

Tema 1. Introducción a las Metaheurísticas

Profesor: Sebastián Ventura Soto

Objetivos

- Conocer problemas P y NP
- Entender el concepto de metaheurística
- Conocer los elementos más importantes de una metaheurística
- Conocer el teorema del *No free lunch*

Motivación

- Muchos problemas de optimización son complejos y difíciles de resolver
 - No pueden ser resueltos de forma exacta en un tiempo razonable
 - Requieren alternativas: algoritmos aproximados
 - **Heurísticas:** dependientes del problema
 - **Metaheurísticas:** más generales y aplicables a gran variedad de problemas
 - Rapidez en la resolución de problemas
 - Resolución de problemas más complejos
 - Robustez de los algoritmos
 - Metáforas naturales (evolución de especies, procesos físicos como el enfriamiento de partículas, sociedades de insectos, etc).

Contenidos del tema

- Optimización de problemas
- Algoritmos de búsqueda
- Algoritmos aproximados
- *No free lunch*

Optimización de problemas

- Teoría de la complejidad
 - **Cómo crece el coste computacional** de resolver un determinado problema en relación a lo que crece el tamaño de dicho problema
Ordenar 1.000 números cuesta más que ordenar 100 ¿Cuánto más?
Complejidad lineal (función para calcular el tiempo es lineal)

Números:	1.000	10.000	100.000	1.000.000
Tiempo:	1	10	100	1.000
 - Problema **no es lo mismo** que algoritmo
 - El problema es el mismo y habrá algoritmos que lo hagan mejor y otros peor
 - Ordenación: Heapsort (complejidad $n \log(n)$) Vs Burbuja (n^2)
 - **Complejidad del problema** es la del **mejor algoritmo** que lo resuelva

Optimización de problemas

- En teoría computacional
 - P: conjunto de problemas en los que podemos encontrar una solución en un tiempo razonable (polinomial)
 - NP: conjunto de problemas en los que podemos comprobar en un tiempo razonable (polinomial) si una respuesta al problema es o no solución
 - Todos los problemas que están en P están también en NP
 - Si encontramos una solución en tiempo razonable también encontraremos en un tiempo razonable si es respuesta correcta o no
 - No está tan claro que un problema que esté en NP esté también en P
 - Comprobar una respuesta en un tiempo razonable no implica que la respuesta sea encontrada en un tiempo razonable

Optimización de problemas

n números positivos y negativos: +1, +7, +9, -4, -3, -2, -5, -1, -8

¿Puedes coger unos cuantos de forma que sumen 0?

- Si me dan una respuesta, puedo comprobar fácilmente si es solución
- Encontrar ese conjunto no es tan fácil
- **Problema P=NP** se pregunta si estos dos conjuntos son iguales o no, es decir, si para cualquier conjunto en el que podemos comprobar fácilmente una solución podemos encontrar fácilmente una solución
- 1 MILLÓN DE DÓLARES RESOLVERLO!!!

Optimización de problemas

- Complejidad cuando cada paso requiere un microsegundo

	n=10	n=100	n=1.000	n=10.000
Constante	1	1	1	1
Logarítmica ($\log n$)	1	2	3	4
Lineal (n)	10	100	1.000	10.000
Polinomial (n^2)	100	10.000	1 segundo	100 segundos
Exponencial (2^n)	1.024	4,02 x 10 ¹⁶ años

Algoritmos de búsqueda

- Existen problemas reales (de optimización o búsqueda) de difícil solución que requieren tareas como:
 - Encontrar el camino más corto entre varios puntos
 - Establecer un plan de mínimo coste para repartir mercancías
 - Asignar de manera óptima tareas a los trabajadores
 - Establecer una secuencia óptima de procesos en cadenas de producción
 - etc.
- Estos problemas tienen como características:
 - Son NP-hard
 - Los algoritmos exactos son ineficientes o imposibles de aplicar

Algoritmos de búsqueda

- **Problemas NP**
 - Problema de la mochila (*knapsack problem*)
 - Viajante de comercio (TSP)
 - Optimización de parámetros
 - Controlador de SuperMario Bros
 - Distribución de carga de trabajo
 - etc.
- **Función objetivo.** Maximizar (o minimizar) una función
- **Espacio de búsqueda.** Valores de las variables que serán evaluados

Algoritmos de búsqueda

- **Viajante de comercio**

- Se tiene una red de nodos, que pueden ser ciudades o lugares dentro de una misma ciudad
- Se sabe la distancia entre cada uno de los nodos
- Se parte de un lugar inicial, y se tiene que recorrer todos los nodos sin pasar más de una vez por cada nodo, volviendo al lugar inicial

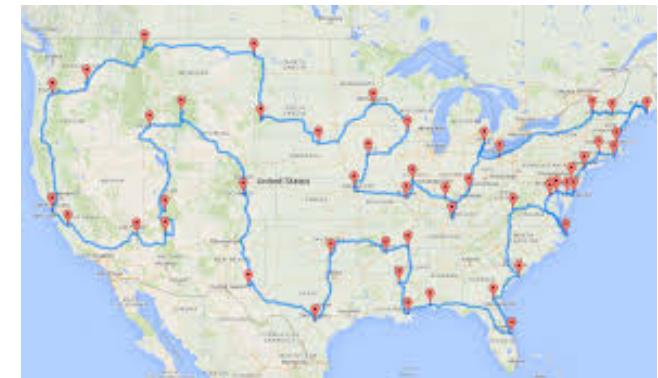
- Problema

- 4 ciudades: 6 posibles recorridos ($0.1 \text{ seg}/\text{evaluación} = 0.6 \text{ segundos}$)
- 50 ciudades: 3×10^{64} posibles recorridos ($0.1 \text{ seg}/\text{evaluación} = 9.6 \times 10^{51} \text{ años}$)

Algoritmos de búsqueda

- **Viajante de comercio**

- Esquema de representación
 - Cada ciudad (nodo) se enumera con un número
 - Permutación de {1, 2, ..., n}
 - Cada solución es un camino diferente (3 1 5 2 4 6)



- Función objetivo

$$\text{Min } C(S) = \sum_{i=1}^{n-1} (D[S[i], S[i+1]]) + D[S[n], S[1]]$$

Algoritmos de búsqueda

- **Problema de la mochila**
 - Se dispone de una mochila y un conjunto de objetos
 - Cada objeto tiene un peso y un valor
 - Se desea llenar la mochila con aquellos objetos que produzcan un mayor beneficio sin sobrepasar el peso máximo
- Problema
 - 4 objetos: 16 soluciones (tanto válidas como inválidas)
 - 50 objetos: 1.13×10^{15} soluciones (tanto válidas como inválidas)

Algoritmos de búsqueda

- **Problema de la mochila**

- Esquema de representación

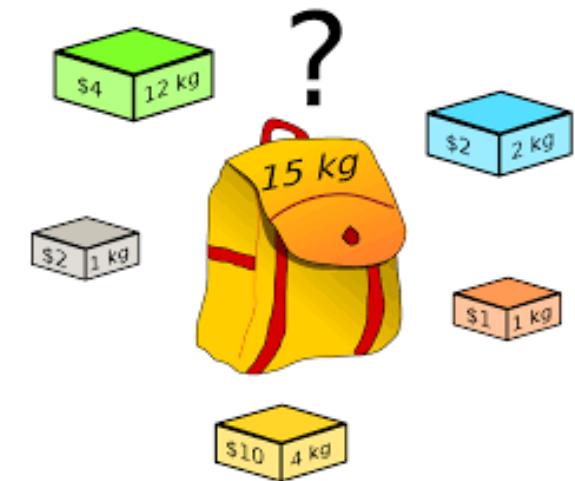
- Se dispone de n objetos, cada objeto X uno con
 - Un peso W
 - Un valor P
 - Cada objeto puede estar o no en la mochila
 - Cada solución X_i es una secuencia binaria (1 0 0 1 1 0)
 - Se tiene un peso máximo M de mochila

- Función objetivo

- Maximizar

- Restricción

$$\begin{array}{|c|}\hline \sum_{i=1}^n P_i X_i \\ \hline \end{array}$$
$$\begin{array}{|c|}\hline \sum_{i=1}^n W_i X_i \leq M \\ \hline \end{array}$$



Algoritmos de búsqueda

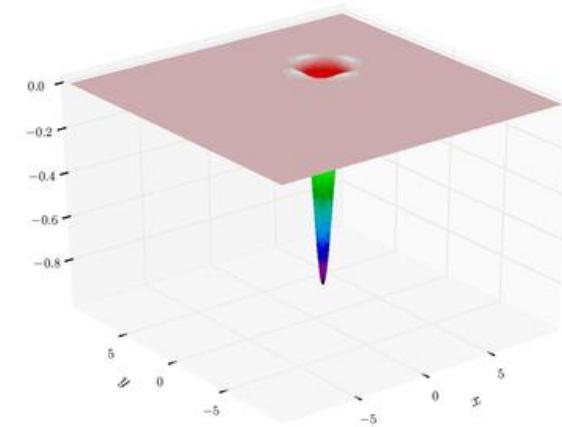
- **Problema de la mochila (elección múltiple)**
 - Se dispone de una mochila y un conjunto de productos
 - Cada producto tiene un peso y un valor
 - Se desea llenar la mochila con aquellos productos (0 o m) que produzcan un mayor beneficio sin sobrepasar el peso máximo
- Esquema de representación
 - Cada solución X_i es un vector discreto (1 0 4 1 5 2)
- Función objetivo
 - Maximizar $\sum_{i=1}^n P_i X_i$
 - Restricción $\sum_{i=1}^n W_i X_i \leq M$

Algoritmos de búsqueda

- **Optimización de funciones (Easom function)**
 - Se tiene una función y se desea minimizar, es decir obtener el menor valor de dicha función
 - Esquema de representación como par (x, y)
 - Ejemplo: (2.45 1.01)
 - Función a minimizar
 - Solución

$$f(x, y) = -\cos(x) \cos(y) \exp\left(-\left((x - \pi)^2 + (y - \pi)^2\right)\right)$$

$$f(\pi, \pi) = -1$$



Algoritmos de búsqueda

- **Tipos de codificación**

- Binaria (1 0 0 1 0 1)
 - Problema de la mochila
- Entera (1 0 4 1 5 2)
 - Problema de la mochila con elección múltiple
- Ordinal (3 1 5 2 4 6)
 - Viajante de comercio
- Real (2.45 1.01)
 - Optimización de funciones

Algoritmos aproximados

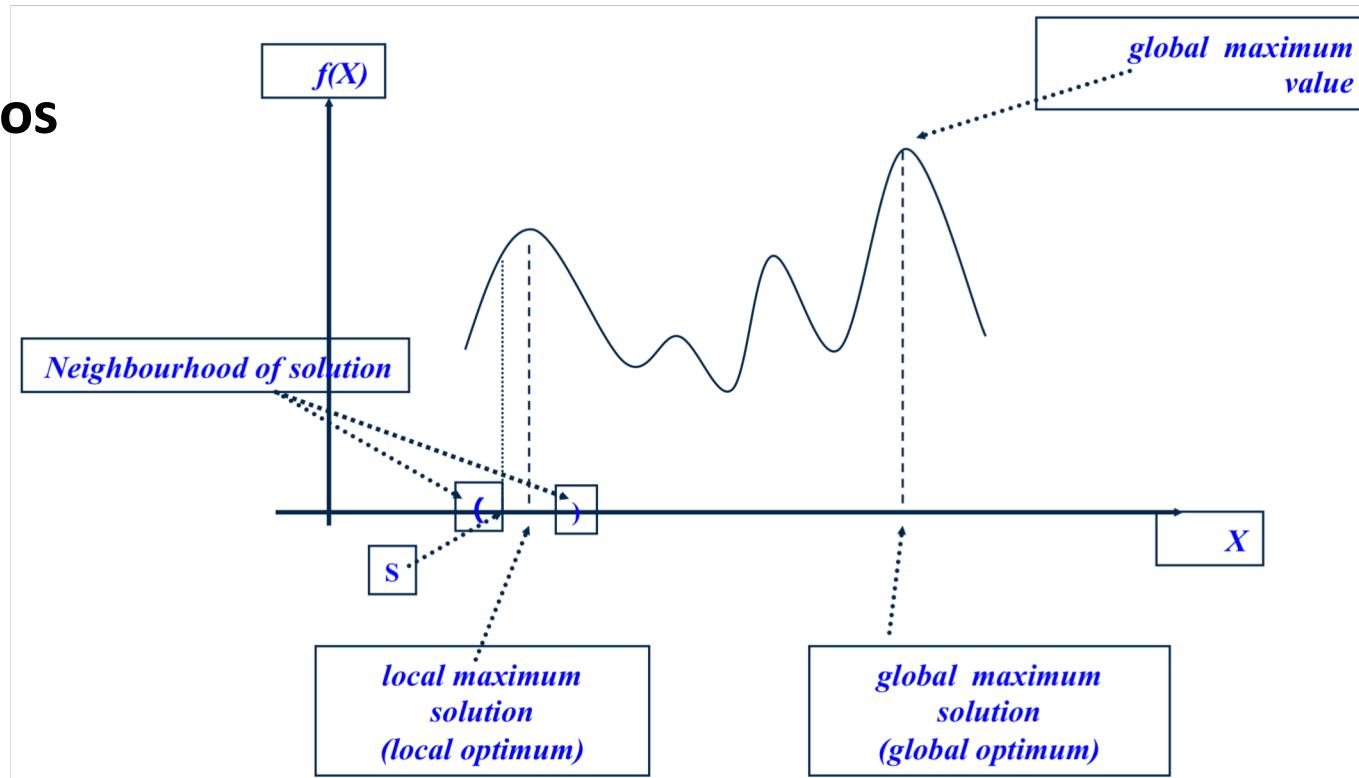
- Los **algoritmos aproximados** aportan soluciones cercanas a la óptima en problemas complejos (NP) en un tiempo razonable. Uso cuando:
 - No existe un método exacto de resolución
 - Existe un método exacto pero requiere mucho tiempo de cálculo
 - No se necesita la solución óptima sino que basta con una buena en un tiempo aceptable
- **Heurísticas:** dependientes del problema
- **Metaheurísticas:** más generales y aplicables a gran variedad de problemas
 - Rapidez en la resolución de problemas más complejos
 - Robustez de los algoritmos
 - Metáforas naturales (evolución de especies, procesos físicos como el enfriamiento de partículas, sociedades de insectos, etc).

Algoritmos aproximados

- **Búsqueda** se usa para construir/mejorar soluciones y obtener el óptimo o soluciones casi-óptimas
- Elementos de un algoritmo aproximado:
 - **Solución:** representación de la solución al problema
 - **Entorno:** soluciones cercanas
 - **Movimiento:** transformación de la solución actual en otra
 - **Evaluación:** factibilidad de la solución y función objetivo

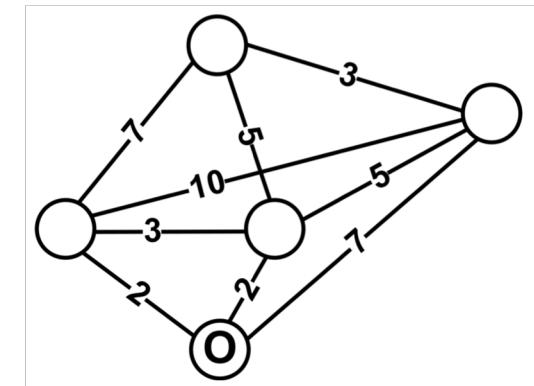
Algoritmos aproximados

Elementos



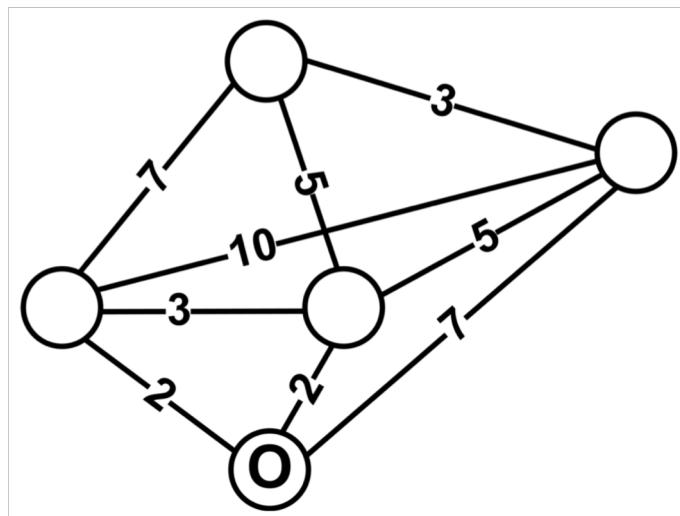
Algoritmos aproximados

- **Heurística Greedy.** Tratar de construir una solución eligiendo de manera iterativa los elementos de menor costo
- La solución encontrada no tiene por qué ser la óptima
- Ejemplo de aplicación al **viajante de comercio**



Algoritmos aproximados

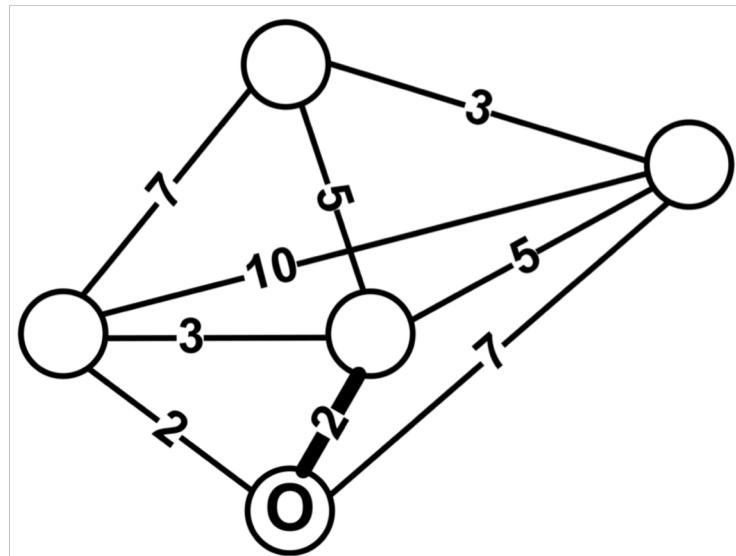
- **Heurística Greedy.** Elijo la de menor costo (ciudad más próxima) entre las no visitadas



S. Ventura - J. M. Luna

Algoritmos aproximados

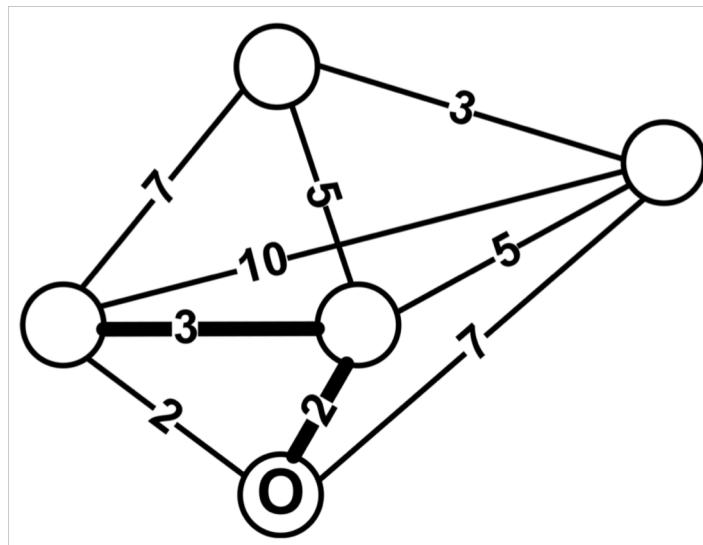
- **Heurística Greedy.** Elijo la de menor costo (ciudad más próxima) entre las no visitadas



S. Ventura - J. M. Luna

Algoritmos aproximados

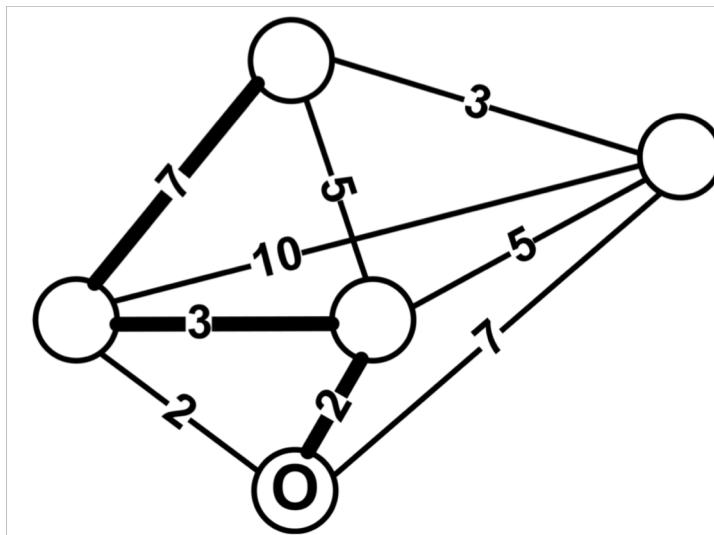
- **Heurística Greedy.** Elijo la de menor costo (ciudad más próxima) entre las no visitadas



S. Ventura - J. M. Luna

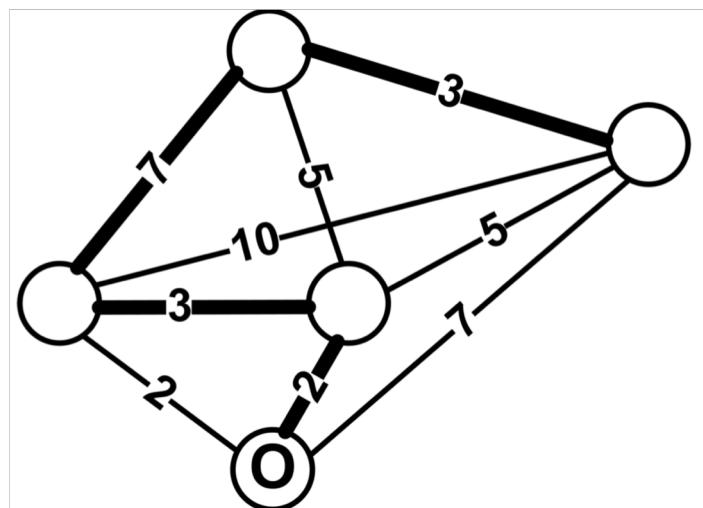
Algoritmos aproximados

- **Heurística Greedy.** Elijo la de menor costo (ciudad más próxima) entre las no visitadas



Algoritmos aproximados

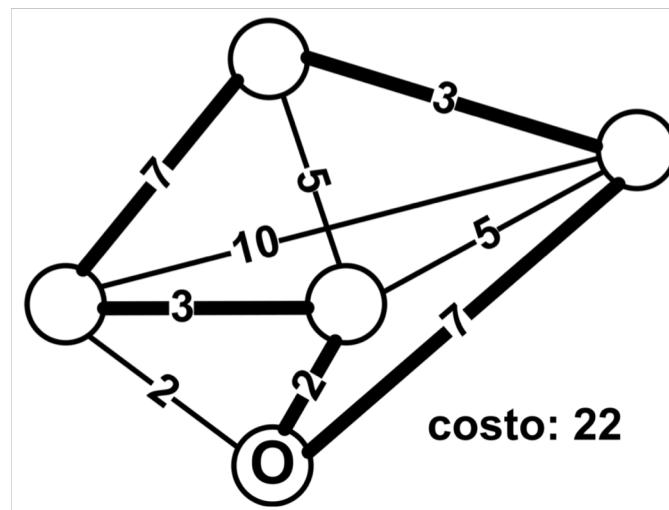
- **Heurística Greedy.** Elijo la de menor costo (ciudad más próxima) entre las no visitadas



S. Ventura - J. M. Luna

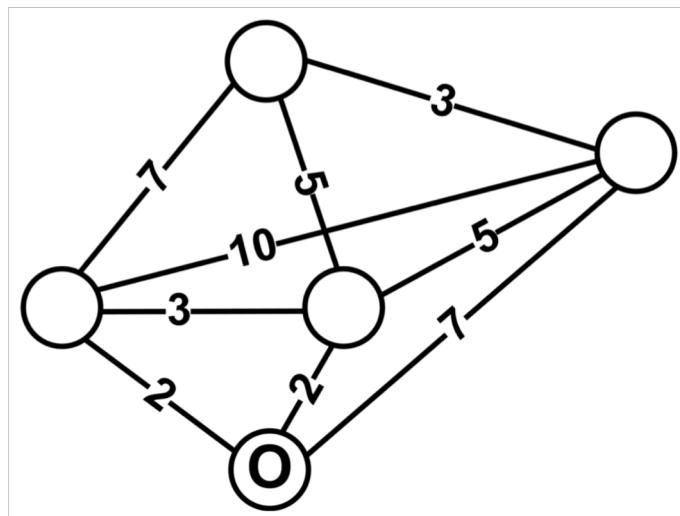
Algoritmos aproximados

- **Heurística Greedy.** Elijo la de menor costo (ciudad más próxima) entre las no visitadas



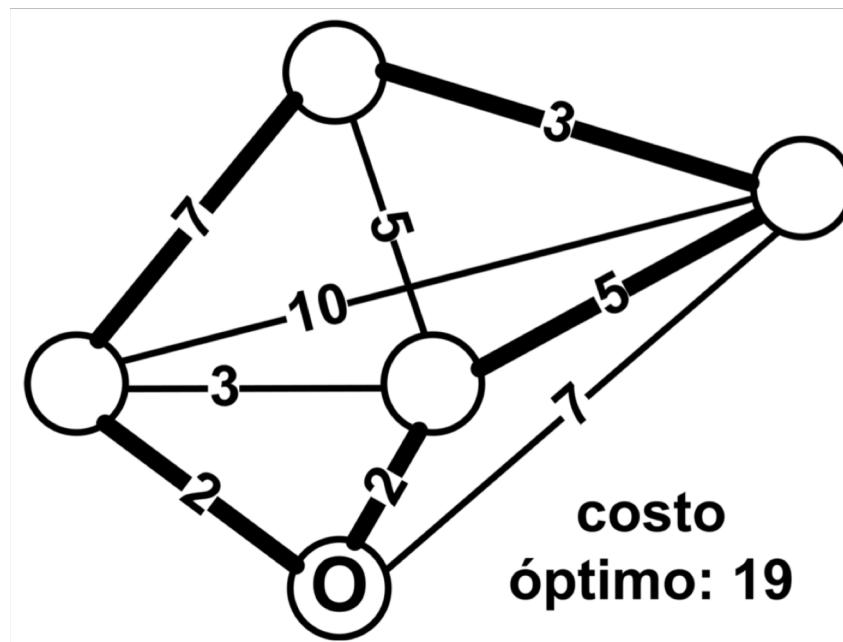
Algoritmos aproximados

- ¿¿¿Es el **costo mínimo**??? Determinar el **costo óptimo**



Algoritmos aproximados

- Costo óptimo 19



Algoritmos aproximados

- **Metaheurísticas:**

- Familia de algoritmos aproximados
- Suelen ser procedimientos iterativos
- Guían una heurística subordinada de búsqueda
- Combina diversificación (exploración global) con intensificación (explor. local)

Ventajas

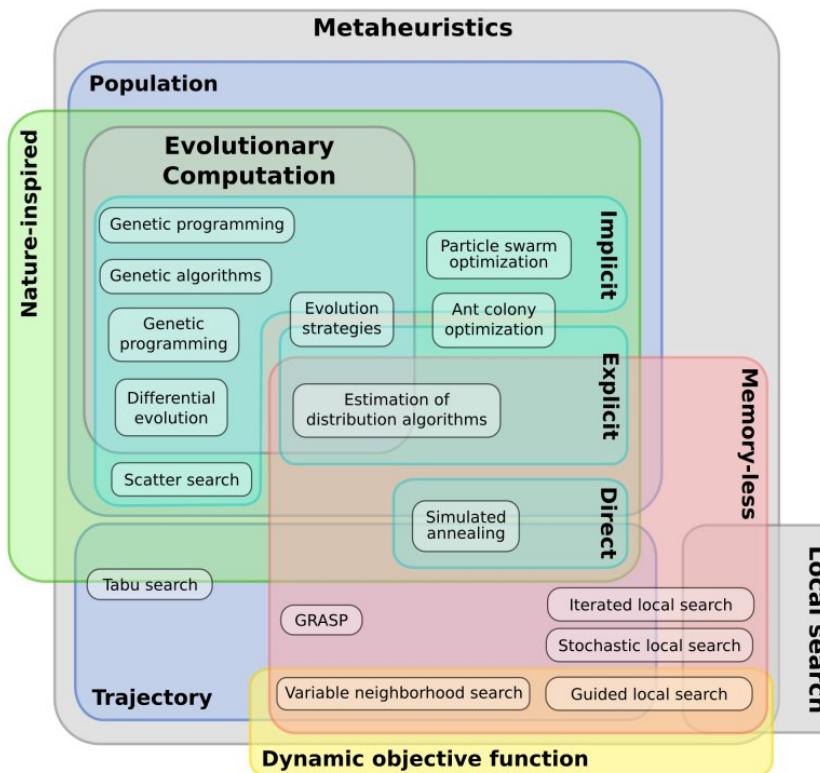
- Algoritmos de propósito general
- Gran éxito
- Fácil de implementar
- Fácil de paralelizar

Inconvenientes

- Aproximados, no exactos
- Estocásticos
- Sin base teórica establecida

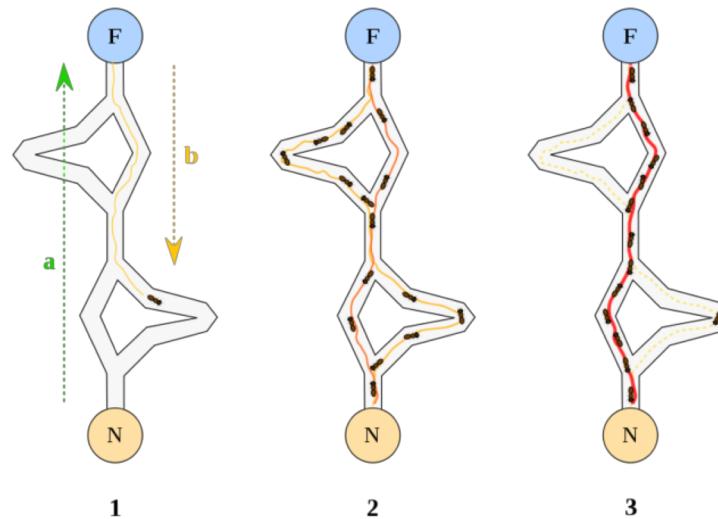
Algoritmos aproximados

Tipos



Algoritmos aproximados

Metaheurísticas basadas en la naturaleza



Algoritmos aproximados

Búsqueda aleatoria

- ¿Converge al óptimo?
- ¿Rápido en encontrar el óptimo?
- Permite abordar cualquier problema
- No está sujeto a ningún supuesto

$k \leftarrow 1$

Mientras que $k \leq N$

 Generar una solución aleatoria e_k

 Calcular $h_k f(e_k)$

 Si $h_k < h_{\text{optimo}}$, entonces

$e_{\text{optimo}} \leftarrow e_k$
 $h_{\text{optimo}} \leftarrow h_k$

 Fin Si

$k \leftarrow k + 1$

Fin Mientras

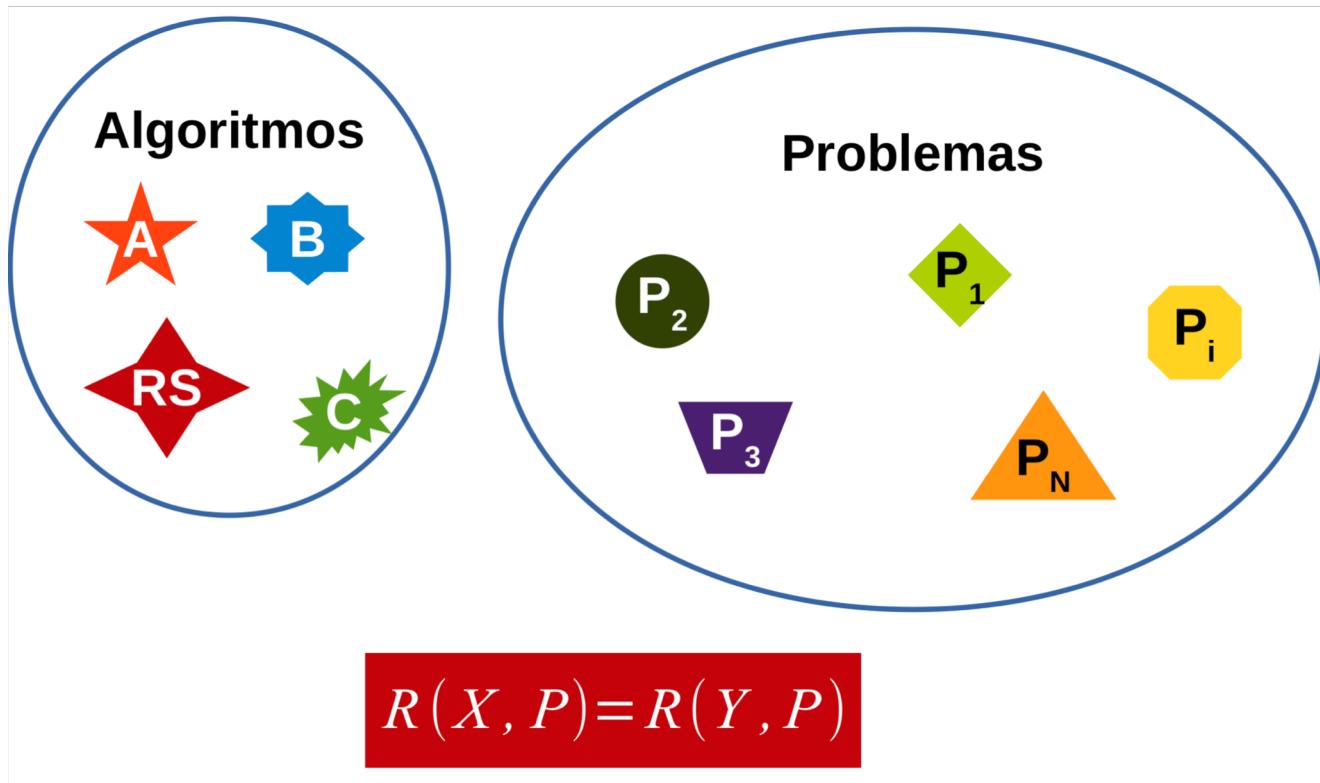
Algoritmos aproximados

Búsqueda aleatoria

- α probabilidad de encontrar la solución óptima
- M conjunto de soluciones
- L ejemplos necesarios

α	M	L	α	M	L
0.9	1000	2302	0.9	3000	6907
0.8	1000	1609	0.8	3000	4828
0.7	1000	1203	0.7	3000	3612
0.6	1000	916	0.6	3000	2748
0.9	2000	4605	0.9	4000	9210
0.8	2000	3219	0.8	4000	6437
0.7	2000	2408	0.7	4000	4816
0.6	2000	1833	0.6	4000	3664

No free lunch



No free lunch

