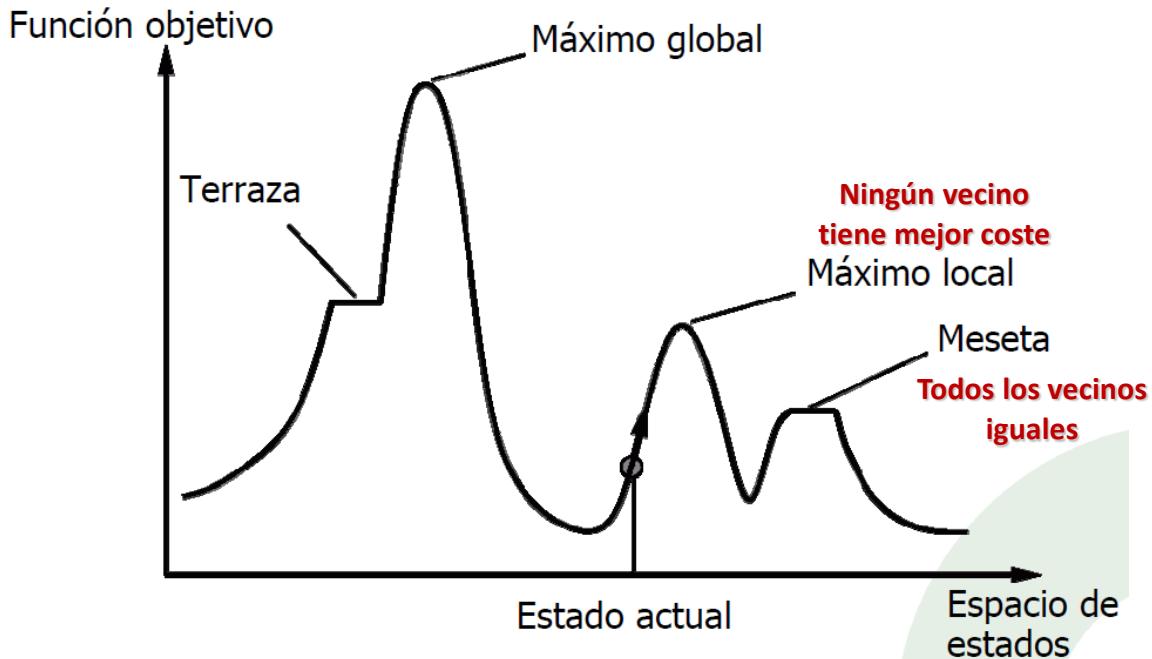
### Condición de Parada

- a) Óptimo Local (ningún sucesor mejora)
- b) Condiciones de parada:
  - Tiempo de cómputo (any time),
  - Nº de iteraciones,
  - Tras  $x$  iteraciones con mejora muy baja,
  - Se alcanza óptimo *razonable*.

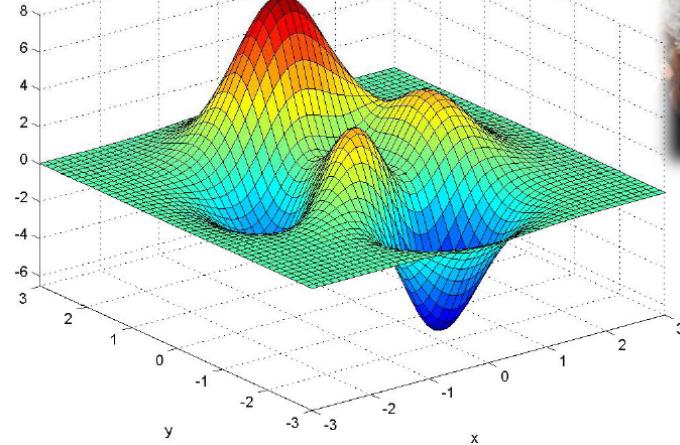
Variante: Se puede usar una pila y guardar los hijos mejores que el padre para hacer backtracking, pero aumentaría el consumo de memoria.

## Inconvenientes:

- La búsqueda se puede parar en un máximo local.
- Se obtienen soluciones localmente óptimas, pero pueden estar muy lejos del óptimo global.
- Búcles (ciclos) y caminos redundantes; debido a no almacenar la secuencia completa en memoria.



*Después de escalar una gran colina, uno se encuentra solo con que hay muchas más colinas que escalar. Nelson Mandela*

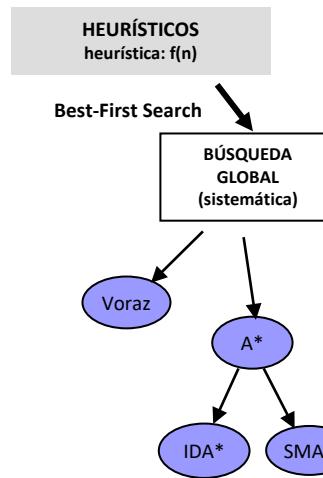


## Posibles soluciones:

Escape de un nodo, en cuyo entorno no existen valores mayores que el actual:

- Salto a otra zona de búsqueda (previamente guardada)  $\approx$  *Beam Search*
- Avanzo al siguiente nivel, o más niveles  $\approx$  *Simulated Annealing*.
- Re-arranque  $\approx$  *Restarting*
- **Metaheurísticas**

# Búsqueda Heurística en IA: Global, Sistemática



Búsqueda sistemática de todo el espacio de búsqueda.

Mantienen abierta toda la frontera de búsqueda: árbol/grafo de búsqueda

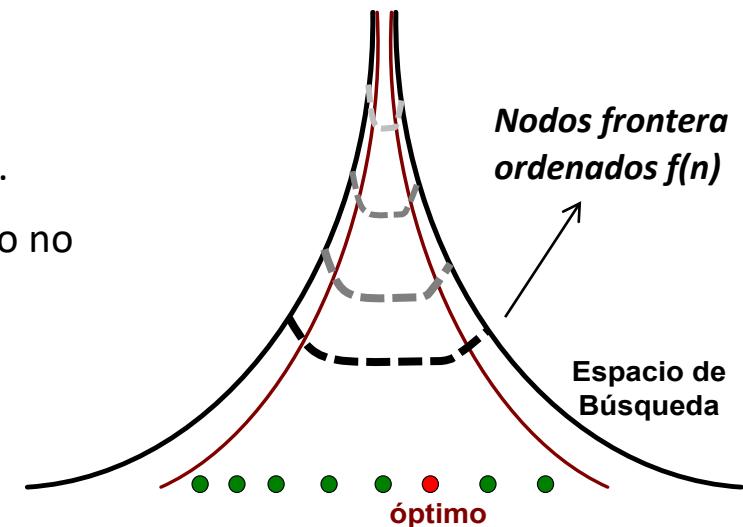
## Búsqueda Voraz (Greedy): $f(n)=h(n)$

- Caso típico de “búsqueda el primero mejor”, donde no importa  $g(n)$
- Completos (con búsqueda global), pero no garantizan la óptima.

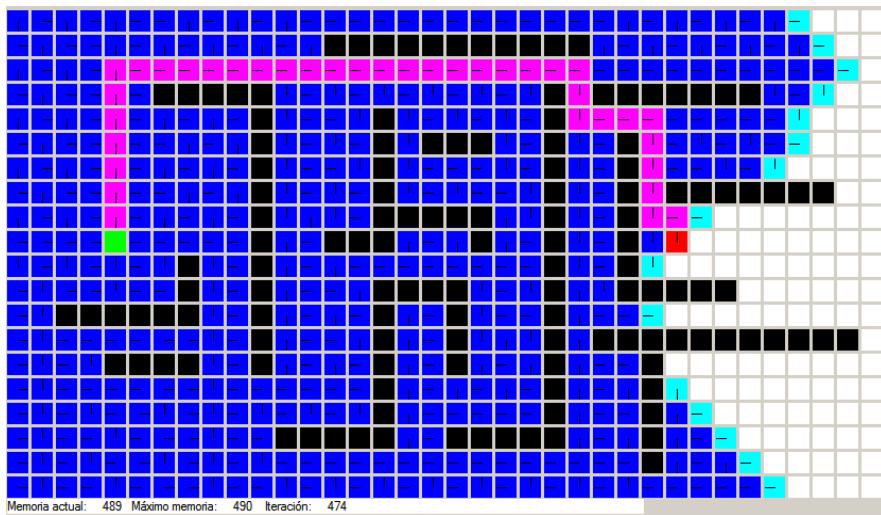
## Algoritmo A: $f(n)=g(n)+h(n)$

Combinación de

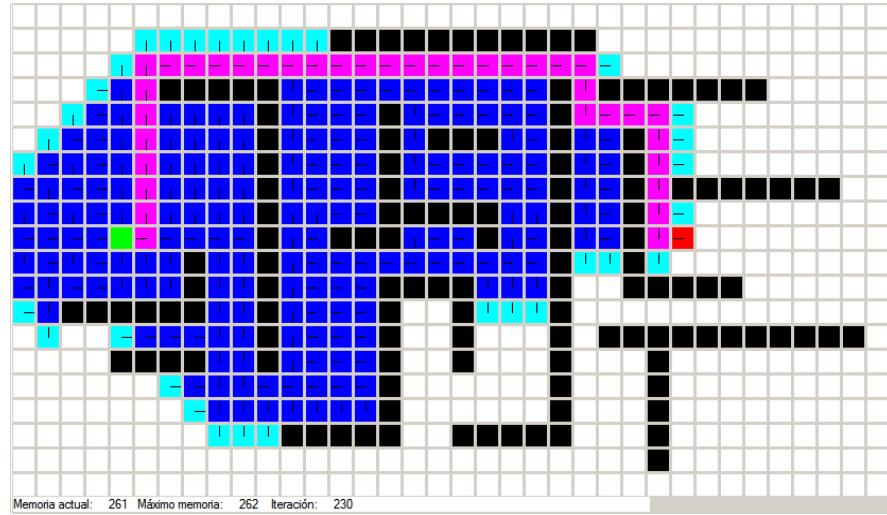
- **coste uniforme**: Completa y óptima,  $g(n)$ , pero ineficiente.
- **búsqueda voraz**: Reduce coste de búsqueda por  $h(n)$ , pero no óptima ni completa, y
- **Algoritmo A\***:  $\forall n, h(n) \leq h^*(n)$   
*Condición para admisibilidad (en árbol, con re-expansión)*  
**Completa y admisible.** ( $h(n)=0 \approx$  Dijkstra, coste uniforme)
- **Pero requiere mucha memoria** (un algoritmo que expanda menos nodos que A\* no garantizará admisibilidad).



### Búsqueda A (h=0)



### Búsqueda A (Manhattan)



## Variantes Algoritmo A:

### Weighted A\* ( $h(n) > h^*(n)$ )

- $f(n) = g(n) + \alpha h(n)$ , donde  $\alpha > 1$ , o bien:  $f(n) = \alpha g(n) + \beta h(n)$
- Es una búsqueda completa, pero no admisible
- Se garantiza encontrar una solución menor que ' $\alpha$ ' veces el coste de la óptima (o acotada por  $1 + \varepsilon$ ), siendo  $\varepsilon$  un parámetro del método.

Además del tiempo computacional,  
**hay un problema de memoria**

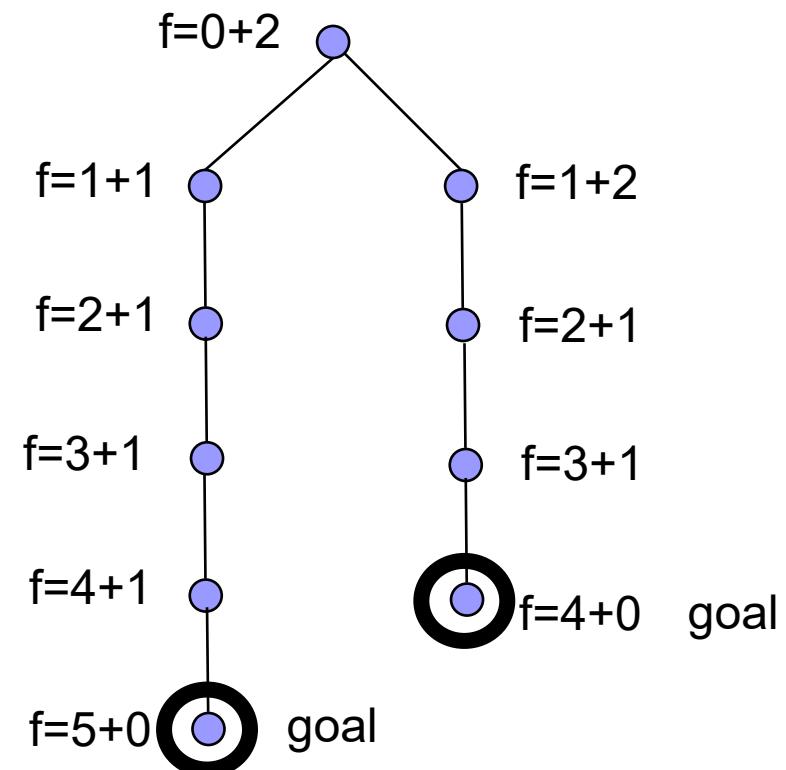
## Iterative-Deepening A\* (IDA\*)

Adapta la *Búsqueda en Profundidad Iterativa (BPI)* a A\*.

- El **criterio de corte** no es el nivel (profundidad), sino el valor de  $f(n) = g(n) + h(n)$
- Cada interacción se reinicia desde la raíz, con un valor de corte: **coste ( $g+h$ ) más pequeño de los nodos que fueron cortados en la iteración anterior.**

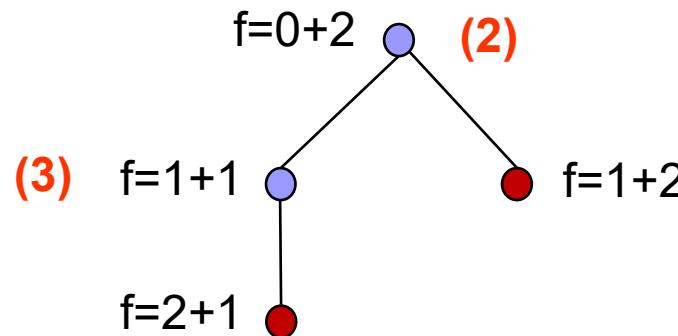
En cada iteración, cuando el valor  $f(n)$  de un nodo supera el valor de corte, se aplica backtracking.

- Si  $h(n) \leq h^*(n)$  IDA\* es admisible
- Útil para problemas de costes  $g(n)$  unitarios
- IDA\* es completo y óptimo y necesita **menos memoria** que A\* (es un iterativo en profundidad), aunque puede generar más nodos en total.
- **Complejidad Temporal:**  $O(b^{2d})$
- **Complejidad Espacial:**  $O(bd)$



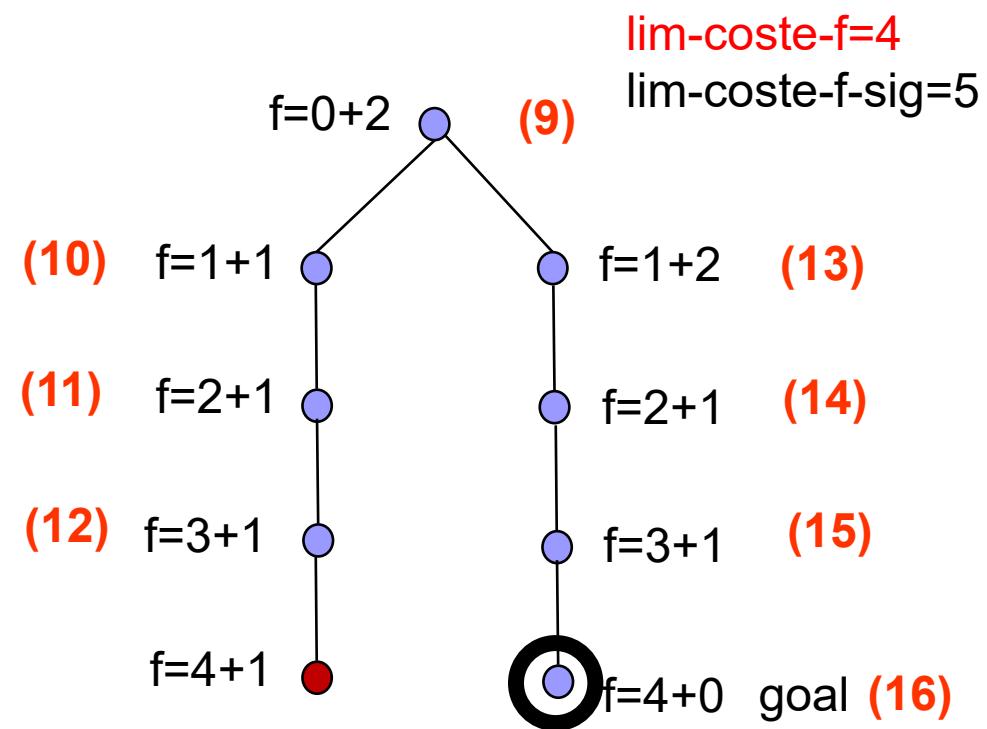
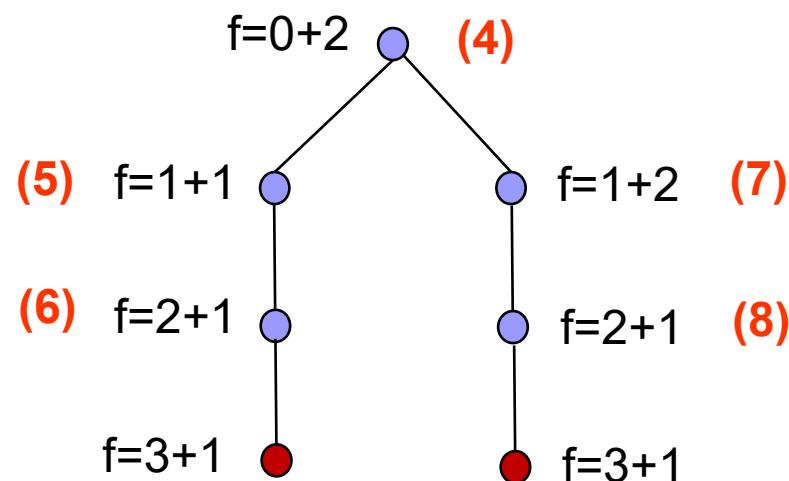
$f=0+2$     (1)

lim-coste-f=2



lim-coste-f-sig=3

lim-coste-f=3  
lim-coste-f-sig=4



## Algoritmo IDA\*

1. Límite de coste  $\text{lim-coste-f} = f(\text{nodo\_raiz})$
2. Lista OPEN = {nodo\_raíz},  $\text{lim-coste-f-sig} = \infty$
3. Si OPEN vacía y  $\text{lim-coste-f-sig} = \infty$ , devolver FALLO.

Si OPEN vacía y  $\text{lim-coste-f-sig} \neq \infty$ ,  $\text{lim-coste-f} = \text{lim-coste-f-sig}$

Entonces Ir a 2 (reinicio).

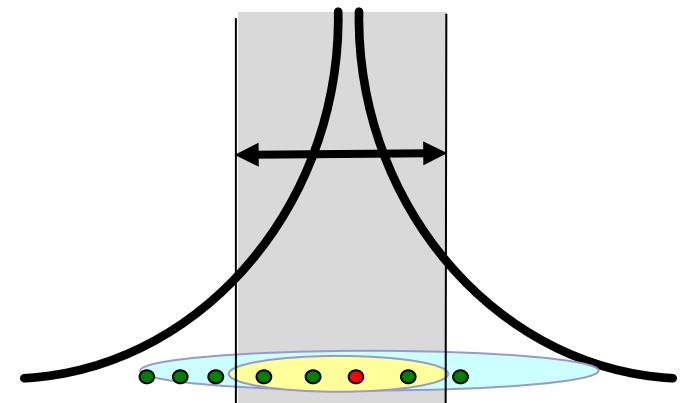
Si\_no,  $n = \text{extrae-primero}(\text{OPEN})$ .

4. Si  $f(n) > \text{lim-coste-f}$ ,  $\text{lim-coste-f-sig} = \min(\text{lim-coste-f-sig}, f(n))$ . Ir a 3.
5. Si  $n$  es objetivo, devolver ÉXITO.  
Si\_no: Expandir  $n$  (generar sucesores de  $n$  y añadirlos a OPEN).  
Ir a 3.

## Búsqueda con memoria acotada: **máximo número de nodos a almacenar (Nmax)**.

Modificación del algoritmo A\*, con un tamaño limitado de la lista open.

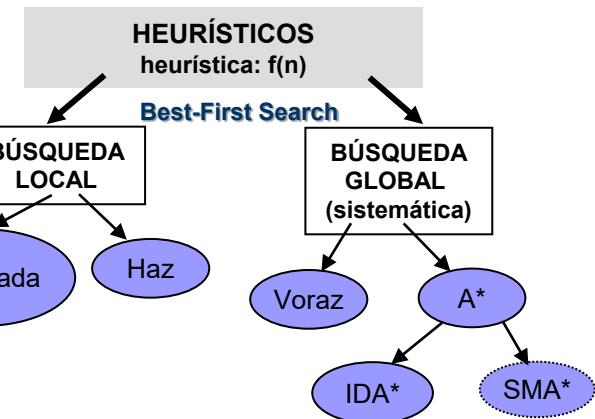
- Cuando no puede almacenar más nodos, se elimina de la frontera del grafo el nodo que tiene el mayor valor de  $f(n)$ : '*nodo olvidado*'
- En cada nodo se recuerda el hijo '*olvidado*' con mejor  $f(n)$ .
- El algoritmo puede '*recuperar*' los nodos olvidados si resultan más prometedores conforme avance la búsqueda (el resto de nodos tiene mayor  $f(n)$ ).



## Características

- Es capaz de adaptarse a la memoria disponible.
- Es completo, si la memoria disponible es suficiente para almacenar la senda más corta.
- Mejor para costes  $g(n)$  no unitarios.
- Es admisible, si la memoria disponible es suficiente para un camino óptimo.
- En otro caso, ***devuelve la mejor solución que puede alcanzar con la memoria disponible.***

# En resumen:



<b>Búsqueda Heurística Local</b>	No Admisibles, Poca memoria Problemas de ciclos, etc.
<b>Búsqueda Heurística Global</b>  <b>Algoritmo A, A*</b>	Búsqueda Global, sistemática Admisibles y completos.  <b>Alto coste memoria / tiempo</b>
<b>Incrementar potencia heurística <math>h(n) &gt; h^*(n)</math> o Variantes Algoritmo A*</b>	¿Pierden admisibilidad?  Bajan / Acotan coste memoria  Siguen requiriendo mucha memoria

<b>Metaheurísticos:</b>  <b>Diseño y estrategias heurísticas</b>	<b>Soluciones Optimizadas (no admisibles)</b>  <b>Bajo coste temporal</b>
--	---

# Estrategias Heurísticas

**Métodos de Mejora**  
(Búsqueda en un Espacio de Soluciones)

**Métodos constructivos**  
(Búsqueda en un Espacio de Estados parciales)

**Espacio de búsqueda**

**(y variantes)**

**óptimo**

**La idea básica de la metaheurística es:**

- Mejorar la búsqueda local,
- Incorporando ideas de ‘búsqueda global’,
- Minimizando la probabilidad de quedar atrapados en óptimos locales,
- Introduciendo componentes de decisión estocásticos y métodos bio-inspirados.

## METAHEURÍSTICAS

### Local Search

- Eficientes en memoria.
- **Inconvenientes:** no garantizan la óptima, incompletitud, ciclos, etc.

**Local / Global**  
**Heurística: guía de la búsqueda**

### Global Search

- Completos y admisibles.
- Pueden garantizar la solución óptima.
- Alto coste en memoria.

# Métodos Metaheurísticos

(termino introducido en “Future paths for integer programming and links to artificial intelligence” F. Glover, 1986)

- Métodos **aproximados**, diseñados para resolver **problemas difíciles** de optimización combinatoria en los que los métodos heurísticos clásicos no son efectivos.
- Se sitúan conceptualmente por encima de los métodos heurísticos:  
**Procedimientos iterativos, que **modifican (modulan)** la decisión de una **heurística subordinada** durante el proceso de búsqueda para producir eficientemente soluciones de alta calidad.**
- Proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la **inteligencia artificial**, la **evolución biológica** y los mecanismos **estadísticos**.
- Obtienen **soluciones optimizadas a complejos problemas** de búsqueda de soluciones en tiempos razonables.
- Evaluación: Eficiencia (*tiempo*), eficacia (*optimalidad*) y efectividad (*para diversos escenarios*).
- Evolución continua. **Complejo ajuste de parámetros** (por ejemplo, ver: <http://www.cs.ubc.ca/labs/beta/Projects/ParamILS/>)
- Los tipos básicos son (*aunque hay diversas categorizaciones*):
  - métodos **constructivos híbridos**,
  - métodos de mejora por búsqueda local,
  - métodos evolutivos, y
  - métodos de inteligencia social (enjambre).

# Clasificación de Metaheurísticas

## Metaheurísticas constructivas:

- Construyen iterativamente la solución en un espacio de estados (*soluciones parciales*) incorporando sucesivamente *elementos* a la misma. Las metaheurísticas constructivas incorporan criterios para seleccionar los elementos de una buena solución.
- **GRASP** (Greedy Randomized Adaptive Search Procedure, **1<sup>a</sup> fase**).

## Metaheurísticas de mejora (en entornos, búsqueda local):

- Recorren el espacio de soluciones transformando iterativamente soluciones de partida: Mejora iterativa de la solución mediante búsqueda local en el vecindario.
- Inconveniente de la búsqueda local: óptimos locales, ciclos, etc.
- Mejora metaheurística: *incorporar una estrategia de búsqueda global* para escapar de óptimos locales de baja calidad (que es el principal problema):
  - a) metaheurísticas de **arranque múltiple**: establecen criterios para reiniciar la búsqueda desde otra solución de arranque (**multi-arranque, backjumping**).
  - b) metaheurísticas de **entorno variable**: modifican la estructura de entornos locales que se aplica: **Búsqueda Tabú** (*búsqueda con memoria*), **Búsqueda en Haz**.
  - c) metaheurísticas de **búsqueda no monótona**: permiten movimientos o transformaciones de la solución que no sean de mejora: **Enfriamiento Simulado** (*simulated annealing*).

# Clasificación de Metaheurísticas

## Metaheurísticas evolutivas:

- Parte de un **conjunto de soluciones** que van evolucionando, reconstruyendo o recombinando. La estrategia heurística consiste en generar, seleccionar, combinar y reemplazar un conjunto de soluciones.
- **Algoritmos Genéticos**, **Búsqueda Dispersa** (Scatter search), **Algoritmos Meméticos**.

**Inteligencia de Enjambre** (*Particle Swarm Optimization*): **Inteligencia colectiva, descentralizada**.

Inspirada en el comportamiento de sociedades de partículas (hormigas, pájaros, peces).

- **Algoritmo de las Hormigas** (*Ant Colony Optimization*): Inspiradas en el comportamiento de las hormigas en la transmisión de información al resto de sus compañeras.
- **Colonia de Abejas** (*Bee Colony Optimization*): Co-existen procesos exploradores y explotadores. Los procesos explotadores mejoran las soluciones, explotando las zonas exploradas (por procesos exploradores) que parecen más prometedoras.
- **Enjambre de Partículas** (*Particle Swarm Optimization*): Las soluciones (partículas) se echan a volar en el espacio de búsqueda guiadas por un líder. Cada partícula evoluciona teniendo en cuenta la mejor solución encontrada en su recorrido, su propia inercia y comportamiento de las partículas en su entorno, y al líder de la bandada. Hay muy diversas variantes.

## Metaheurísticas Híbridas:

**GRASP** (Greedy Randomized Adaptive Search Procedure): Constructiva + Mejora

## HEURÍSTICOS

heurística:  $f(n)$

## META-HEURÍSTICOS

Best-First Search

BÚSQUEDA LOCAL

Escalada

Haz

Poca memoria  
No Completos

BÚSQUEDA GLOBAL  
(sistemática)

Voraz

A\*

IDA\*

SMA\*

Completos  
Mucha memoria

### Mejora Metaheurística

Haz, Tabú,

Enfriamiento Simulado,  
(Inteligencia de Enjambre)

### Híbridos: Constructivos + Mejora Local

GRASP

### Evolutivos

#### Algoritmos Genéticos

(Búsqueda dispersa, Alg. Meméticos)

### Métodos Metaheurísticos:

- Combinación estocástica de búsqueda local/ Búsqueda global. **Explotación vs. Exploración.**
- Utilizan poca memoria. Rápidas respuestas iniciales, que continúan optimizando.
- Mejoran la búsqueda local, sin la complejidad espacial/temporal de una búsqueda global.
- Implementación sencilla. *Dificultad: Heurística y Ajuste.*

# No hay una única clasificación...

## a) Heurísticas de Trayectorias:

- Búsqueda tabú
- Búsqueda local guiada
- Métodos multi-arranque
- GRASP
- Enfriamiento simulado, etc.

## c) Inspiradas o no en la Naturaleza

Inspiradas: Genéticos Hormigas, Enfr. Simulado, etc.

No Inspiradas: Búsqueda dispersa, GRASP...

## b) Heurísticas Poblacionales:

- Búsqueda dispersa
- Algoritmos evolutivos, genéticos
- Algoritmos meméticos
- Sistemas de hormigas
- Inteligencia de enjambre, etc.

## d) Aleatorias vs. determinísticas

Aleatorias: Genéticos, evolutivos, etc.

Deterministas: Tabú, Búsqueda Dispersa, etc.

etc.

*Incluso las redes neuronales, la programación por restricciones, etc., son a veces incluidas en el término ‘metaheurística’.*

# Exploración y Explotación de Soluciones

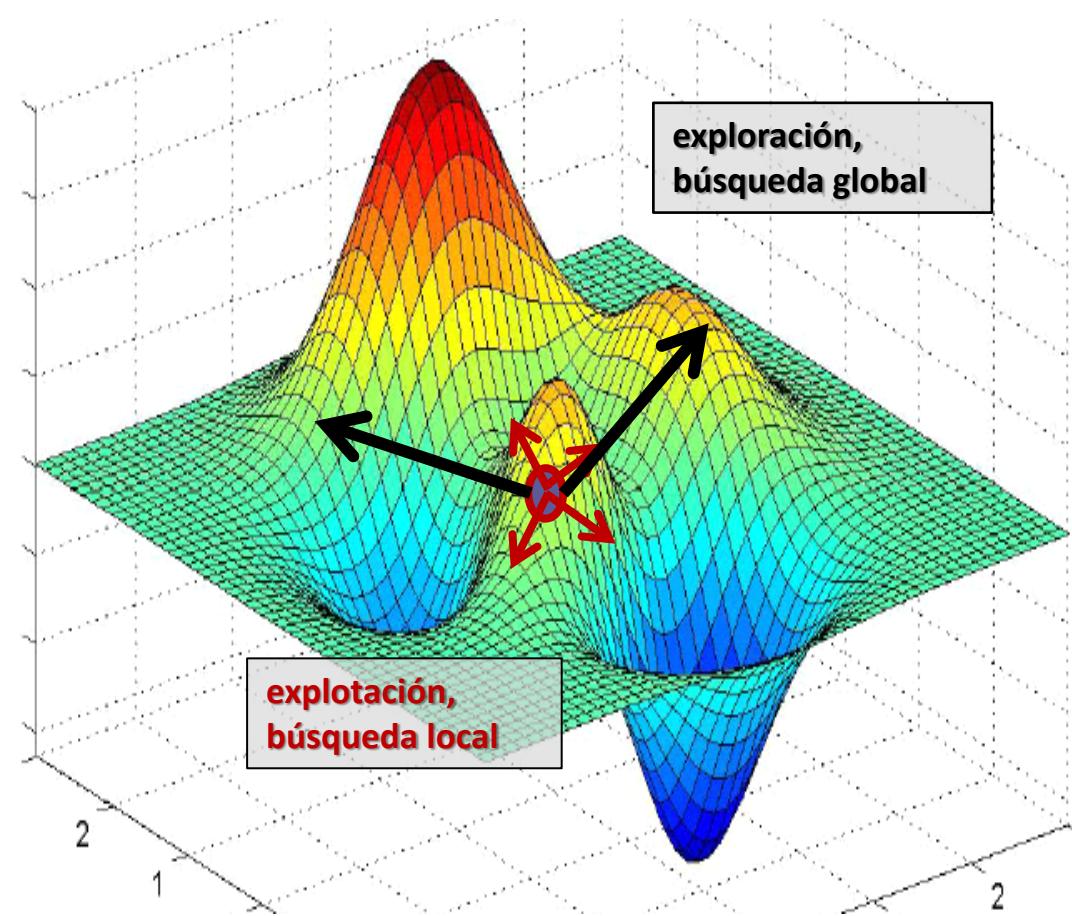
## Exploración de alternativas en el espacio de soluciones.

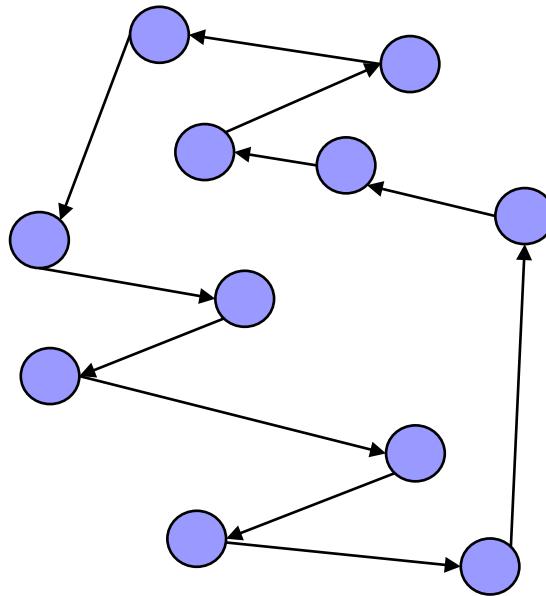
- La exploración busca alternativas un amplio espacio de estados  $\approx$  Búsqueda global.
- Evita convergencia prematura

## Explotación de alternativas vecinas de una buena solución

- Permite focalizar sobre el entorno de buenas soluciones (alto fitness)  $\approx$  Búsqueda local en la vecindad.

*Las metaheurísticas tienden a combinar y/o alternar, con mayor o menor éxito, ambas opciones.*





¿Búsqueda de un mejor vecino?  
*(Explotación)*

¿O mejor empezar desde otro camino?  
*(Exploración)*

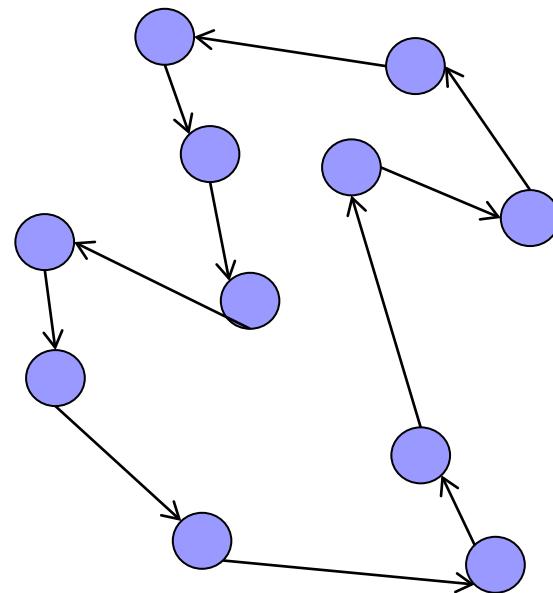
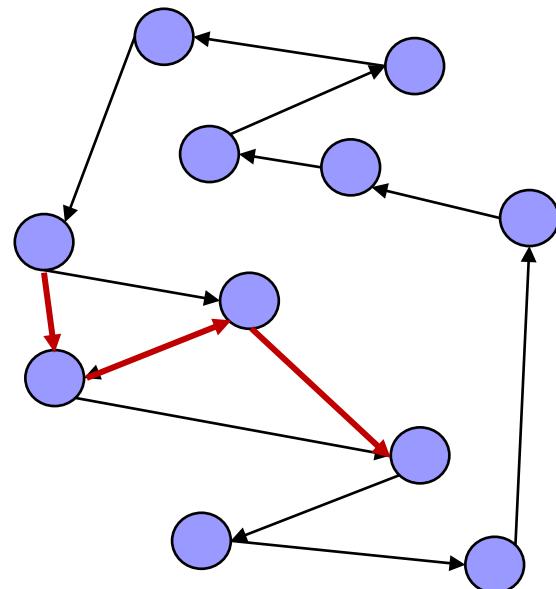
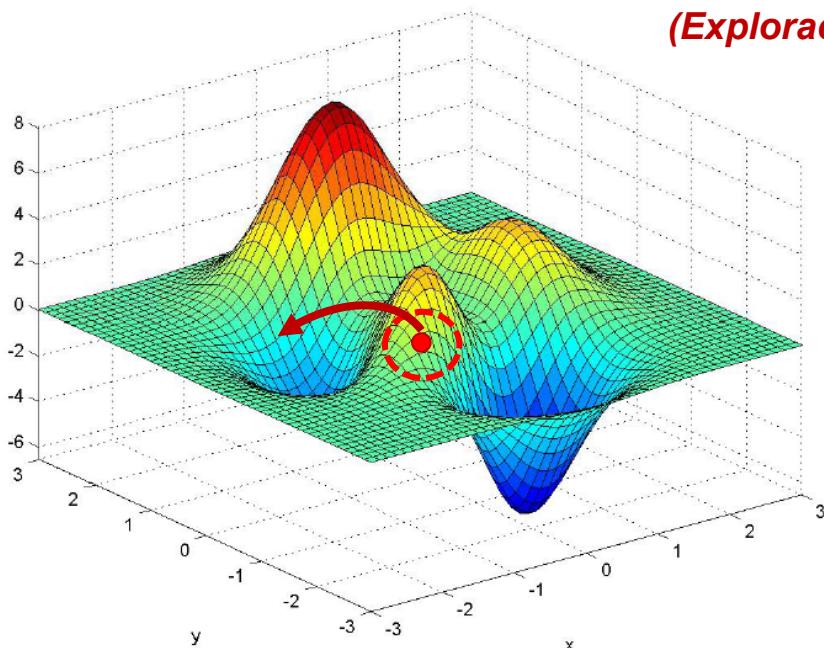


Table 1

Summary of the analysis: techniques and their properties.

Method	Modeling	Anytime	Time	Memory	Tuning	Tool
Chronological backtracking	Coded	Partial	High	Low	No	No
Branch and bound	Coded	Partial	High	Low	No	No
Mixed integer programming	Declarative	No/yes	Low	High	No	GLPK
Constraint logic programming	Declarative	No	High	Low	No	GProlog
Forward checking	Declarative	No	Medium	Medium	No	ConFlex
Synchronous backtracking	Declarative	No/yes	Medium	Medium	No	No
Asynchronous backtracking	Declarative	No	High	High	No	ADOPT
Tabu	Coded	Yes	Low	Medium	Yes	No
GRASP	Coded	Yes	Low	Medium	Yes	No
Combinatorial auctions	Coded	Partial	Medium	High	No	GLPK
Genetic algorithms	Coded	Yes	Low	Low	Yes	No
Ant colony optimization	Coded	No	Low	Medium	Yes	No

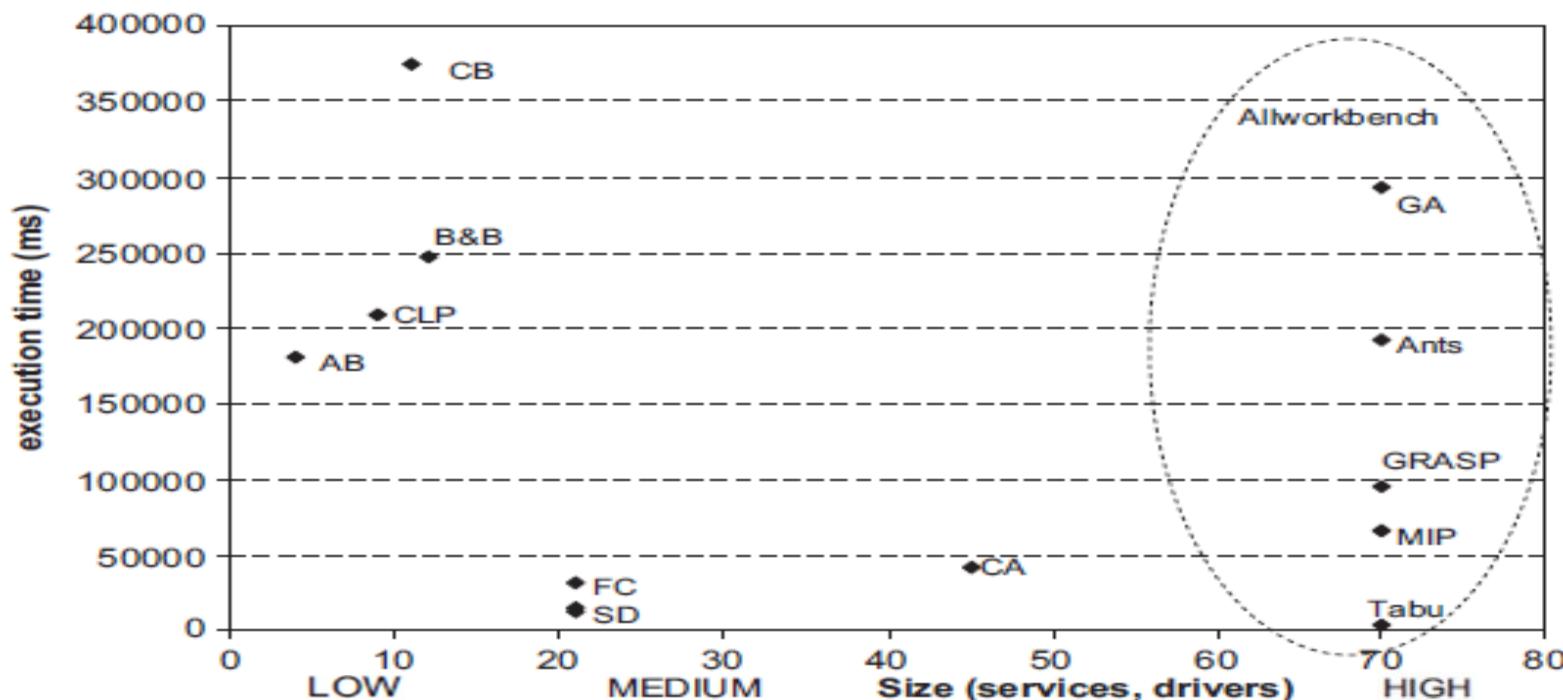


Fig. 2. Maximum problems managed by the methods and the associated execution time.

**Algoritmo de las Hormigas**

**Enfriamiento Simulado**

**Algoritmos Genéticos**

**GRASP**

**Búsqueda Tabú**

**Algoritmo A\***

**Algoritmo de las Abejas**

**Heurística h1(n)**

**Heurística h2(n)**

**Heurística h3(n)**

**¿Cuál es mejor?**

## 'No free lunch' Theorem (Wolpert & Macready, 1997)

- “Todos los algoritmos (de búsqueda o de optimización) tienen la misma evaluación cuando se promedia su aplicación sobre el conjunto de problemas”.

*“Para cada algoritmo de búsqueda/optimización, cualquier superioridad que pudiera tener sobre una clase de problemas es exactamente penalizada sobre las otras clases de problemas”.*

- “En un método de búsqueda, todas las funciones heurísticas (de búsqueda u optimización) tienen la misma evaluación cuando se promedian sus aplicaciones sobre el conjunto de funciones de coste”.

*“La superioridad de un método/heurística, definida sobre una función de coste, es exactamente penalizada cuando se aplica sobre las otras funciones de coste”.*

Por lo tanto, para cualquier algoritmo de búsqueda/optimización, su eventual alto rendimiento sobre una clase concreta de problemas, es exactamente penalizado en su rendimiento sobre otra clase de problemas.

Diversas discusiones relacionadas: <http://www.no-free-lunch.org/>

# Metáfora del Teorema “No Free Lunch” (NFL)

Cada “restaurante” (método/heurística) tiene una carta, que asocia cada “plato” (problema/función de coste) con su “precio” (evaluación del método/heurística en la resolución del problema).

Las cartas de los restaurantes tienen los mismos platos y solo difieren en el precio de los mismos.

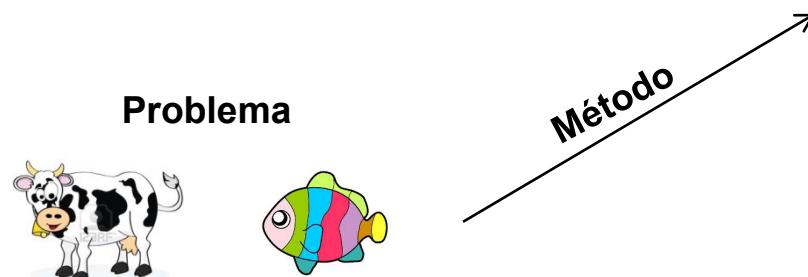
**Restaurante:** Método

**Plato:** Problema / Función de Coste

**Precio:** Evaluación al problema

## Restaurante / Método

Si me gusta un tipo de plato (problema),  
a qué restaurante (método) debo ir?



### Restaur: Algoritmos Genéticos

Carne.1	8.4
Verdura.2	3.2
Pescado.3	7.3

### Restaur: Enfriamiento Simulado

Carne.1	6.8
Verdura.2	4.5
Pescado.3	8.2

### Restaur: Búsqueda Tabú

Carne.1	4.1
Verdura.2	6.4
Pescado.3	9.2

- Para un carnívoro (un tipo de problema), habrá posiblemente un restaurante (método/heurística) más económico (mejor) que otro.

**PERO:**

- Pero si un vegetariano come siempre con un carnívoro, que elige su restaurante más económico, tendrá probablemente que pagar más por su comida.

Es decir, para elegir el mejor restaurante, deben conocerse

los gustos de la persona (problema/función de coste) y

qué cuesta cada tipo de comida en uno u otro restaurante (método/heurística).

*Para una adecuada resolución de un problema se requiere usar información del problema/función de coste (plato) para seleccionar el mejor método/heurística (restaurante).*

**Restaurante: Método/Heurística**

**Plato: Problema / Función de Coste**

**Precio: Evaluación (a minimizar)**

**Carta: Algoritmos Genéticos**

Carne.1	8.4
Verdura.2	3.2
Pescado.3	7.3



**Carta: Enfriamiento Simulado**

Carne.1	6.8
Verdura.2	4.5
Pescado.3	8.2

**Carta: Búsqueda Tabú**

Carne.1	4.1
Verdura.2	6.4
Pescado.3	9.2



# Conclusiones.

- No hay un método/heurística adecuado para todos los problemas, o para todas las funciones de coste aplicables a un problema.
- La bondad de un método de búsqueda/heurística está determinada por su capacidad de captar el conocimiento relevante en el dominio para la mejor resolución del problema concreto.
- El mejor método, en cada caso, es el que permite introducir todo el conocimiento disponible para reducir el espacio de búsqueda y para guiar la búsqueda hacia la mejor solución de acuerdo a la función de coste deseada.

**Incluso más allá...** (D. H. Wolpert, NASA-ARC-05-097)

## LIMITS ON MATH, SCIENCE AND BEYOND

- 1) NFL for supervised learning formalizes Hume:

*Science cannot give guarantees about future experiments based on results of previous experiments.*

- 2) Godel's theorems say math cannot give guarantees about its own conclusions.

- 3) No matter what simulation program it runs, no computer can give guarantees about any future physical experiment.

*More generally, no system - even the universe itself - can give guarantees about prediction, control or observation.*

## IMPOSSIBILITY OF INFERENCE

- *No device can infer itself.*
- *No two distinguishable devices can infer each other*

- 1) *The universe may contain one device that can predict the rest of the universe - but no more than one.*

- 2) *If you have many distinguishable devices, at most one can infer all the others: a God device.*

*I.e., at most one device that can (infallibly) observe / predict / control all distinguishable others: "Monotheism".*

- 3) *A time-translated copy of a God device cannot be a God device.*  
*I.e., God can only be infallible once: "Intelligent design".*