# Tema 1. Heurísticos en Optimización Combinatorial

*Abdelmalik Moujahid, Iñaki Inza y Pedro Larrañaga*

Departamento de Ciencias de la Computación e Inteligencia Artificial

Universidad del País Vasco

http://www.sc.ehu.es/isg/

# Introducción

Formulación general: optimizar $f(\mathbf{x})$ siendo

$$\begin{aligned} f: \quad & D_1 \times D_2 \times \ldots \times D_n & \longrightarrow \mathbb{R} \\ & \mathbf{x} = (x_1, x_2, \ldots, x_n) & \longrightarrow f(\mathbf{x}) \end{aligned}$$

$f(\mathbf{x})$ sujeta a un conjunto de restricciones:

$$\begin{aligned} h_i(\mathbf{x}) &< b_i \qquad & i &= 1, \ldots, l \\ h_i(\mathbf{x}) &> b_i \qquad & i &= l+1, \ldots, m \\ h_i(\mathbf{x}) &= b_i \qquad & i &= m+1, \ldots, r \end{aligned}$$

# Introducción

- En optimización combinatorial
  - $D_i \subset \mathbb{N} (i = 1, \ldots, n)$
  - Por tanto el espacio de soluciones está formado por permutaciones o por productos cartesianos de subconjuntos de $\mathbb{N}$
- Ejemplos de problemas de optimización combinatorial:
  - Problema del viajante de comercio
  - Problema de la mochila

# Heurísticos

- *Heuriskein*: encontrar, descubrir (Arquímides)
- Polya (1957) lo utiliza en *How to Solve It*
- Simon (1963) define heurístico como

  un proceso que puede resolver un problema dado pero no ofrece garantías de hacerlo

- Hoy en día:

  procedimiento simple, a menudo basado en el sentido común, que se supone que va a ofrecer una buena solución (no necesariamente la óptima) de un modo fácil y rápido a problemas difíciles

# Heurísticos

- Justificación de uso: problemas NP para los cuales no existe un algoritmo de resolución polinomial con el tamaño del problema
- Ventajas:
  - Flexibilidad frente al encorsetamiento de la investigación operativa clásica
  - Más comprensibles e intuitivos (en general) que los métodos exactos
- Inconvenientes:
  - Imposible conocer la cercanía con respecto del óptimo global de la solución obtenida

# Heurísticos

Condiciones en las que se aconseja su uso:

- No existe un método exacto de resolución, o el mismo requiere de mucho gasto computacional y/o de memoria
- No es necesario encontrar la solución óptima; basta con una solución suficientemente buena
- Los datos son poco fiables y por tanto no tiene sentido el tratar de encontrar el óptimo global para dichos datos
- Existen limitaciones de tiempo (y/o de memoria) en proporcionar la respuesta
- Se va a utilizar como solución inicial para un algoritmo exacto de tipo iterativo

# Heurísticos

3 tipos de búsquedas heurísticas

- *Heurísticas constructivas* añaden componentes individuales a la solución inicial hasta que se obtiene una solución final factible,
- *Heurísticas basadas en la mejora de una solución* parten de una solución para en cada paso buscar en la vecindad de la misma una solución mejor,
- *Heurísticas basadas en poblaciones* trabajan con poblaciones de individuos que evolucionan iterativamente.

# El problema del viajante de comercio

- Dadas $n$ ciudades y conocido el coste de trasladarse de una ciudad a otra, se trata de encontrar la gira que visitando todas las ciudades, una y sólo una vez, tenga asociada un coste mínimo
- Referenciado por vez primera en 1932 como TSP (*Travelling Salesman Problem*)
- Buen número de problemas de optimización en Inteligencia Artificial se relacionan con la búsqueda de la permutación óptima

# El problema del viajante de comercio

Dadas $n$ ciudades y una matriz de costos cuyos elementos se denotan por $c_{ij}$ (con $i, j = 1, \ldots, n$), se trata de encontrar la permutación $\pi^* = (\pi^*(1), \ldots, \pi^*(n))$ con menor costo asociado

Es decir:

$$\pi^* = (\pi^*(1), \ldots, \pi^*(n)) = argmin_\pi \sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)} + c_{\pi(n)\pi(1)}$$

# El problema del viajante de comercio

Ejemplo de matriz de costos, correspondiente a un TSP con $n = 6$ ciudades:

$$\begin{pmatrix} 0 & 6 & 2 & 1 & 4 & 10 \\ 6 & 0 & 3 & 4 & 3 & 1 \\ 2 & 3 & 0 & 2 & 8 & 3 \\ 1 & 4 & 2 & 0 & 5 & 6 \\ 4 & 3 & 8 & 5 & 0 & 9 \\ 10 & 1 & 3 & 6 & 9 & 0 \end{pmatrix}$$

# El problema del viajante de comercio

1. Escoger una ciudad al azar como ciudad de partida
2. Mientras queden ciudades por visitar, escoger -de entre las no visitadas- la que se encuentre a menor distancia de la última visitada
3. Mostrar la solución

Pseudocódigo de un algoritmo heurístico constructivo voraz para el TSP

# El problema del viajante de comercio

El heurístico constructivo voraz aplicado a la matriz de costes anterior

$$\begin{pmatrix} 0 & 6 & 2 & 1 & 4 & 10 \\ 6 & 0 & 3 & 4 & 3 & 1 \\ 2 & 3 & 0 & 2 & 8 & 3 \\ 1 & 4 & 2 & 0 & 5 & 6 \\ 4 & 3 & 8 & 5 & 0 & 9 \\ 10 & 1 & 3 & 6 & 9 & 0 \end{pmatrix}$$

proporciona la solución (se supone que la ciudad 4 es seleccionada como ciudad de partida):

$$4 - 1 - 3 - 2 - 6 - 5$$

El costo asociado a dicha gira es:

$$c_{41} + c_{13} + c_{32} + c_{26} + c_{65} + c_{54} = 1 + 2 + 3 + 1 + 9 + 5 = 21$$

# El problema del viajante de comercio

- Con el heurístico anterior:
  - No hay garantía de que se alcance el óptimo global
  - El algoritmo es determinista
- *Versión estocástica* del algoritmo heurístico constructivo voraz: de entre las no visitadas seleccionar con una probabilidad inversamente proporcional a la distancia con respecto a la última ciudad visitada

# El problema del viajante de comercio

1. Seleccionar al azar una permutación de ciudades $\pi$
2. Repetir

   Obtener la permutación $\pi'$ vecina de $\pi$

   con menor costo

   Hacer $\pi := \pi'$

   Hasta que el costo de ninguna permutación vecina sea menor que el costo de la solución actual

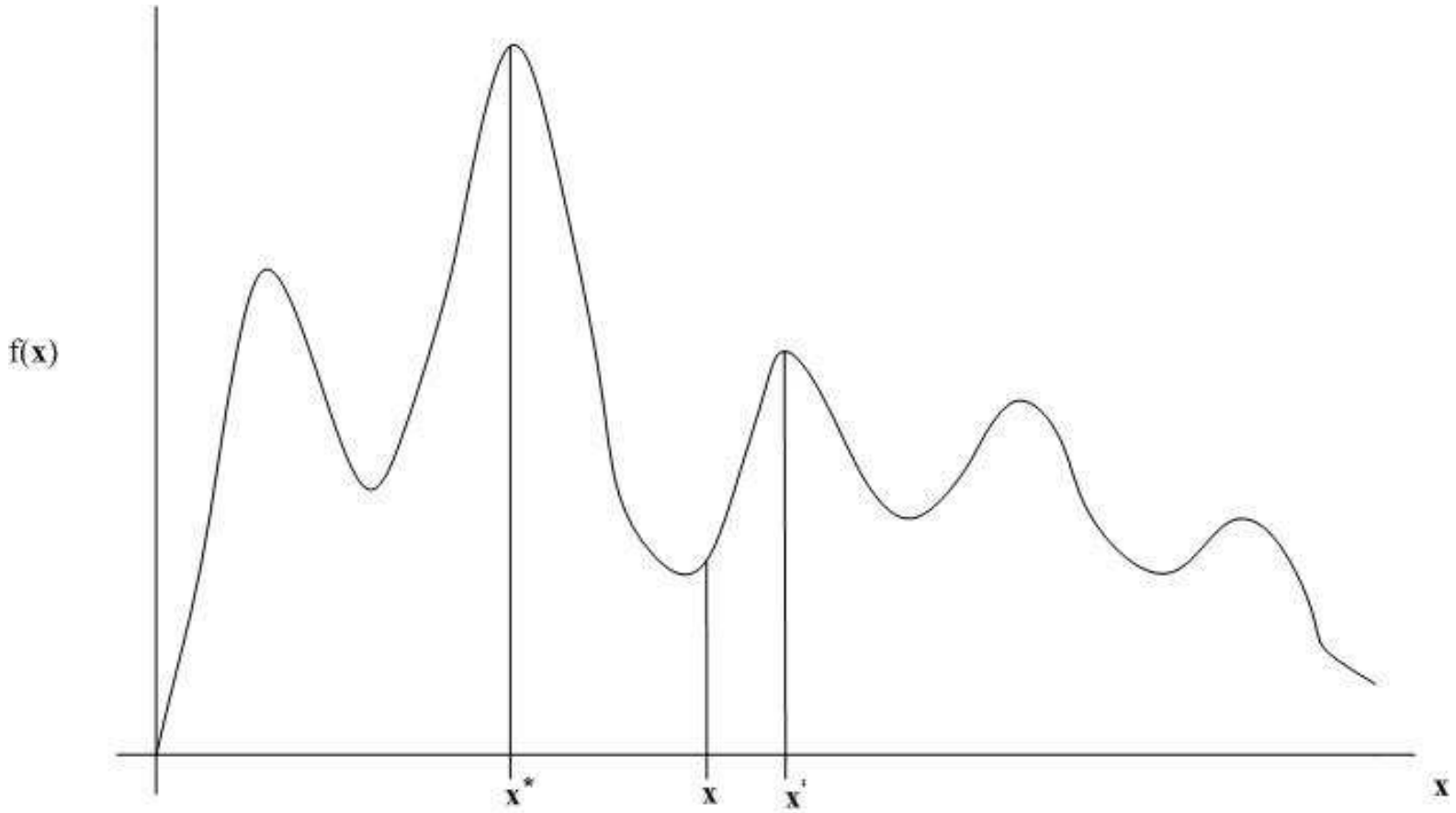
Algoritmo de mejora iterativa de una solución para el TSP. Ascenso (descenso) por la colina (*hillclimbing*)

# El problema del viajante de comercio

Variantes del algoritmo de mejora iterativa

- *Versión estocástica* permitiendo saltar a una solución vecina de la actual, escogida al azar, sin que necesariamente dicha solución constituya la mejor de todas las soluciones pertenecientes a la vecindad de la solución actual
- *Threshold accepting* acepta saltos a soluciones vecinas cuya evaluación supere el $\alpha$ por ciento (por ejemplo, el 95 %) de la evaluación de la actual solución
- O saltar a soluciones que estando en la vecindad de la solución actual tengan peor función objetivo

# El problema del viajante de comercio




Influencia de la vecindad en el heurístico de mejora iterativa de una solución

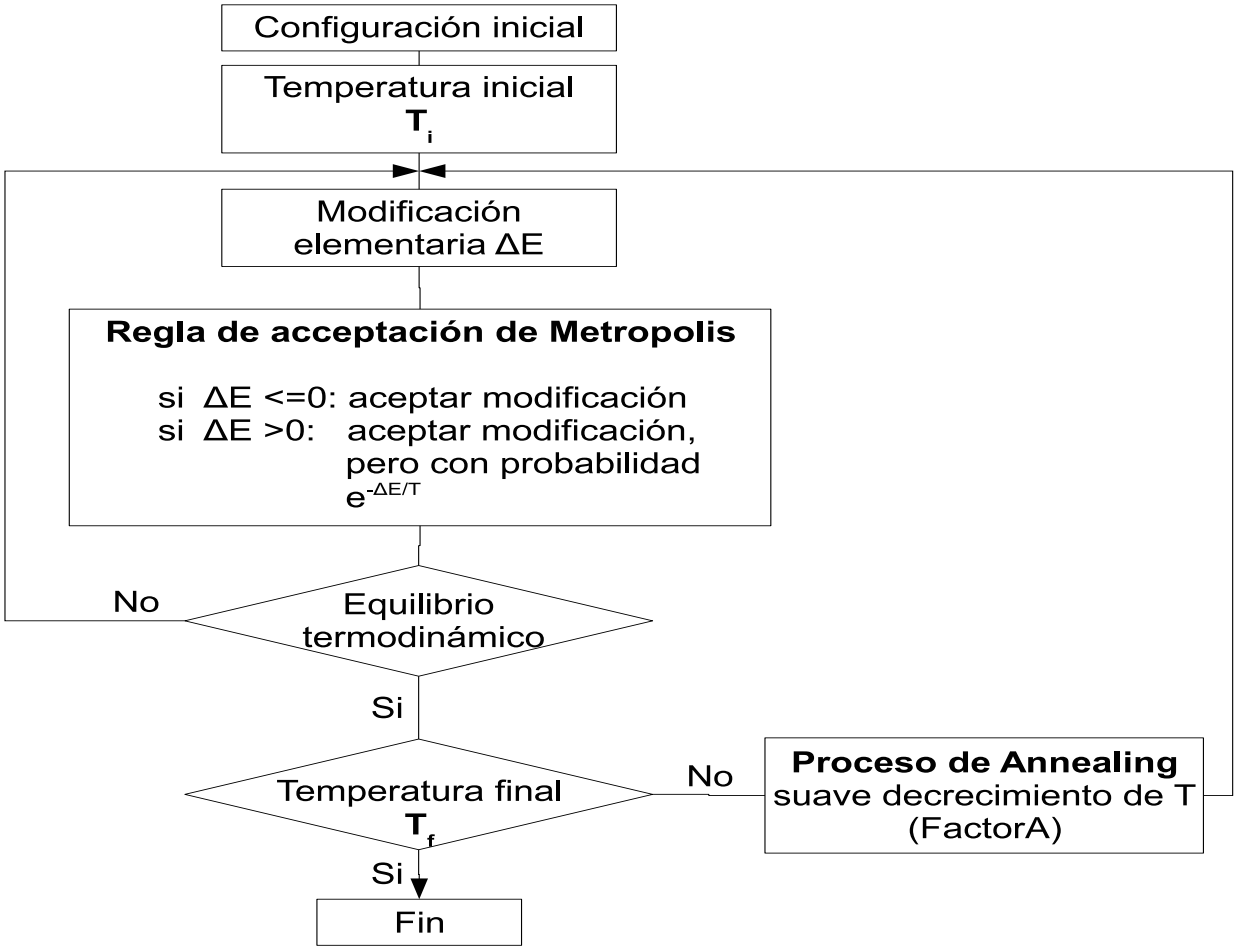
# Metaheurísticos

Heurísticos genéricos e independientes del problema que tratan de optimizar

| Metaheurístico | Estocástico | Población |
|---|---|---|
| Simulated Annealing | sí | no |
| Algoritmos Genéticos | sí | sí |
| Algoritmos de Estimación de Distribuciones | sí | sí |
| Estrategias Evolutivas | sí | sí |
| Programación Genética | sí | sí |
| Vecindad Variable | no | no |
| Búsqueda Tabú | no | no |

Características de metaheurísticos

# Simulated Annealing



# Simulated Annealing

- Para temperaturas altas, el término $e^{\frac{-\triangle E}{T}} \longrightarrow 1$, implicando la aceptación de la mayoría de las modificaciones. En este caso el algoritmo se comporta como un paseo aleatorio simple en el espacio de búsqueda.
- Para temperaturas bajas, el término $e^{\frac{-\triangle E}{T}} \longrightarrow 0$, implicando el rechazo de la mayoría de las modificaciones. En este caso el algoritmo se comporta como una mejora iterativa clásica.
- Para valores intermedios de la temperatura, el algoritmo autoriza de manera intermitente movimientos que degradan la función objetivo.

# Simulated Annealing

## Sugerencias prácticas

- Temperatura inicial $T_0$: se puede calcular como un paso preliminar de la siguiente manera:
  -aplicar 100 modificaciones aleatoriamente; evaluar el promedio $< \Delta E >$ de las variaciones $\Delta E$ correspondientes;
  -elegir un ratio inicial de aceptación $\tau_0$ para las modificaciones degradantes; por ejemplo: $\tau_0 = 50\,\%$ (empezando a altas temperaturas);
  -deducir $T_0$ de la relación $e^{\frac{-<\Delta E>}{T_0}} = \tau_0$.
- Regla de aceptación de Metropolis: si $\Delta E > 0$ , se genera un número aleatorio $r$ entre $[0, 1]$, se acepta la modificación si $r < e^{\frac{-\Delta E}{T}}$ , donde $T$ es la temperatura actual.

# Simulated Annealing

## Sugerencias prácticas

- Cambio de la temperatura: se procede a un cambio de la temperatura si se cupmle alguna de las condiciones siguientes:
  -$12.N$ modificaciones aceptadas;
  -$100.N$ modificaciones intentadas, N indica el número de grados de libertad (o parámetros) del problema.
- Decrecimiento de la temperatura: el decrecimiento de la temperatura puede llevarse a cabo según la siguiente regla: $T_{k+1} = 0{,}9.T_k$
- Finalizar el programa: se puede finalizar el programa tras 3 cambios sucesivos de la temperatura sin nigúna aceptación.