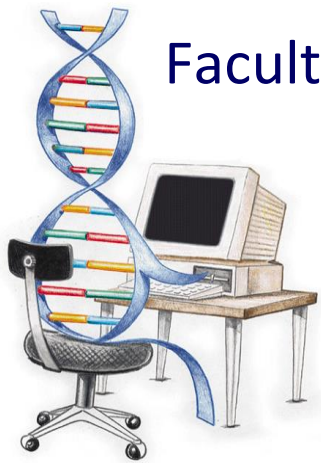


# ALGORITMOS EVOLUTIVOS

## Curso 2024

### Tema 1: Introducción

Centro de Cálculo, Instituto de Computación  
Facultad de Ingeniería, Universidad de la República, Uruguay

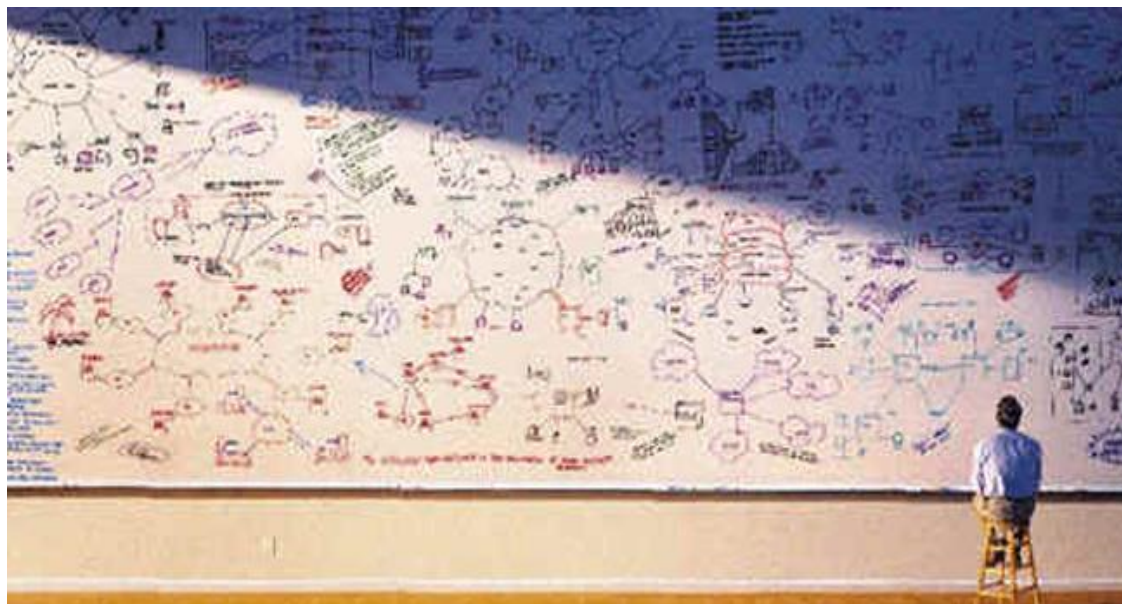


cecal



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

# 1



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

# INTRODUCCIÓN

# 1 - Introducción

## Contenido



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

1. Problemas computacionales y complejidad algorítmica
2. Problemas de optimización
3. Técnicas heurísticas
4. Metaheurísticas



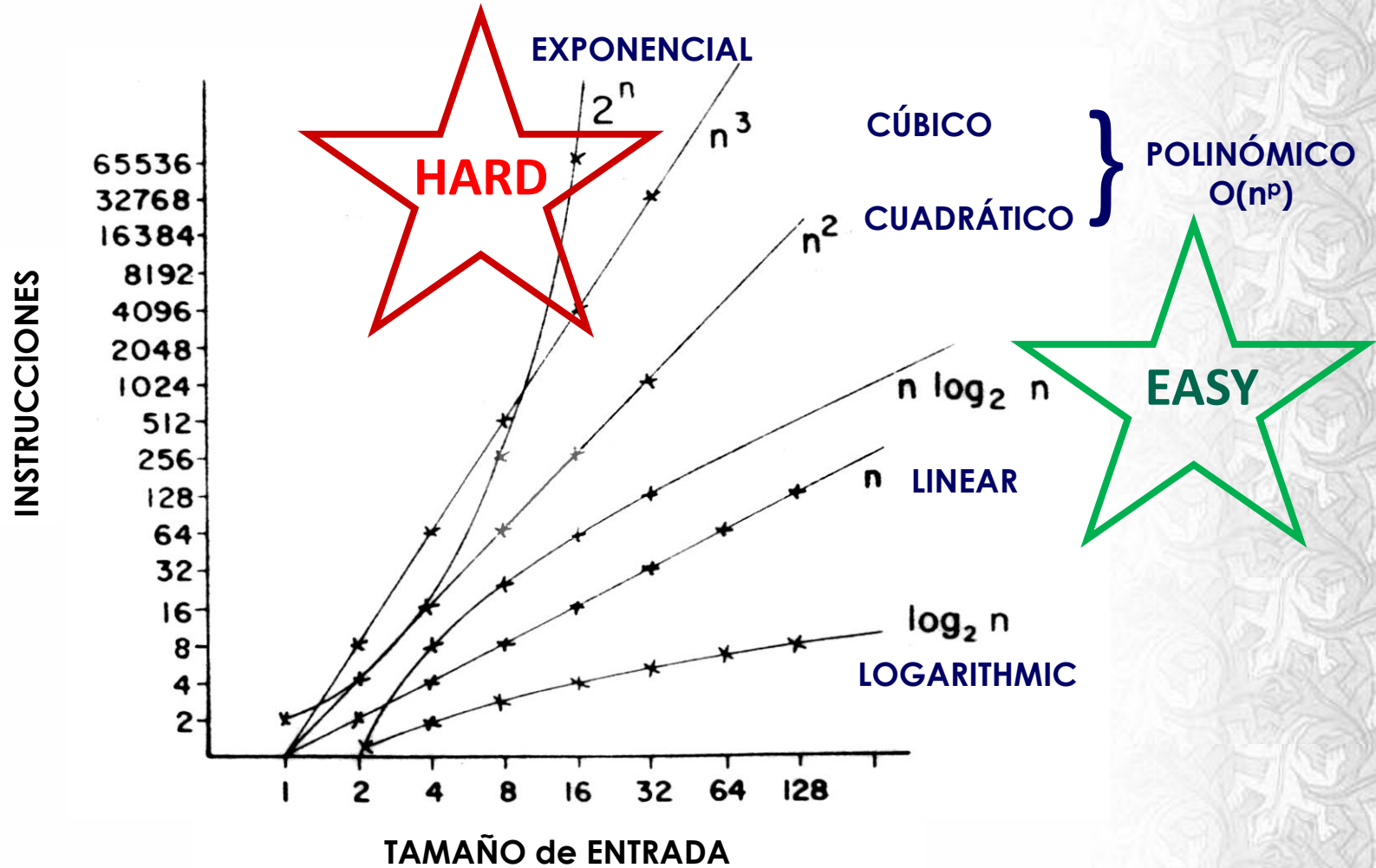
- Problemas “sencillos” y problemas “difíciles”
  - Sencillos: Ordenamiento de valores, multiplicación de matrices
  - Difíciles: problemas combinatorios
- ¿ Cómo evaluar la complejidad computacional ?
  - Comportamiento en caso promedio y peor caso.
  - Independiente de la arquitectura y lenguaje de programación (a priori)
  - Se analiza el **orden de magnitud** del número de instrucciones de un algoritmo, en función del **tamaño de la entrada**
  - Un algoritmo de complejidad  $O(g(n))$  ejecutado con datos de tamaño  $n$ , tendrá un tiempo de ejecución menor a  $|g(n)|$

# 1 - Introducción

## Complejidad algorítmica



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY



# 1 - Introducción

## Clases de complejidad



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

- Clase P
  - Un problema pertenece a la clase P si puede ser resuelto en tiempo **polinomial** en una computadora **determinística**
  - Ejemplos: Ordenamiento, búsqueda binaria, multiplicación matricial
- Clase NP
  - Un problema pertenece a la clase NP si puede ser resuelto en tiempo polinomial, pero usando una computadora **no determinística**
  - Fácil de verificar una solución, pero difícil hallarla

# 1 - Introducción

## Clases P y NP



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

- La clase P contiene problemas que pueden **resolverse de modo eficiente** (rápidamente)
- La clase NP contiene problemas cuya solución puede **verificarse de modo eficiente**
- En 1971 se planteó la pregunta: **¿Es  $P = NP$ ?**
- Desde entonces, sigue siendo una pregunta abierta para los teóricos (es la **principal conjetura** de la teoría de la computación)
- Se cree que  **$P \neq NP$** , pero no ha podido ser demostrado

# 1 - Introducción



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

## Problemas NP

- Es la clase de los problemas **difíciles** de resolver
- Ejemplos: TSP, ciclo hamiltoniano, SAT
- TSP (Travelling Salesman Problem)

*“Encontrar una permutación que represente el recorrido de una serie de ciudades de tal forma que todas sean visitadas (sólo una vez), minimizando la distancia total viajada”*



Ejemplo: TSP para doce ciudades de Alemania

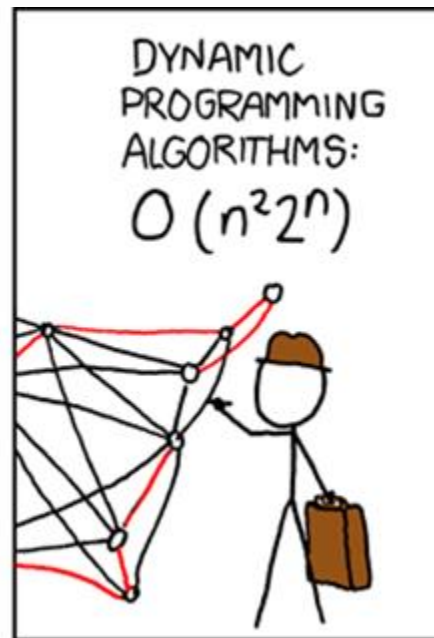
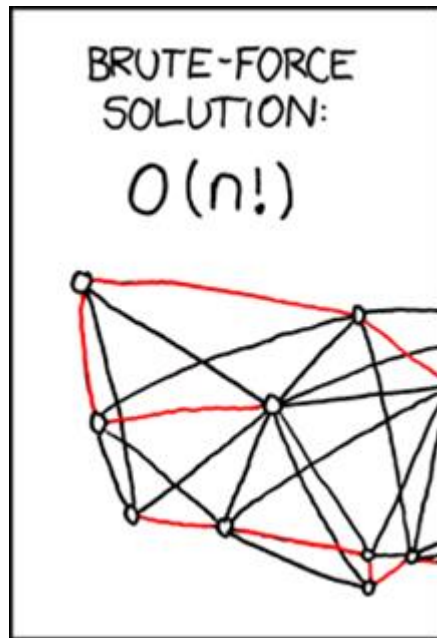


- ¿ Es realmente difícil de resolver ?
  - Considerando  $n$  ciudades (tamaño de la entrada =  $n$ ), la dimensión del espacio de búsqueda (permutaciones) es:  $(n-1)!/2$ 
    - Para  $n=10$ , hay 181.440 permutaciones posibles
    - Para  $n=12$  (caso del ejemplo para Alemania) hay 19.958.400 permutaciones posibles
    - Para  $n=20$  hay **60.822.550.204.416.000** permutaciones posibles
- Explorando 1.000 permutaciones por segundo, una búsqueda exhaustiva demandaría un tiempo estimado de:
  - $n=10 \rightarrow 3$  m
  - $n=12 \rightarrow 5$ h 30 m
  - $n=20 \rightarrow$  **1.928.670 años !**

SE NECESITAN  
ALGORITMOS  
MÁS EFICIENTES

# 1 - Introducción

## Resolviendo problemas NP-difíciles



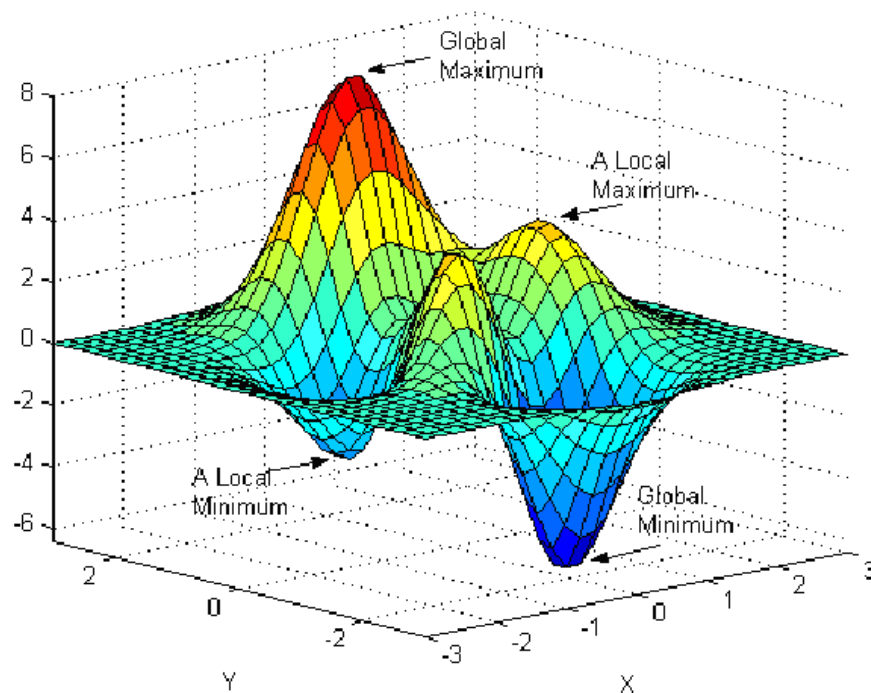
Hallar métodos  
inteligentes para  
su resolución



## Problemas NP

- Es necesario utilizar algoritmos de resolución **más eficientes** que la búsqueda exhaustiva, a medida que se incrementa el tamaño del problema
- La complejidad del algoritmo de resolución del problema se incrementa de manera superpolinomial con el tamaño de la entrada. Ejemplo: TSP  $O(n^2 * 2^n)$
- Una clase particular de problemas NP son los **problemas de optimización**
  - Objetivo: hallar la(s) solución(es) óptima(s) de un problema (de acuerdo a una función objetivo determinada)

# 2



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

# OPTIMIZACIÓN

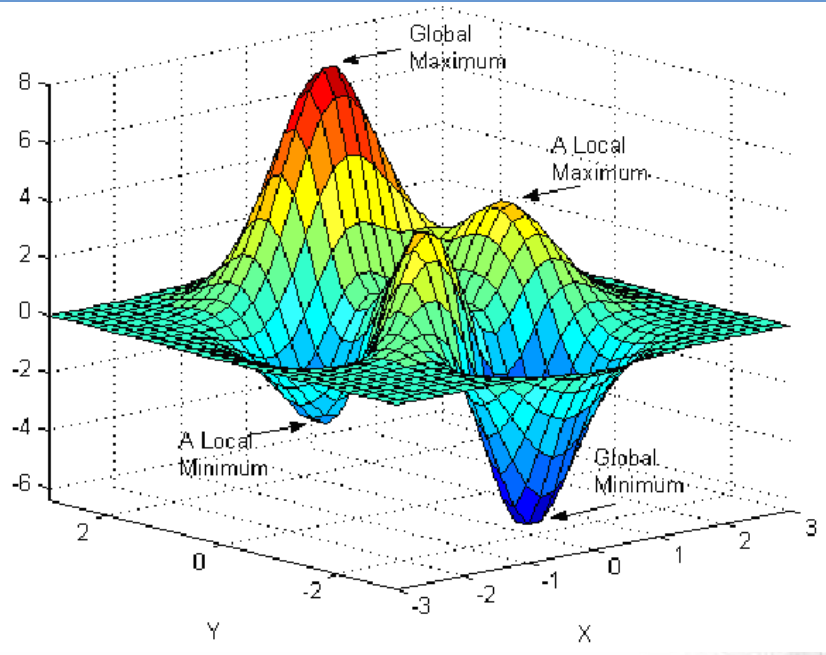
- Optimización numérica

$Min / Max f(\vec{x})$

sujeto a

$$g_i(\vec{x}) > 0 \quad i = 1, \dots, p$$

$$h_j(\vec{x}) = 0 \quad j = 1, \dots, r$$



$\vec{x}$  son las variables de decisión del problema,  $\vec{x} = (x_1, \dots, x_k)$

$g_i(\vec{x})$  y  $h_j(\vec{x})$  son las restricciones (de desigualdad y de igualdad, respectivamente)

$f(\vec{x})$  es la función a optimizar

## Problemas de optimización

- Optimización combinatoria
  - Variables de decisión **discretas** (dominio discreto)
  - Soluciones se representan como permutaciones o combinaciones
  - Ejemplo: TSP
- Problemas de búsqueda
  - Encontrar un elemento que cumple un conjunto de propiedades, entre un conjunto finito de elementos
  - Pueden formularse como problemas de optimización (minimizar la distancia al elemento buscado)

- Técnicas analíticas
  - Aplicables a un número muy reducido de problemas
- Métodos exactos
  - Utilizan técnicas de enumeración total o parcial:
    - Backtracking
    - Programación dinámica
    - Branch & Bound
  - Basados en Programación Matemática:
    - Método Simplex
    - Método del punto interior
  - Únicos errores: redondeo (y eventualmente truncamiento)

## Técnicas de resolución

- Métodos de aproximación
  - Encuentran una solución con error conocido a priori
  - Error fijo o derivado de un compromiso con el esfuerzo computacional requerido (mayor esfuerzo computacional, menor error)
  - Ejemplos: métodos markovianos, inferencia probabilística
- Métodos aleatorios
  - Con cierta probabilidad, encuentran una solución aproximada al problema, con un error máximo dado
  - Ejemplos: método Monte Carlo, método Las Vegas
- **Heurísticas y metaheurísticas**

-



3



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

# HEURÍSTICAS



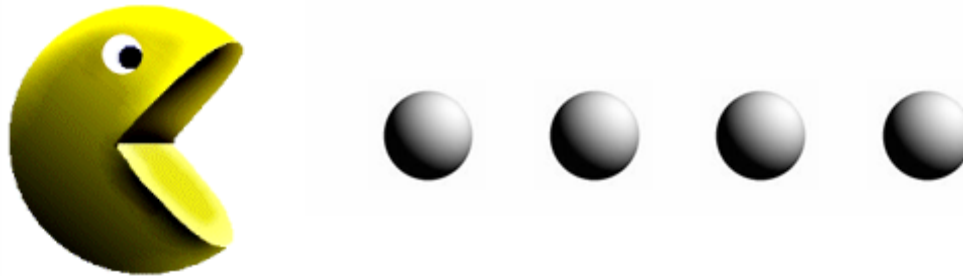
## Conceptos

- Métodos de resolución basados en procedimientos **conceptualmente simples** para encontrar soluciones de **buena calidad** (no necesariamente hallan la solución óptima) a problemas difíciles, de un modo **sencillo y eficiente**
- Los procedimientos de búsqueda a menudo están basados en el sentido común o emulación de fenómenos bien conocidos
- “Heurística” deriva del griego heuriskein (ὑρισκειν), que significa “encontrar” o “descubrir”

# 3 - Heurísticas

## Heurísticas constructivas

- Agregan iterativamente elementos para construir una solución
- Hasta que se completa una solución o se llega a algún criterio de parada
- El criterio de construcción es clave !
- Ejemplo: métodos greedy



# 3 - Heurísticas

## Heurísticas constructivas: método greedy

- greedy = “ávido” o “goloso”
- Idea: tratar de construir una solución seleccionando iterativamente elementos componentes de menor costo
- Para algunos problemas con estructura particular, la solución construida es una solución óptima (en general, **NO** es el caso)



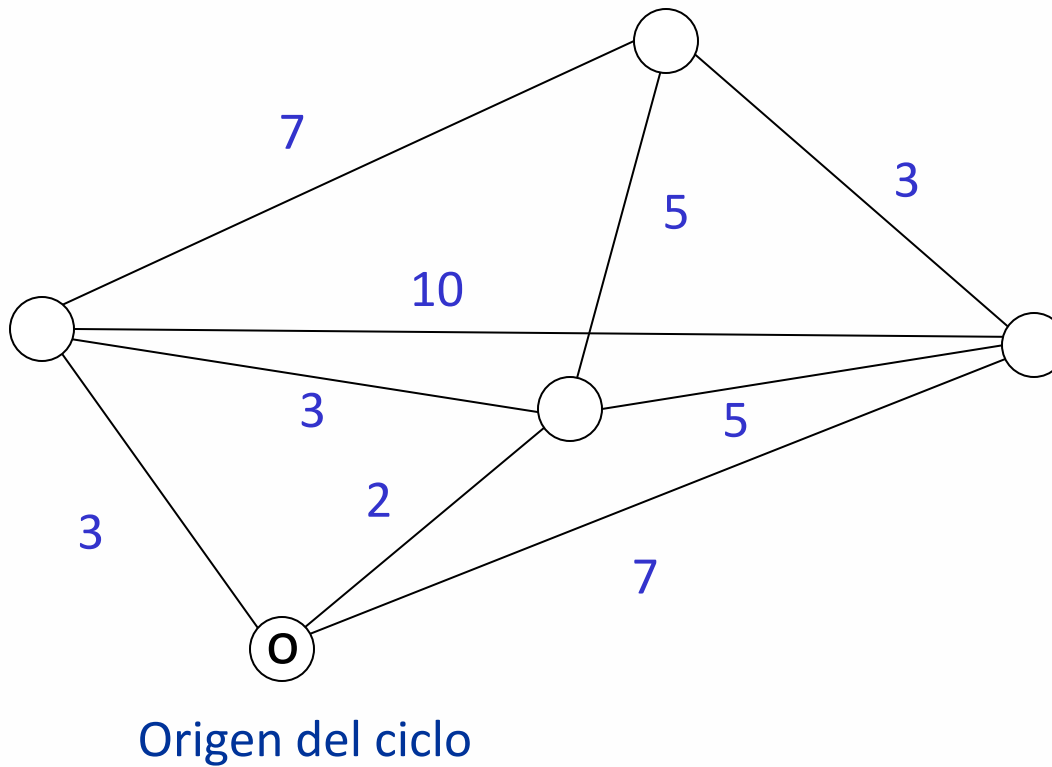
# 3 - Heurísticas



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

## Ejemplo: greedy para el TSP

- Método greedy para la instancia del TSP definida por el grafo:



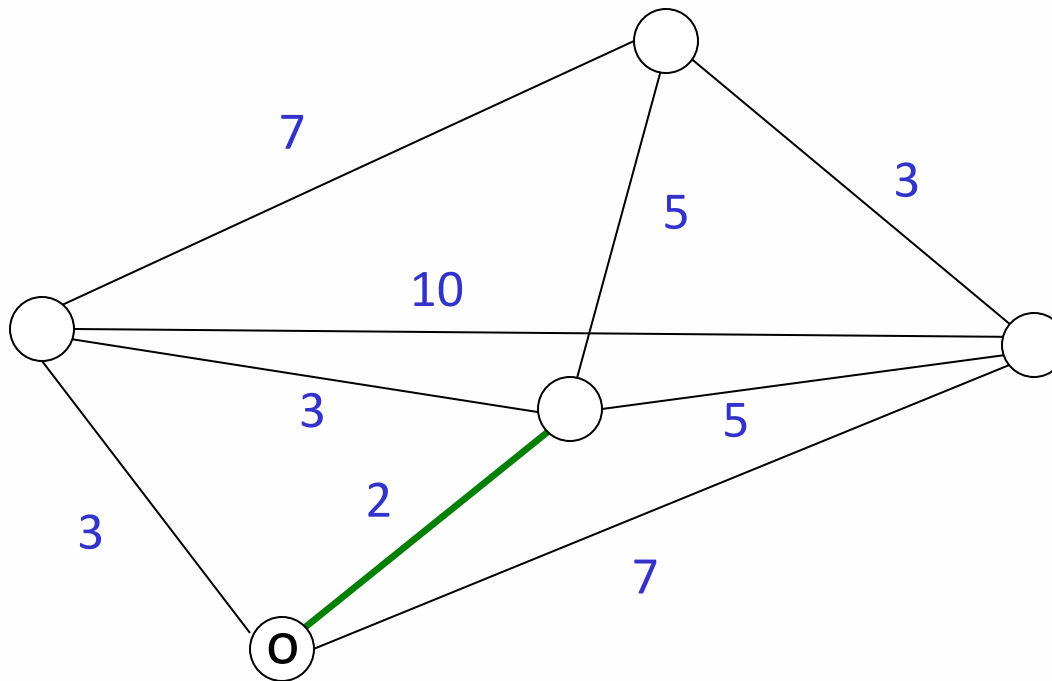
# 3 - Heurísticas



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

## Ejemplo: greedy para el TSP

- Greedy - Paso 1:



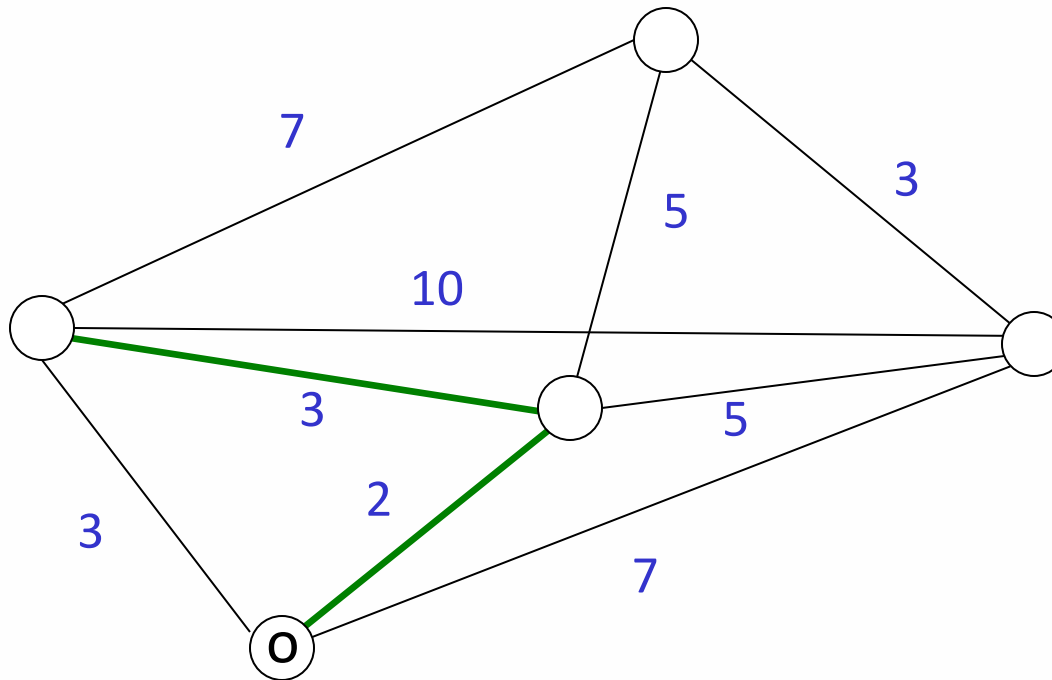
# 3 - Heurísticas



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

## Ejemplo: greedy para el TSP

- Greedy - Paso 2:



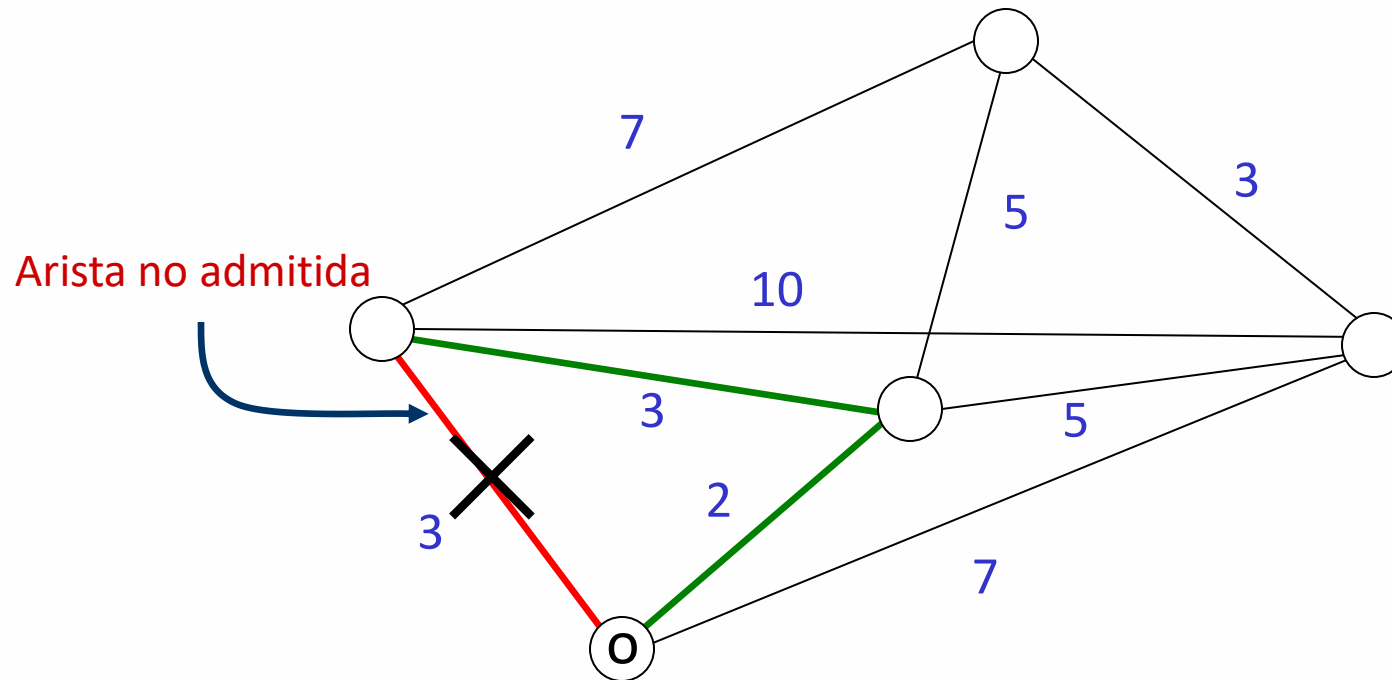
# 3 - Heurísticas



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

## Ejemplo: greedy para el TSP

- Greedy - Paso 3:





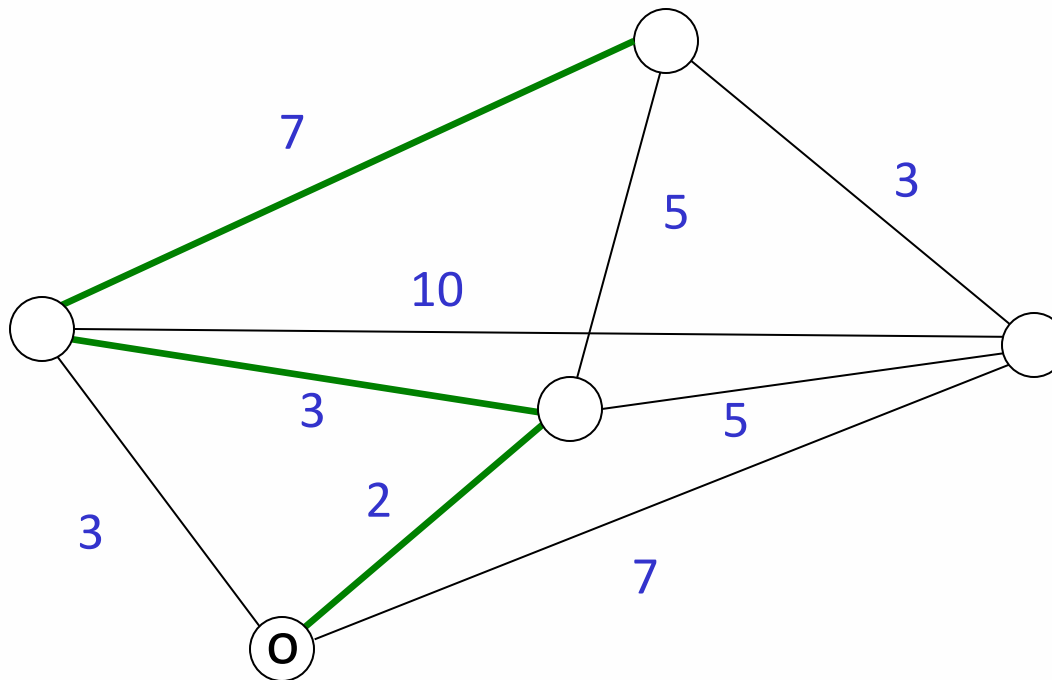
# 3 - Heurísticas



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

## Ejemplo: greedy para el TSP

- Greedy - Paso 3:



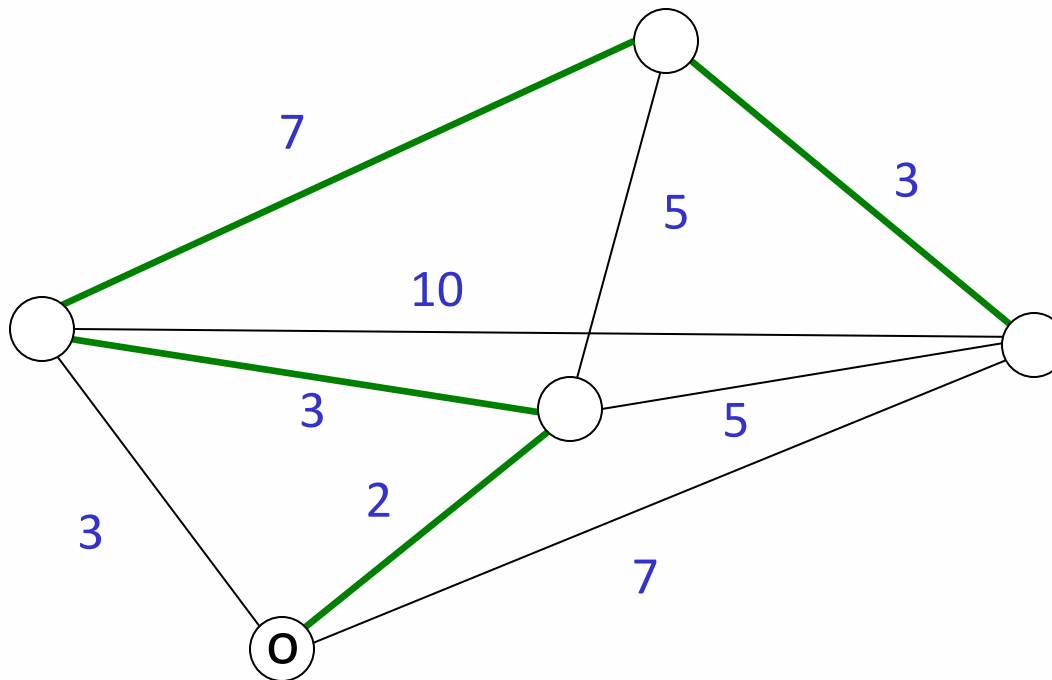
# 3 - Heurísticas



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

## Ejemplo: greedy para el TSP

- Greedy - Paso 4:



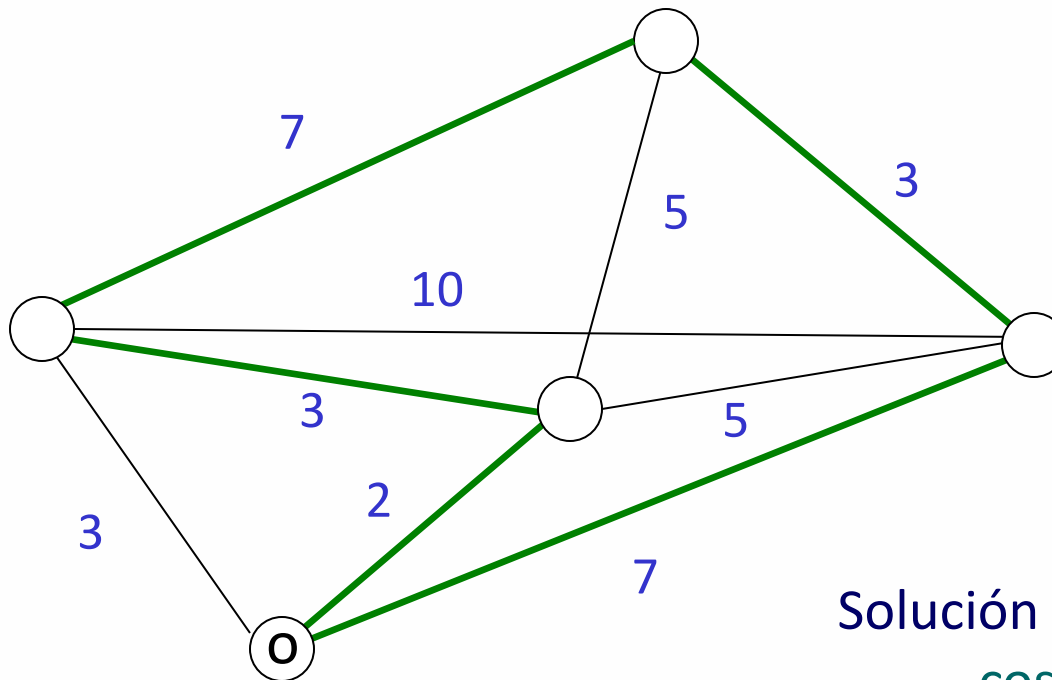
# 3 - Heurísticas



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

## Ejemplo: greedy para el TSP

- Greedy - Paso 5:

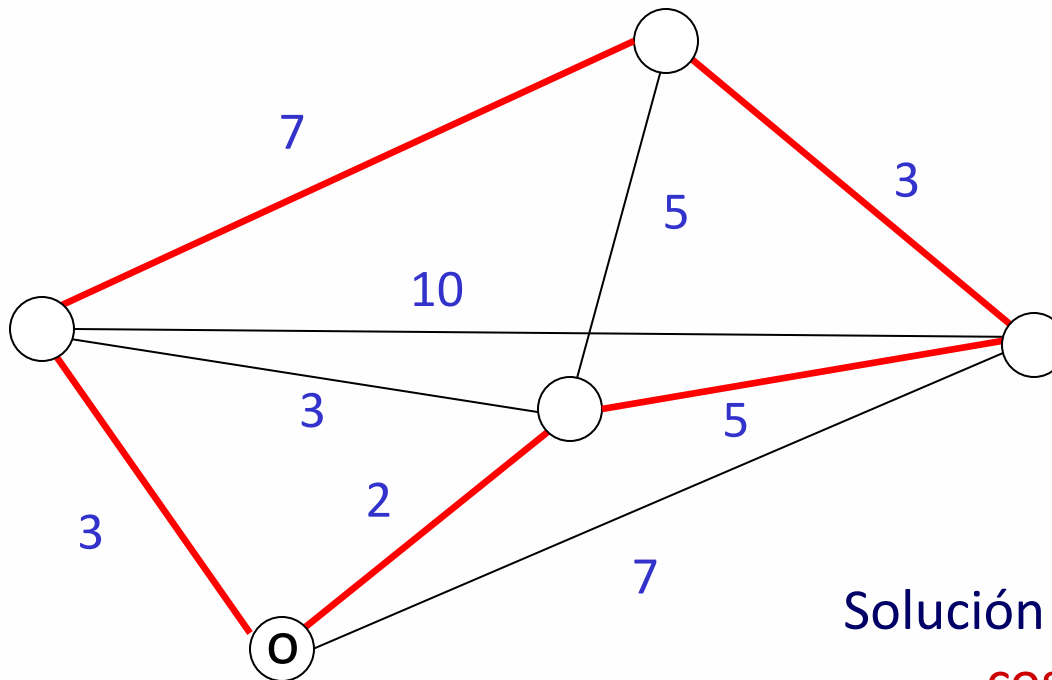


Solución construida:  
costo 22

# 3 - Heurísticas

## Ejemplo: greedy para el TSP

- Sin embargo, es posible obtener:



Solución alternativa:  
costo 20

# 3 - Heurísticas



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

## Búsqueda local (local search, LS)

- Mejor alternativa para la optimización. Obtiene mejores resultados que los métodos greedy
- Como contrapartida, requiere mayor esfuerzo computacional
- **Vecindad:**  $N : S \rightarrow 2^S$  a cada solución  $s$  se le asigna un conjunto de vecinos  $N(s) \subseteq S$
- En general, se define  $N(s)$  especificando los cambios sobre  $s$  para obtener sus vecinos (movimiento)
- Se reemplaza la solución inicial por una mejor solución de su vecindad

# 3 - Heurísticas

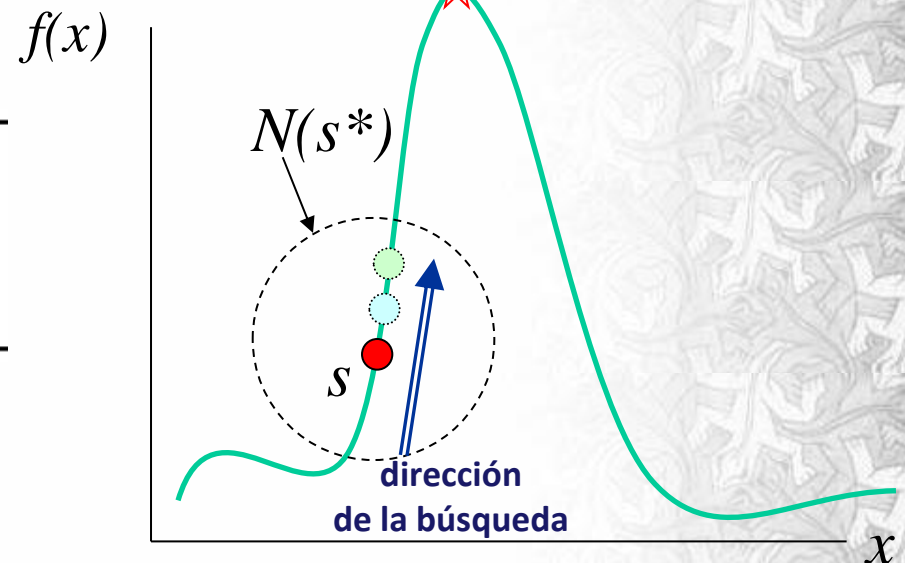
## Búsqueda Local Iterada (IILS)

- Búsqueda iterativa de mejores soluciones en una estructura de vecindades  $N(s)$ 
  - $N(s)$ : soluciones que pueden alcanzarse a través de una transformación (movimiento) en  $s$
- El algoritmo se detiene luego de encontrar un óptimo local/global o luego de realizar un número dado de iteraciones

---

```
1:  $s \leftarrow \text{GenerateInitialSolution}()$ 
2: repeat
3:    $s \leftarrow \text{Improve}(N(s))$ 
4: until no improvement is possible
```

---

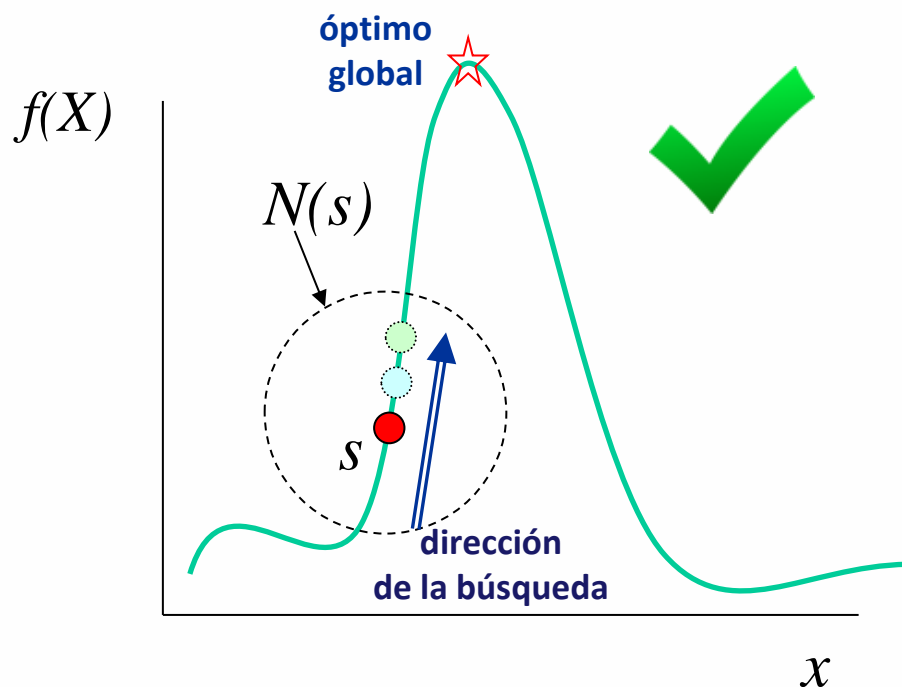


# 3 - Heurísticas

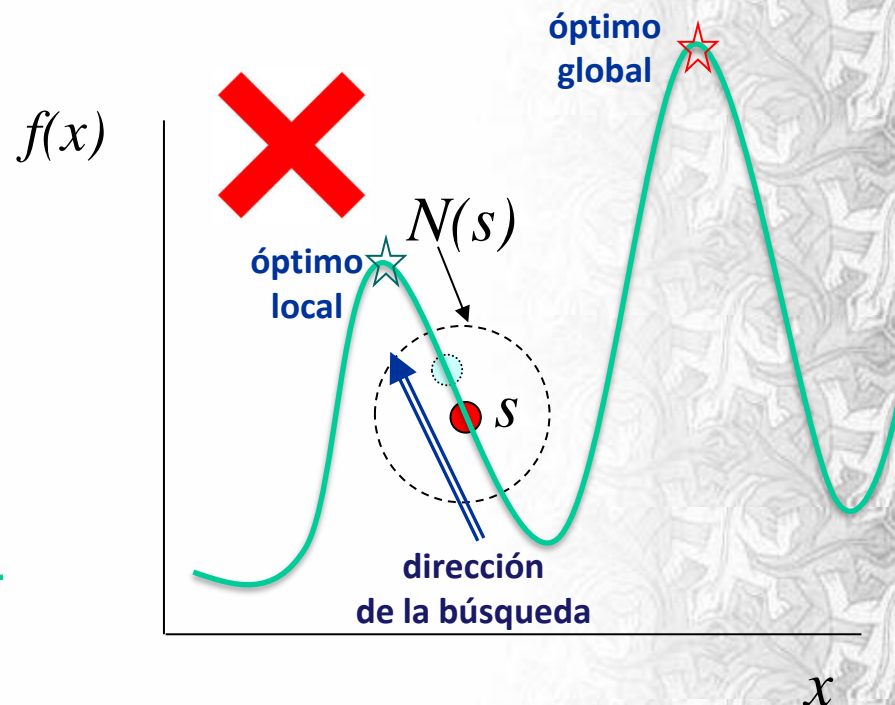
## Búsqueda Local Iterada (IILS)

- Qué tan efectiva es la Búsqueda local Iterada ?

### a – función unimodal

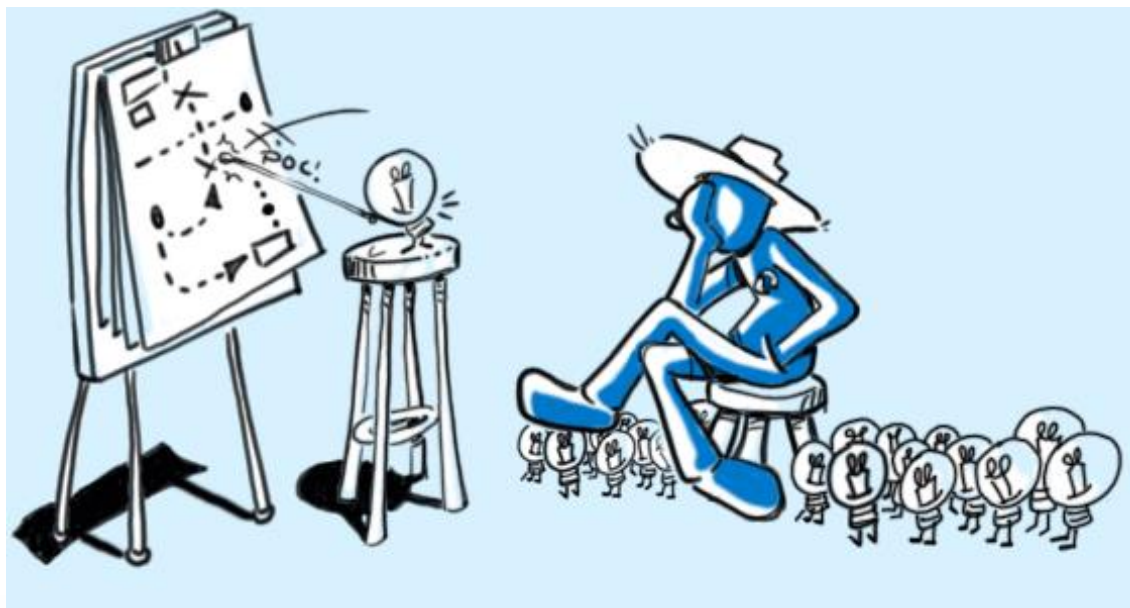


### b – función multimodal



- La Búsqueda local Iterada puede quedar atrapada en óptimos locales

# 4



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

# METAHEURÍSTICAS





## Conceptos

- Métodos eficientes y precisos para hallar soluciones de calidad a problemas de optimización complejos
- “High-level strategy that combines a set of underlying simpler operation techniques (usually heuristics) aimed at obtaining a more powerful procedure” (Glover, 1986)
- Principales características:
  1. **Estrategias** que guían la búsqueda, usualmente empleando heurísticas subordinadas
  2. Admiten una descripción abstracta **de alto nivel**
  3. Tienen **utilización genérica** (no específica para una clase de problemas)
  4. Deben ser **instanciadas** para clase concreta de problemas a resolver

# 4 - Metaheurísticas



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

## Principales objetivos

- Hallar eficientemente (i.e. rápidamente) **soluciones de buena calidad** (con valores de la función objetivo cercanos al óptimo)
- Cubrir el espacio de soluciones **sin quedar atrapadas** en zonas específicas (especialmente óptimos locales)



# 4 - Metaheurísticas

## Un ejemplo: Simulated Annealing

- Heurística: Búsqueda Local Iterada
  - Búsqueda en la vecindad de la solución actual
  - Si se encuentra una mejor solución, reemplazar a la actual
  - Desventaja: puede quedar atrapada en un óptimo local
- Metaheurística: Simulated Annealing (SA)
  - Búsqueda local iterada, pero **una solución peor que la actual puede ser aceptada** de acuerdo a una cierta probabilidad



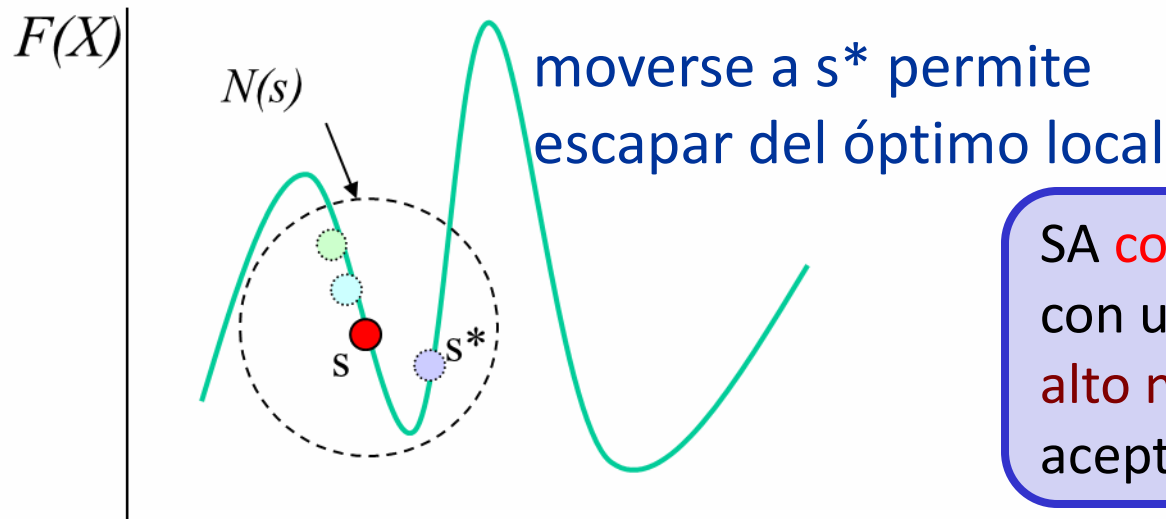
# 4 - Metaheurísticas

## Simulated annealing

```
1:  $s \leftarrow \text{GenerateInitialSolution}()$ 
2:  $T \leftarrow T_0$ 
3: while termination conditions not met do
4:    $s^* \leftarrow \text{PickAtRandom}(N(s))$ 
5:   if  $f(s^*) < f(s)$  then
6:      $s \leftarrow s^*$ 
7:   else
8:     Accept  $s^*$  as new solution with probability  $p(T, s, s^*)$ 
9:   end if
10:   $\text{Update}(T)$ 
11: end while
```

búsqueda  
local

Idea clave: permite  
seleccionar peores soluciones  
que la actual, tratando de  
escapar de óptimos locales



SA **combina** una heurística  
con un **procedimiento de  
alto nivel** que permite  
aceptar peores soluciones

# 4 - Metaheurísticas

## Simulated annealing



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

Combinar resulta siempre una buena idea, no es así ?



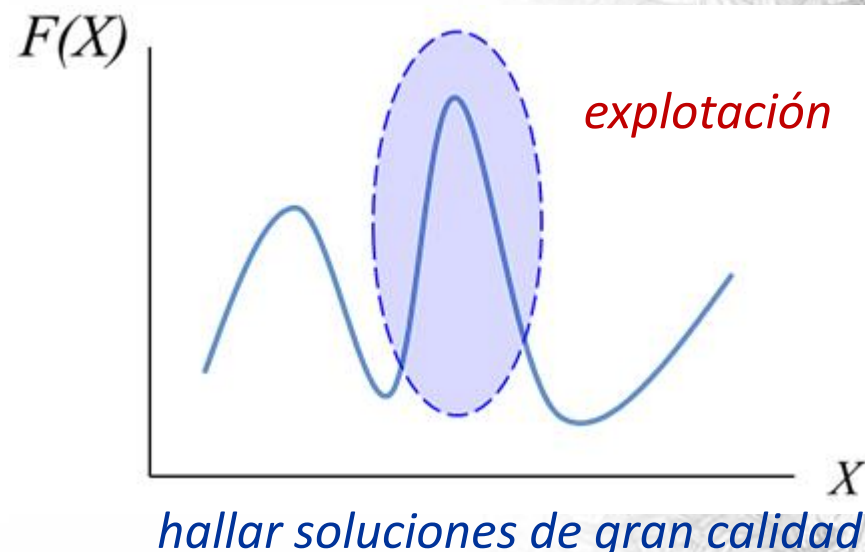
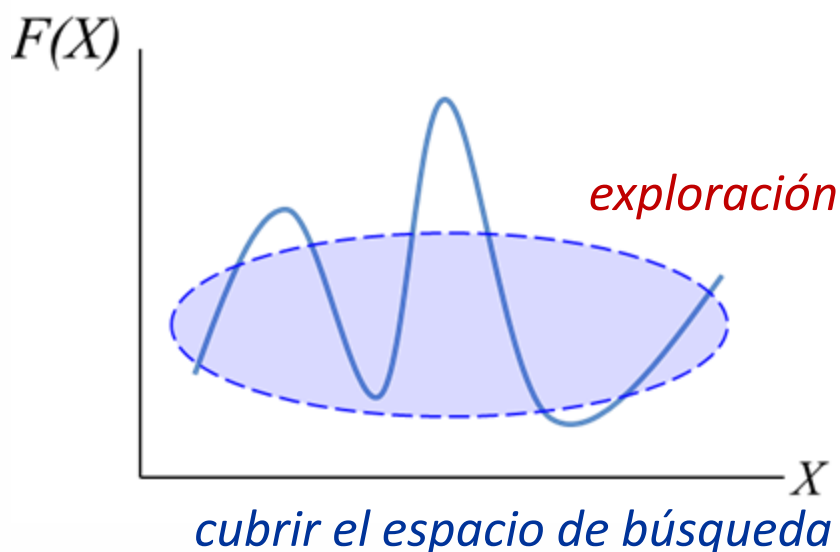
# 4 - Metaheurísticas



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

## Características y conceptos

- Principales objetivos:
  1. Hallar (eficientemente) **soluciones de buena calidad** para problemas de optimización
  2. Cubrir el espacio de búsqueda **sin quedar atrapadas** en zonas específicas
- Dos conceptos fundamentales
  - **Exploración** del espacio de búsqueda o **diversificación**
  - **Explotación** de buenas soluciones encontradas o **intensificación**



# 4 - Metaheurísticas

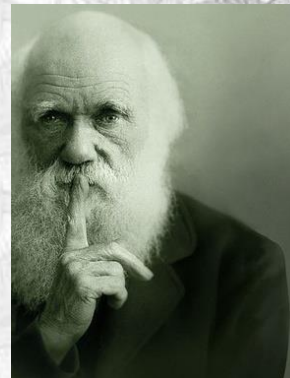
## Metaheurísticas y clasificación

- Lista de ejemplos (no exhaustiva)

- Simulated Annealing
- Tabu Search
- Greedy Randomized Adaptive Search Procedure
- Variable Neighborhood Search
- Guided Local Search

- Ant Colony Optimization
- Particle Swarm Optimization
- Evolutionary Algorithms

- Existen múltiples criterios de clasificación
- Considerando el número de soluciones tentativas:  
*basadas en trayectoria* vs.  
*basadas en población*



# 4 - Metaheurísticas

## Basadas en trayectoria y en población

- Basadas en trayectoria: solución única
  - Una única solución que se reemplaza en cada paso
  - Define una *trayectoria* en el espacio de búsqueda
  - Ejemplos:
    - Simulated annealing, Tabu search, etc.
- Basadas en población: usan un conjunto de soluciones (la *población*)
  - Algunas soluciones se reemplazan en cada iteración
  - Permite la *recombinación* de soluciones
  - Ejemplos :
    - Ant Colony Optimization
    - Particle Swarm Optimization
    - **Algoritmos Evolutivos**, etc.



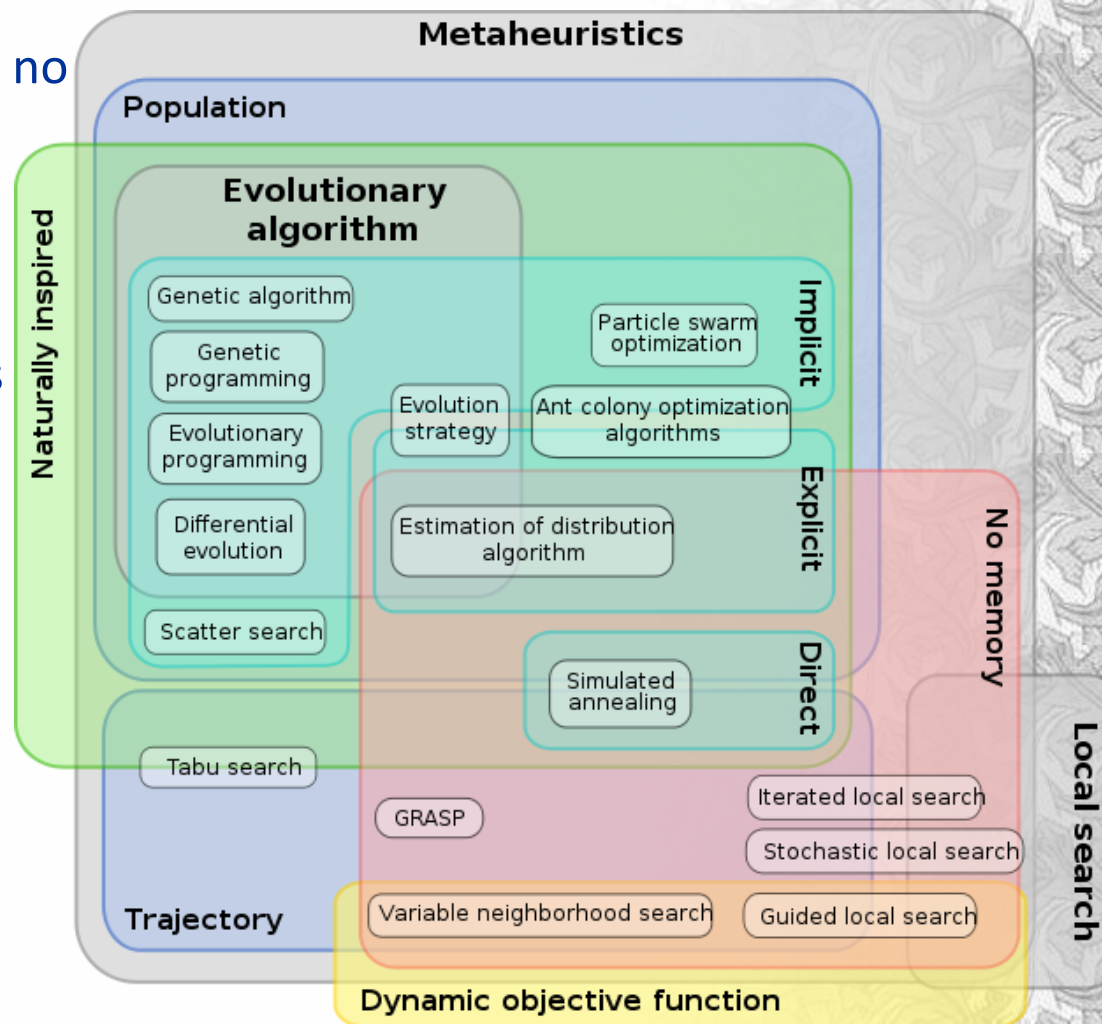


# 4 - Metaheurísticas



## Metaheurísticas y clasificación

- Otros criterios de clasificación
  - Inspiradas en la naturaleza o no
  - Aleatorias/deterministas
  - Constructivas o no
  - Con o sin memoria
  - Implícitas/explicitas/directas





## **Autoestudio: Introducción a las Metaheurísticas**

Slides 51 a 61 y texto “Essentials of metaheuristics”

<http://cs.gmu.edu/~sean/book/metaheuristics/>

# 4 - Metaheurísticas



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

## Lista (no exhaustiva) de ejemplos

- Simulated Annealing (SA)
- Tabu Search (TS)
- Greedy Randomized Adaptive Search Procedure (GRASP)
- Variable Neighbourhood Search (VNS)
- Guided Local Search (GLS)
- Iterated Local Search (ILS)
- Scatter Search (SS)
- Path Relinking (PR)
- Ant Colony Optimization (ACO)
- Particle Swarm Optimization (PSO)
- **Computación Evolutiva (CE): variantes y derivados**

## Simulated annealing

- Basado en el procedimiento físico del enfriamiento de los cristales
- Idea: permitir movimientos a soluciones que empeoren la función objetivo para escapar a óptimos locales
- La probabilidad de realizar ese tipo de movimientos decrece durante la búsqueda
- Las soluciones que mejoran la función objetivo siempre son aceptadas
- Es un algoritmo probabilístico de búsqueda local
- Es una técnica aleatoria, que mantiene un único individuo, sin memoria, y basada en trayectoria



## Tabu Search

- Búsqueda local con memoria a corto plazo para escapar de mínimos locales y evitar ciclos
- Algunas soluciones examinadas recientemente son incluidas en la memoria y se vuelven tabú (prohibidas) al tomar decisiones acerca del siguiente punto de búsqueda
- Generalmente, la lista tabú se maneja en forma FIFO
- Es determinística, aunque se le pueden agregar elementos probabilísticos
- Es una técnica determinística (en general), que mantiene un único individuo, con memoria, y basada en trayectoria



## Greedy Randomized Adaptive Search Procedure (GRASP)

- Combina procedimientos constructivos y de búsqueda local
- Es un procedimiento iterativo en dos etapas: una de construcción de la solución y otra de mejora
- La construcción se realiza a partir de la selección de subpartes en forma aleatoria de conjuntos de candidatos
- La etapa de mejora consiste en aplicar búsqueda local a la solución construida
- Es una técnica aleatoria, que mantiene un único individuo, sin memoria, y constructiva

## Variable Neighbourhood Search

- Se define un conjunto de estructuras de vecindad (en general se utilizan vecindades de cardinalidad creciente)
- La técnica opera en tres fases
  - *Shaking*: seleccionar aleatoriamente una solución  $s'$  perteneciente a la vecindad  $k$ -ésima de la presente solución  $s$
  - *Búsqueda local*:  $s''$  partiendo de  $s'$
  - *Movimiento*: se acepta  $s''$  si mejora  $s$
- Una mala solución en una vecindad puede ser buena en otra
- Existen múltiples variantes:
  - Variable Neighbourhood Descent (VND)
  - Variable Neighbourhood Decomposition Search (VNDS)
  - Skewed Variable Neighbourhood Search (SVND)
- Es una técnica aleatoria, que mantiene un único individuo, sin memoria, y constructiva



## Guided Local Search

- La búsqueda se conduce a través de cambios dinámicos en la función objetivo
- Idea: salir de óptimos locales, empeorando el óptimo local, mediante el cambio de la función objetivo
- Basado en características o propiedades de la solución que se usan para discriminar una solución de otra
- Es una técnica determinística, que mantiene un único individuo, sin memoria, y basada en trayectoria



# 4 - Metaheurísticas



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

## Iterated Local Search (con perturbación)

- Consiste en una aplicación iterativa de una búsqueda local
- Se aplica la búsqueda local a una solución inicial hasta encontrar un óptimo local
- Luego se *perturba* la solución (se aplica un movimiento altamente disruptivo) y se realiza nuevamente la búsqueda local
- Es una técnica aleatoria, que mantiene un único individuo, sin memoria y basada en trayectoria

## Scatter Search

- Se genera un conjunto de soluciones combinando subconjuntos de un conjunto de soluciones de referencia (factibles)
- Las soluciones generadas no necesariamente son factibles, deben factibilizarse (mediante un algoritmo de “reparación”) y pueden ser mejoradas mediante búsqueda local
- El conjunto de referencia se actualiza con las soluciones generadas
- Las soluciones se codifican como puntos en un espacio euclideo. Las nuevas soluciones se obtienen mediante combinaciones lineales de las soluciones de referencia (se permite utilizar pesos positivos y negativos, siendo posible salir de la región factible)
- Es una técnica poblacional, sin memoria, y constructiva. Puede ser determinística ó aleatoria (de acuerdo al procedimiento que genera las nuevas soluciones)



## Path Relinking

- Surgió como una generalización de Scatter Search
- El espacio euclideano se sustituye por espacios de vecindades. En este caso, las combinaciones lineales son caminos entre soluciones en el espacio de vecindades
- Caminos habilitados: desde la solución inicial los movimientos deben introducir sucesivamente atributos seleccionados en una búsqueda guiada
- Al igual que SS, es una técnica poblacional, sin memoria, y constructiva. Puede ser determinística ó aleatoria (de acuerdo al procedimiento que genera las nuevas soluciones)

## Ant Colony Optimization

- Se basan en el comportamiento social de las hormigas (cada hormiga tiene capacidades restringidas, pero en conjunto logran un comportamiento “inteligente”)
- Existe comunicación indirecta a través de sustancia química (feromona) que sirve de referencia a otras hormigas
- Idea del algoritmo: cada hormiga artificial construye una solución agregando componentes a una solución parcial en consideración
- La decisión sobre qué componentes agregar es probabilística, a partir de la feromona depositada y un factor heurístico (para asegurar la exploración)
- Al terminar de construir la solución, la regla de actualización incrementa la feromona en las componentes con “mejor calidad”
- Es una técnica aleatoria, poblacional, con memoria, y constructiva

## Particle Swarm Optimization

- Técnica inspirada en el comportamiento de los enjambres de aves e insectos
- Si un insecto encuentra un camino atractivo, el resto del enjambre lo puede seguir rápidamente, incluso si están en el lado opuesto del enjambre
- Se modela con partículas que tienen atributos de posición y velocidad en un espacio multidimensional
- Las partículas se mueven en ese espacio y recuerdan la mejor posición que conocen
- Las posiciones de mejor calidad son comunicadas, y las partículas ajustan sus atributos en base a ellas
- Es una técnica aleatoria, poblacional, sin memoria, y basada en trayectoria

## Computación evolutiva

- Técnicas de optimización que trabajan sobre una **población** de soluciones que **evoluciona** mediante mecanismos de selección y construcción de soluciones candidatas por recombinación de características de las soluciones anteriores
- Etapas:
  - Evaluación de la función de fitness (evalúa calidad de las soluciones)
  - Selección de individuos adecuados (de acuerdo al fitness)
  - Aplicación de operadores evolutivos
  - Reemplazo o recambio generacional
- Múltiples variantes y derivados: Algoritmos Genéticos, Programación Evolutiva, Estrategias de Evolución, Algoritmos Meméticos, Algoritmos de Estimación de Distribuciones, etc.
- Son técnicas aleatorias, poblacionales, sin memoria, y constructivas

## Aplicaciones

- La mayoría de los problemas complejos del mundo real no pueden resolverse usando algoritmos en tiempo polinomial
- En muchas aplicaciones prácticas, no es posible ni siquiera determinar la existencia de una solución eficiente
- Hay muchos problemas para los cuales el mejor algoritmo que se conoce requiere tiempo exponencial

**HEURÍSTICAS Y METAHEURÍSTICAS CONSTITUYEN ALTERNATIVAS PARA HALLAR SOLUCIONES DE BUENA CALIDAD EN TIEMPOS RAZONABLES**

