

---

# COMPUTACIÓN EVOLUTIVA

## CONTEXTO

### ► Que es Soft Computing?

✓ 90s

✓ Lofti Zadeh

► Flexibilidad, imprecisión, inexactitud, vaguedad, aleatoriedad, no determinismo, etc.

Hard	Soft
Rigid	Flexible
Fixed	Movable/Adjustable
Systematic	Random
Well-defined	Vague
Exact	Inexact/Approximate
Precise	Imprecise
Measurable	Perceivable
Solid	Porous
Deterministic	Non-deterministic
...	...

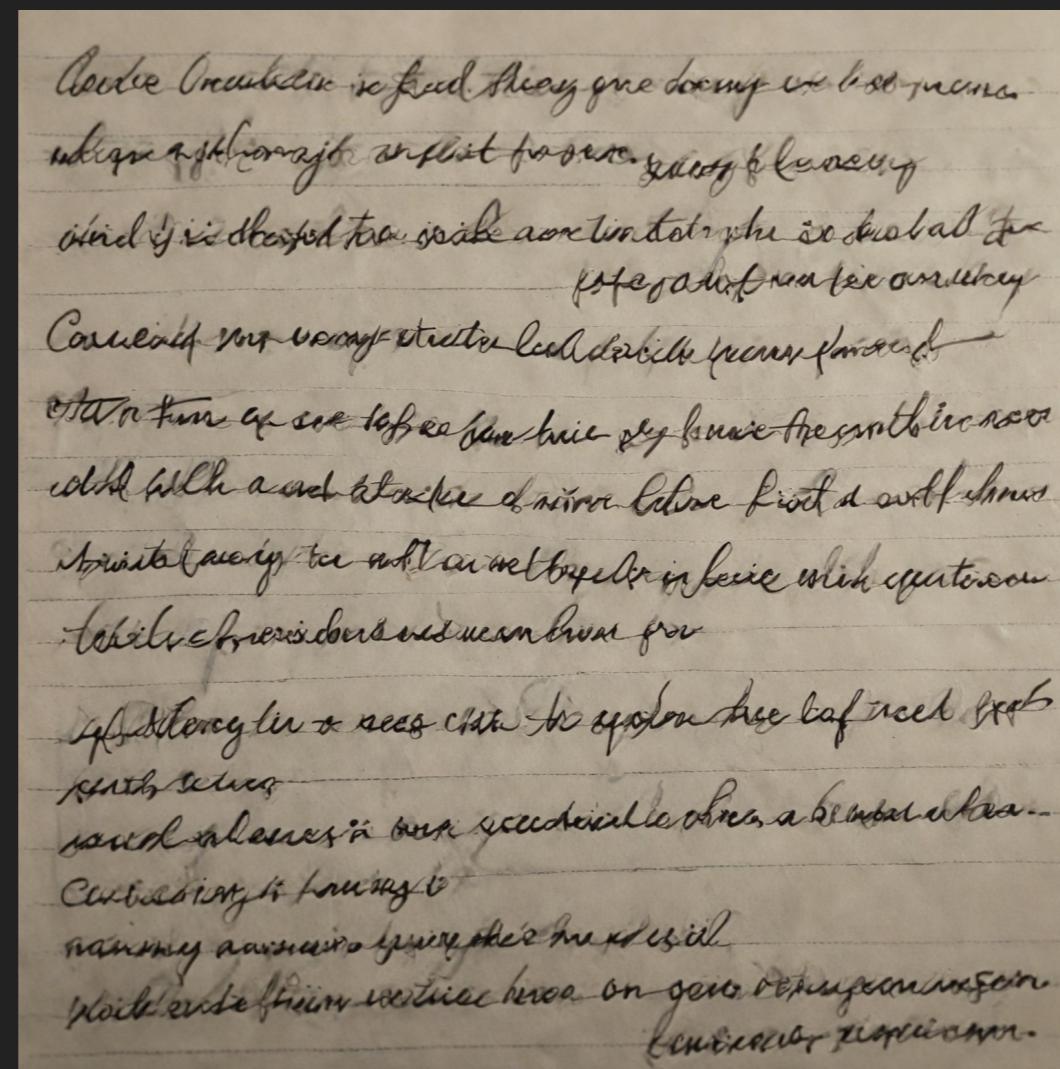
## CONTEXTO - SOFT COMPUTING - EJEMPLOS

- ▶ Estacionar un coche en un espacio estrecho.



# CONTEXTO - SOFT COMPUTING - EJEMPLOS

- ▶ Reconocimiento de caracteres escritos a mano.



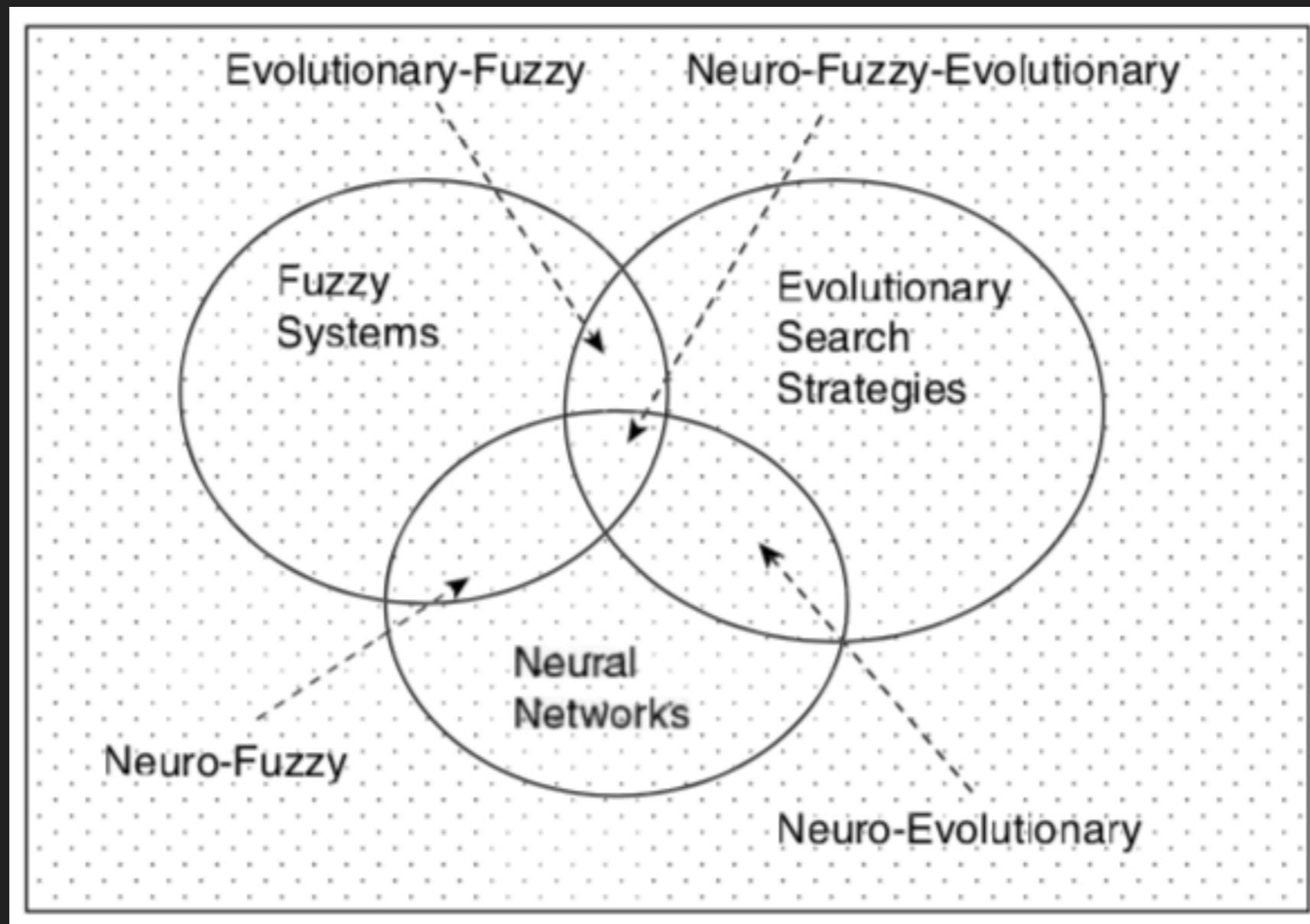
# CONTEXTO - SOFT COMPUTING - DEFINICIÓN

- ▶ **Soft Computing** es una familia de técnicas con capacidad para resolver una clase de problemas para los cuales otras técnicas convencionales resultan inadecuadas.
- ▶ Componentes de Soft Computing

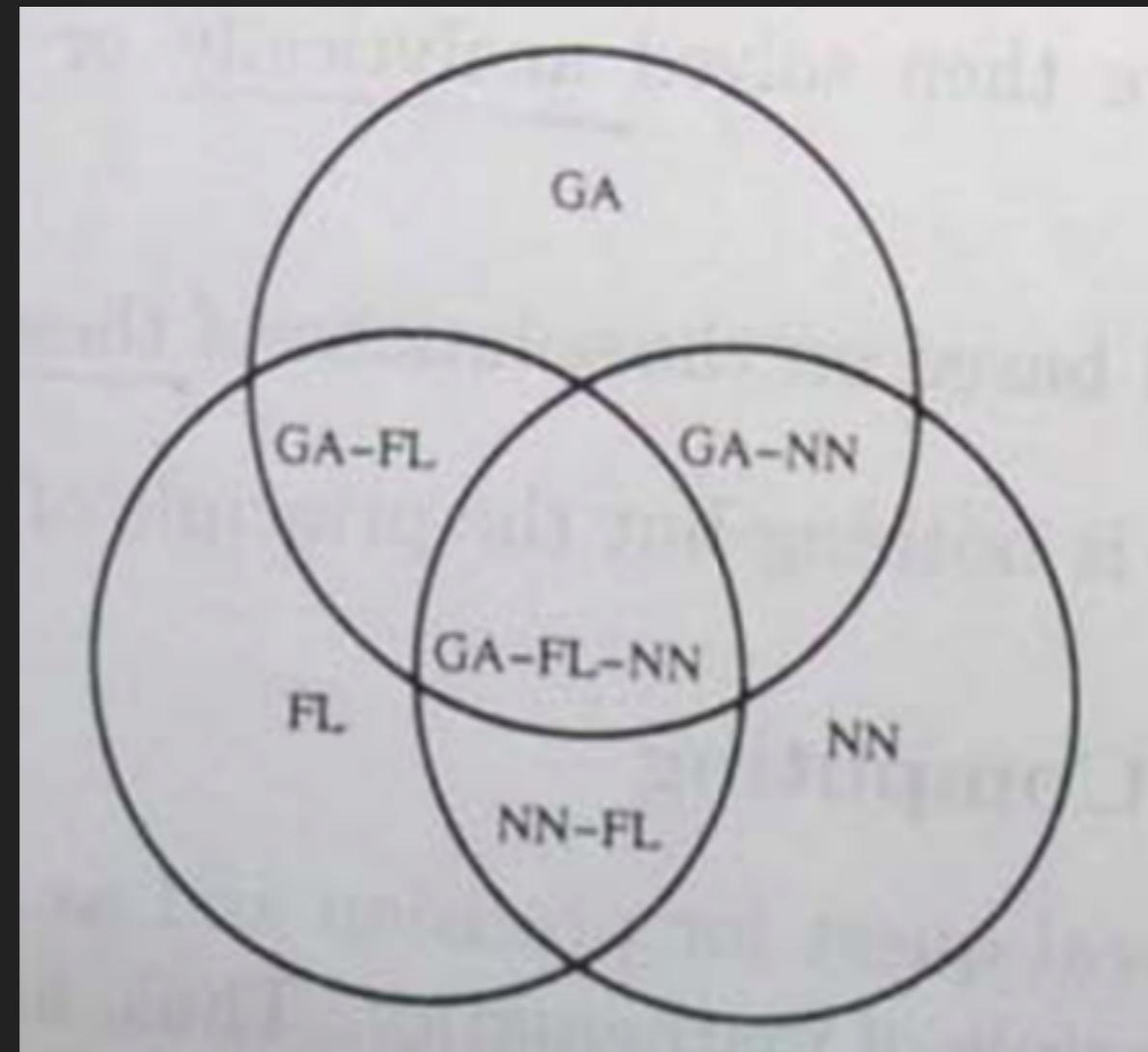


Technique	Application domain
Fuzzy systems	Vagueness / imprecision / inexactness / Approximate reasoning
Evolutionary searches	Complex optimization
Artificial neural networks	Learning and curve fitting / Pattern classification, association, clustering

# CONTEXTO - SOFT COMPUTING - COMPONENTES



## CONTEXTO - SOFT COMPUTING - COMPONENTES



# PROBLEMAS DE INGENIERÍA

- ▶ Enfoque de caja negra (Entrada -> Modelo -> Salida)
- ▶ El sistema espera alguna **entrada**
  - ✓ persona
  - ✓ sensor
  - ✓ otra computadora
- ▶ Puede haber 1 a N **entradas** de diferentes tipos
- ▶ El **modelo computacional** (simple o muy complejo)
- ▶ Los **tipos de problemas** existentes dependen de cual de estos tres componentes se **desconocen** (Entrada, Modelo o Salida)



## TIPOS DE PROBLEMAS

- ▶ Optimización
- ▶ Modelado
- ▶ Simulación



## TIPOS DE PROBLEMAS

- ▶ Optimización
- ▶ Modelado
- ▶ Simulación



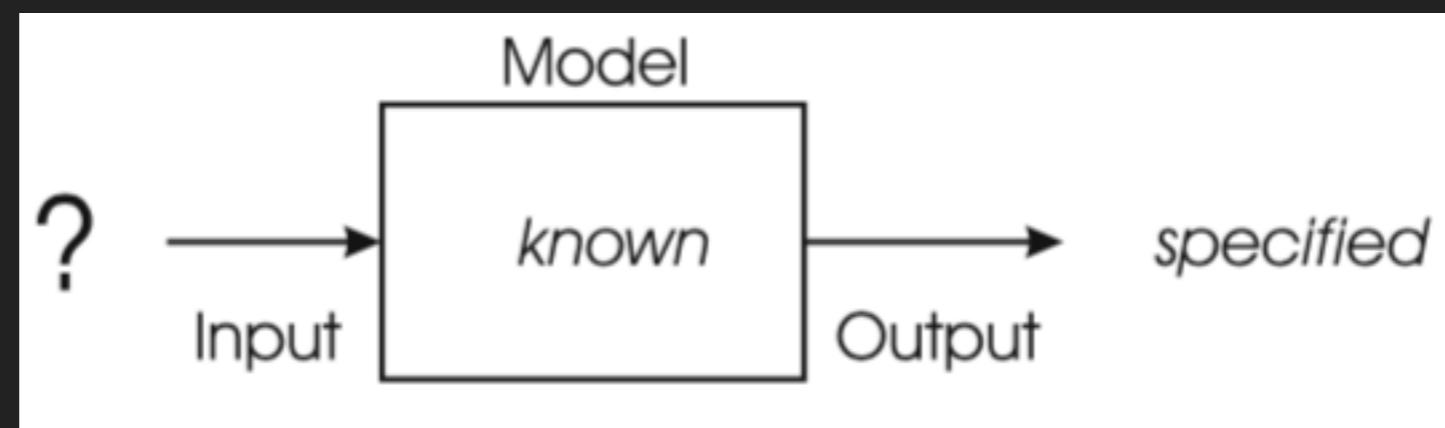
# OPTIMIZACIÓN

- ▶ La **optimización** es el proceso de encontrar la mejor, entre todas las soluciones factibles para un problema específico y con base en ciertos criterios de desempeño.



# OPTIMIZACIÓN

- ▶ En un problema de **optimización** se conoce el **modelo**, junto con el **resultado deseado** (o una descripción del resultado deseado), y la tarea es encontrar la(s) **entrada(s)** que conducen a este resultado
- ▶ Conocer el **modelo** significa que podemos calcular la **salida** para cualquier **entrada**



## EJEMPLO DE OPTIMIZACIÓN (II)

- ▶ ¿Qué valores de  $x_1$  y  $x_2$  se necesita para que  $Z$  sea máxima?

Maximizar

sujeta a

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

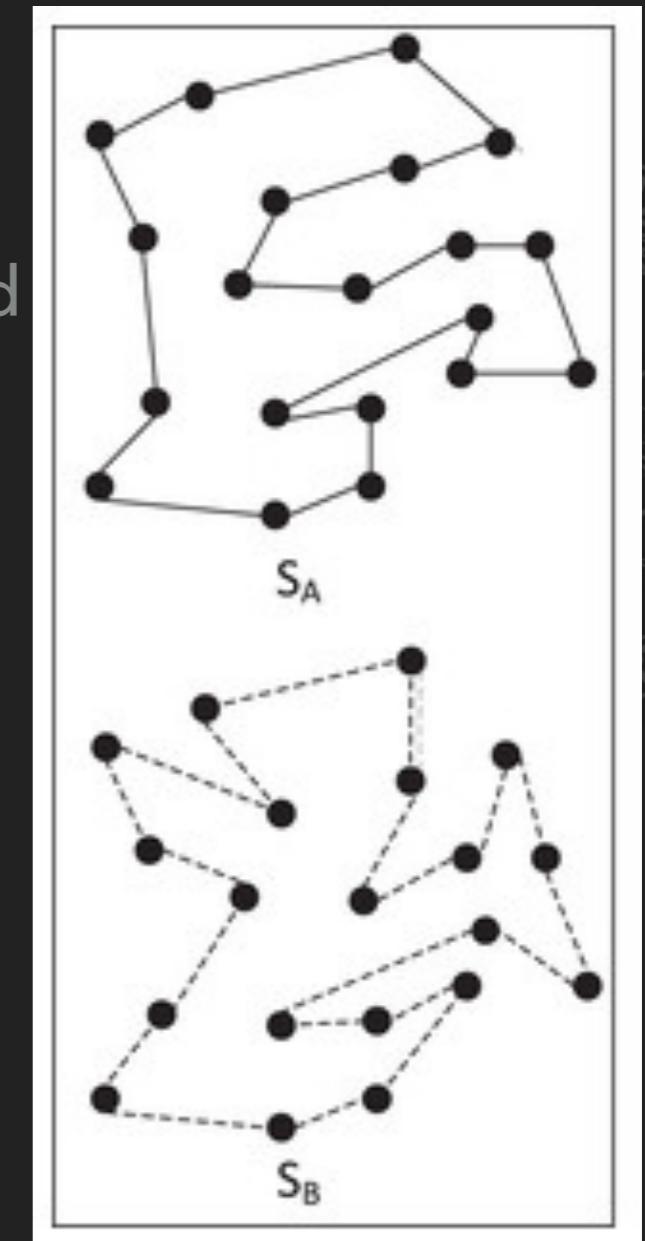
y

$$x_1 \geq 0, \quad x_2 \geq 0.$$



## EJEMPLO DE OPTIMIZACIÓN (II)

- ▶ Problema del viajante de comercio
- ▶ Consiste en un conjunto de ciudades y tenemos que encontrar el recorrido más corto que visite cada ciudad exactamente una vez
- ▶ **modelo -> fórmula**
- ▶ Cada secuencia de ciudades (**las entradas**)
- ▶ Se calcula la duración del recorrido (**la salida**)
- ▶ El problema es encontrar una **entrada** con una **salida** deseada, es decir, una secuencia de ciudades con una longitud óptima (mínima).



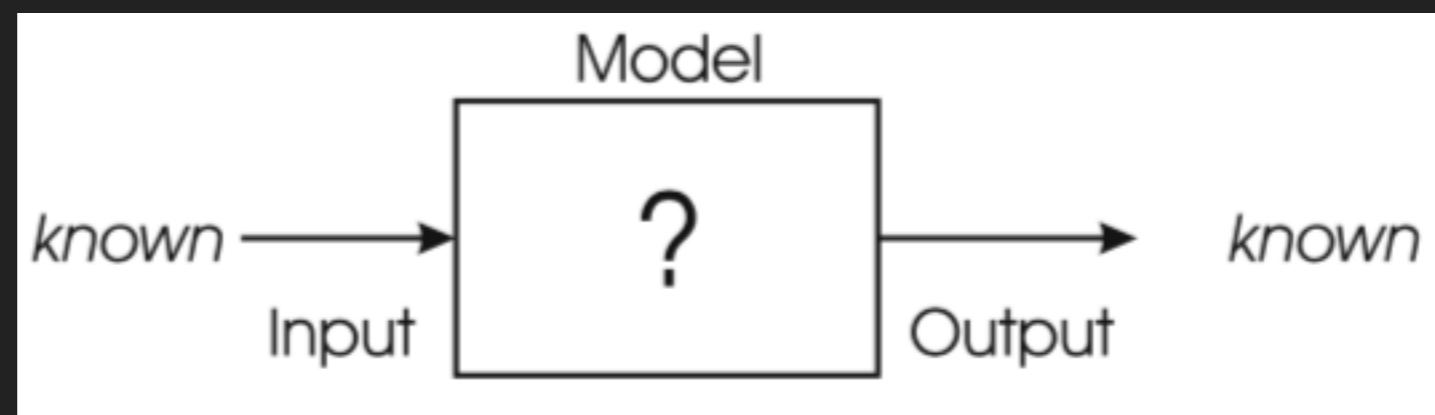
## TIPOS DE PROBLEMAS

- ▶ Optimización
- ▶ Modelado
- ▶ Simulación



## MODELADO (I)

- ▶ En un problema de **modelado** o identificación de sistemas, se conocen los conjuntos correspondientes de **entradas** y **salidas**, y se busca un **modelo** del sistema que proporcione la salida correcta para cada entrada conocida
- ▶ Se busca encontrar un **modelo** que coincida con un comportamiento conocido ante **entradas** y sus correspondiente respuesta de **salida**
- ▶ El modelo puede (o no) generalizar a ejemplos aún no vistos.



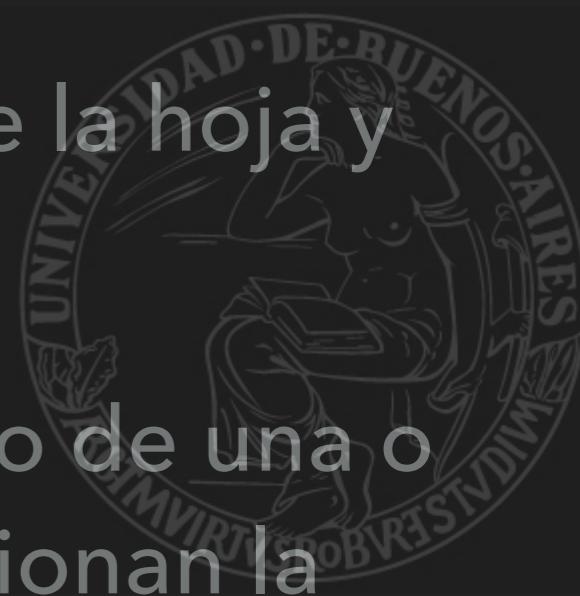
## EJEMPLO DE MODELADO (I)

- ▶ La bolsa de valores
- ▶ Las **entradas** (insumos) son los índices económicos y sociales (por ejemplo, la tasa de desempleo, el precio del oro, el tipo de cambio euro-dólar, etc.)
- ▶ La **salida** (producto) es el índice Dow Jones
- ▶ Encontrar una fórmula que vincule los insumos conocidos con los productos conocidos, representando así un **modelo** de este sistema económico



## EJEMPLO DE MODELADO (II)

- ▶ Estufa de secado de tabaco
- ▶ Las **entradas** son el tipo de hoja, la humedad de la hoja y la madurez de la hoja, acciones del operario
- ▶ El **modelo** puede representarse con un conjunto de una o mas ecuaciones lineales o no lineales que relacionan la entrada con la salida
- ▶ Las **salidas** son la temperatura y la humedad dentro de la estufa de secado



## MODELADO A OPTIMIZACIÓN

- ▶ Los problemas de **modelado** se pueden transformar en problemas de **optimización**
- ▶ ¿De qué manera se transforma?
- ▶ Primer paso: elegir la tecnología para encontrar el **modelo**  
m. Por ejemplo: una red neuronal artificial, una expresión matemática (polinomio), un árbol de decisión, etc.
- ▶ Segundo paso: es buscar la **combinación de entradas** que permitan encontrar un máximo o un mínimo en el modelo



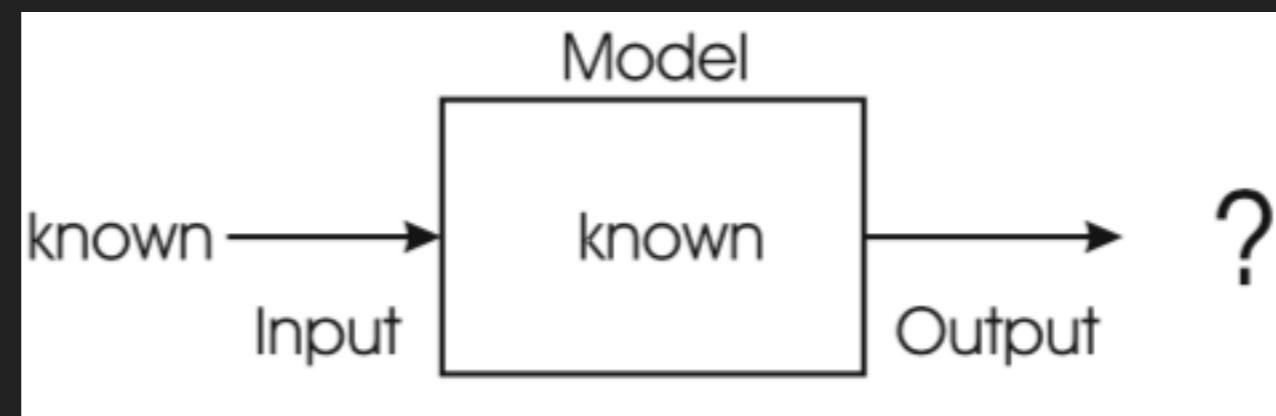
## TIPOS DE PROBLEMAS

- ▶ Optimización
- ▶ Modelado
- ▶ Simulación



# SIMULACIÓN

- ▶ En un problema de **simulación** conocemos el **modelo** del sistema y algunas **entradas**, y necesitamos calcular las **salidas** correspondientes a estas entradas



## EJEMPLOS DE SIMULACIÓN

- ▶ Ecuación Logística: Modela el crecimiento de una población.

$$\frac{dN}{dt} = rN \left(1 - \frac{N}{K}\right)$$

- ▶ Ecuaciones de Navier-Stokes: Describe el movimiento de fluidos viscosos.

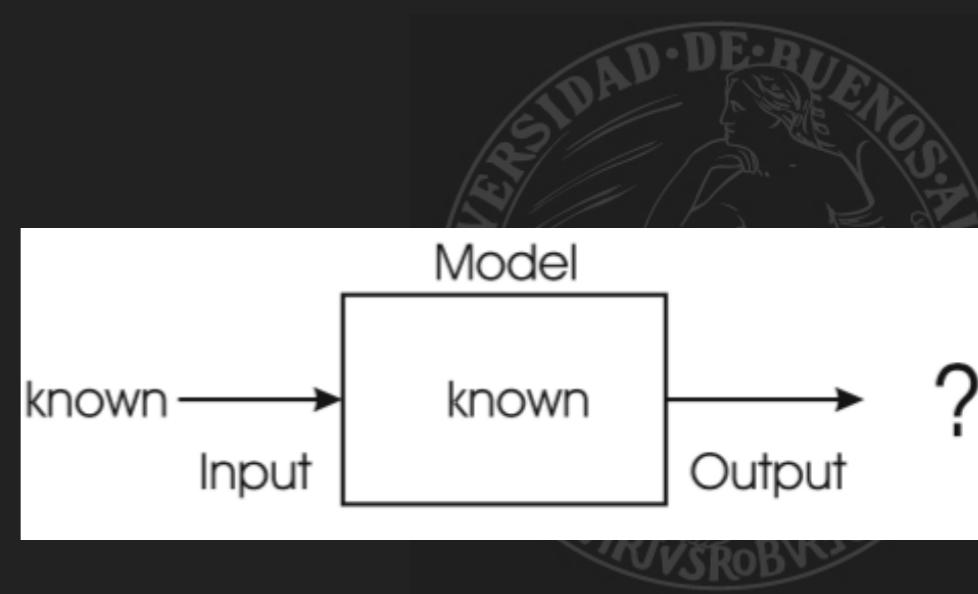
$$\rho \left( \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) = -\nabla p + \mu \nabla^2 \mathbf{v} + \mathbf{f}$$

- ▶ Ecuación del Calor: Describe cómo un fenómeno físico cambia con el tiempo.

$$\frac{\partial u}{\partial t} = D \nabla^2 u$$

- ▶ Ecuaciones de Euler-Lagrange: Describe movimiento de un sistema físico.

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} = 0$$



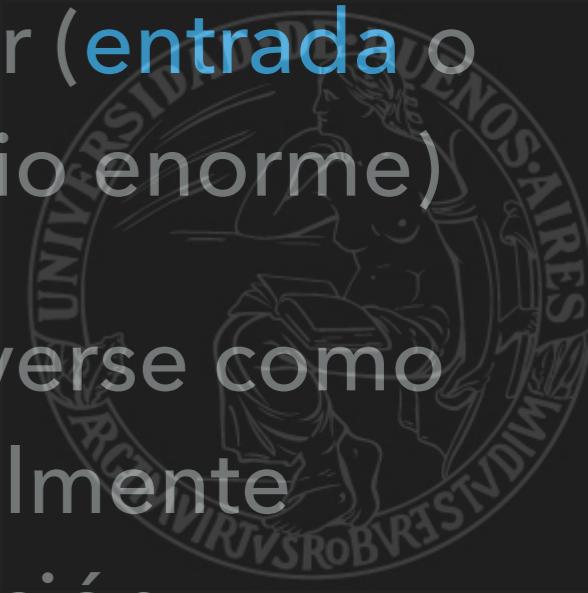
# ¿OPTIMIZACIÓN, MODELADO O SIMULACIÓN?

- ▶ Diseño de alas de avión
- ▶ Entradas: descripción de la forma de ala propuesta.
- ▶ Modelo: ecuaciones complejas de la dinámica de fluidos
- ▶ Salida: son los coeficientes de resistencia y sustentación de cualquier forma de ala



## PROBLEMAS DE BÚSQUEDA

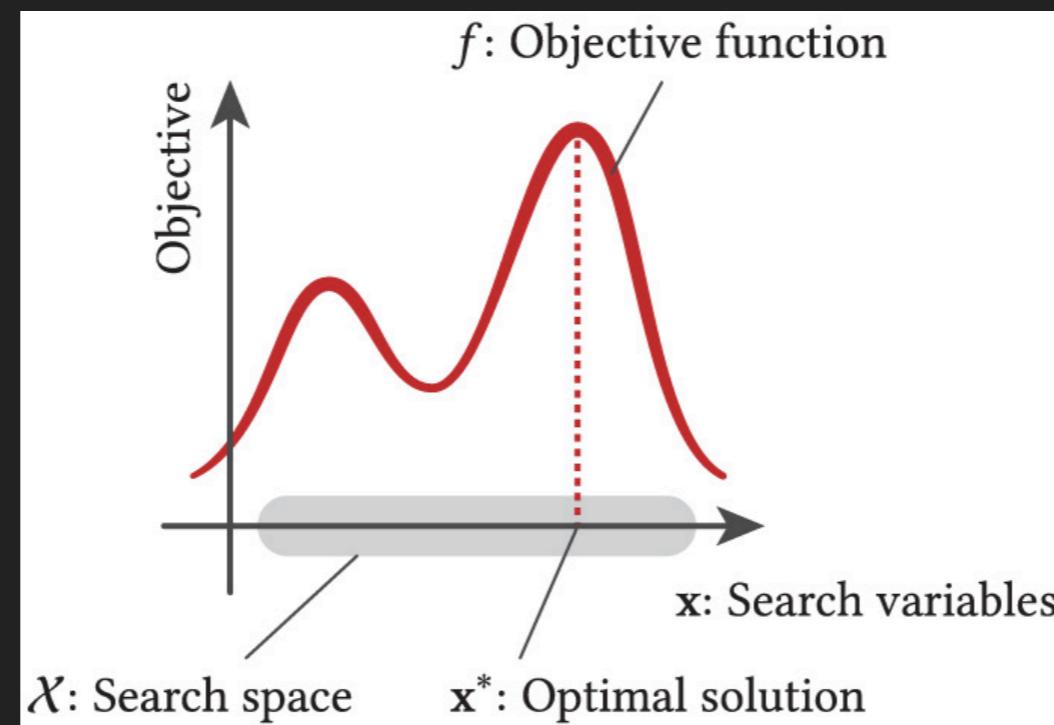
- ▶ Resolver un problema de **optimización** o de **modelado** requiere la identificación de un objeto particular (**entrada** o **modelo**) en un espacio de posibilidades (espacio enorme)
- ▶ El proceso de resolución de problemas puede verse como una **búsqueda** a través de un conjunto potencialmente enorme de posibilidades para encontrar la solución deseada.
- ▶ Este tipo de problemas se conocen como **Problemas de Búsqueda** (optimización y modelado)



# PROBLEMAS DE BÚSQUEDA

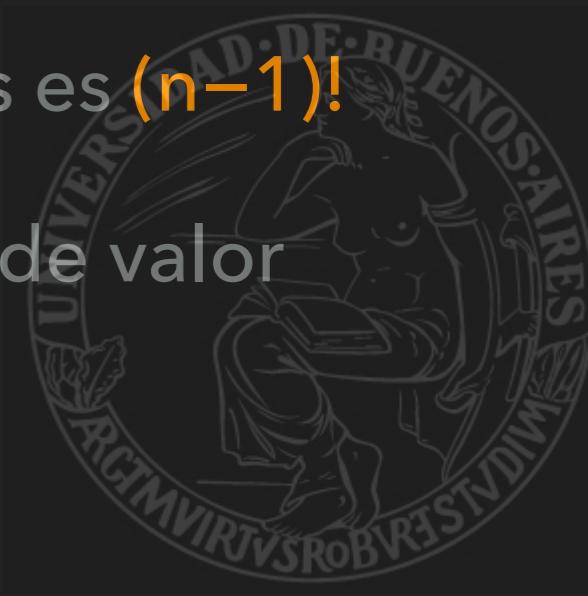
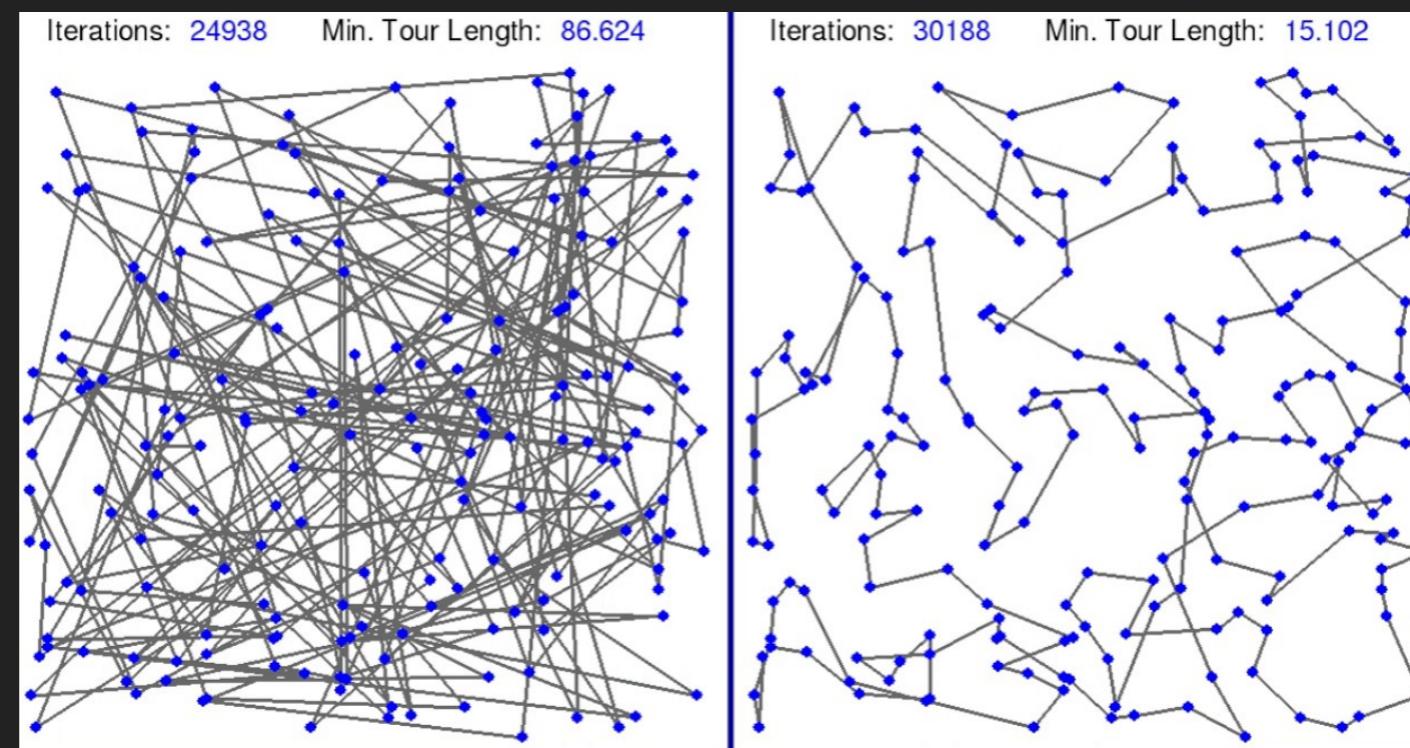
- ▶ **Espacio de búsqueda**, es la colección de todos los objetos de interés incluida la solución que buscamos

- ✓ Todas las entradas posibles a un modelo (problemas de **optimización**)
- ✓ Todos los modelos computacionales posibles que describen el fenómeno que estudiamos (problemas de **modelado**).



## PROBLEMAS DE BÚSQUEDA

- ▶ Estos **espacios de búsqueda** pueden ser muy grandes. Ejemplos:
  - ✓ el número de recorridos diferentes por **n** ciudades es **(n–1)!**
  - ✓ el número de árboles de decisión con parámetros de valor **real** es infinito.



# PROBLEMAS DE BÚSQUEDA

► Pasos para definir un problema de búsqueda:

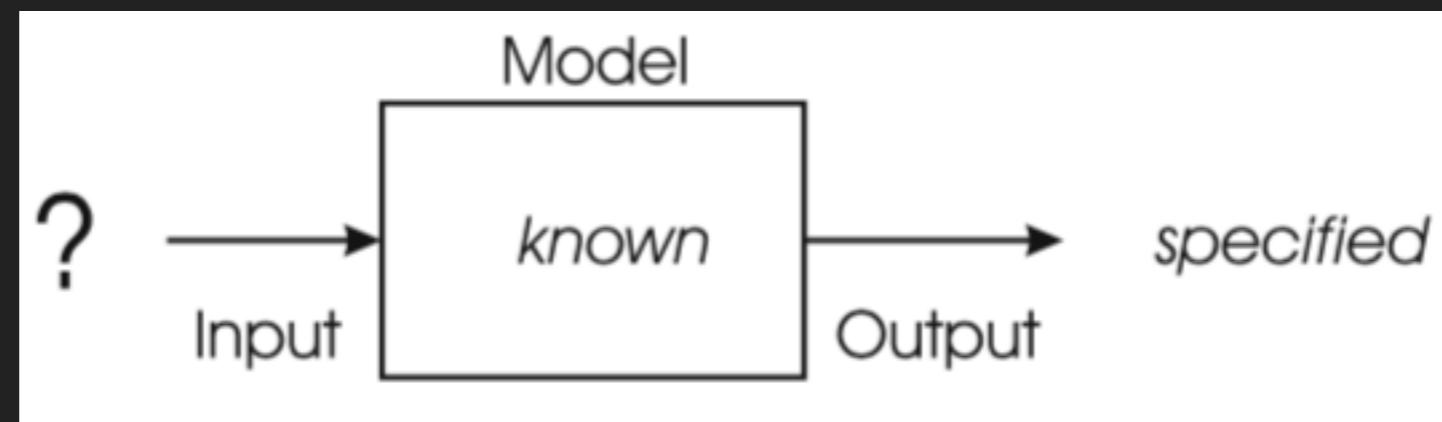
- ➡ Paso 1: especificar del **espacio de búsqueda**
- ➡ Paso 2: **definición de una solución** que se busca (es decir, ¿qué es lo que quiero resolver?)



## PROBLEMAS DE BÚSQUEDA

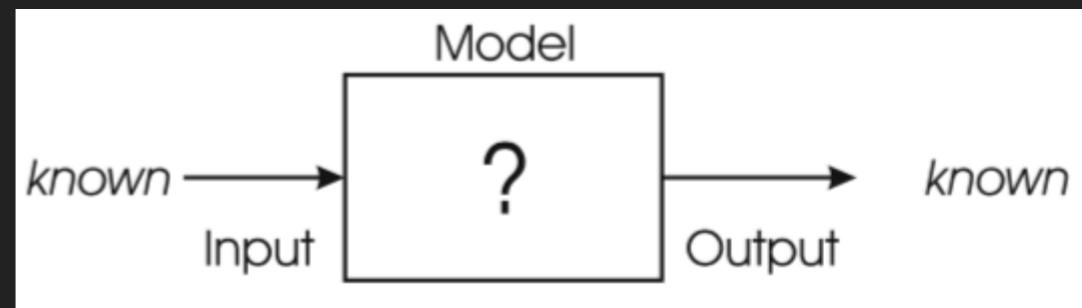
- ▶ Para problemas de optimización:

- ✓ espacio de búsqueda: todas las posibles rutas existentes entre N ciudades.
- ✓ definición de una solución: encontrar el camino más corto entre dos ciudades A y B (minimización)



## PROBLEMAS DE BÚSQUEDA

- ▶ Para problemas de **modelado**:



- ✓ **espacio de búsqueda:** todos los posibles **modelos** que podrían describir un fenómeno.
- ✓ **definición de una solución:** la propiedad del modelo que produce la salida correcta para cada entrada.



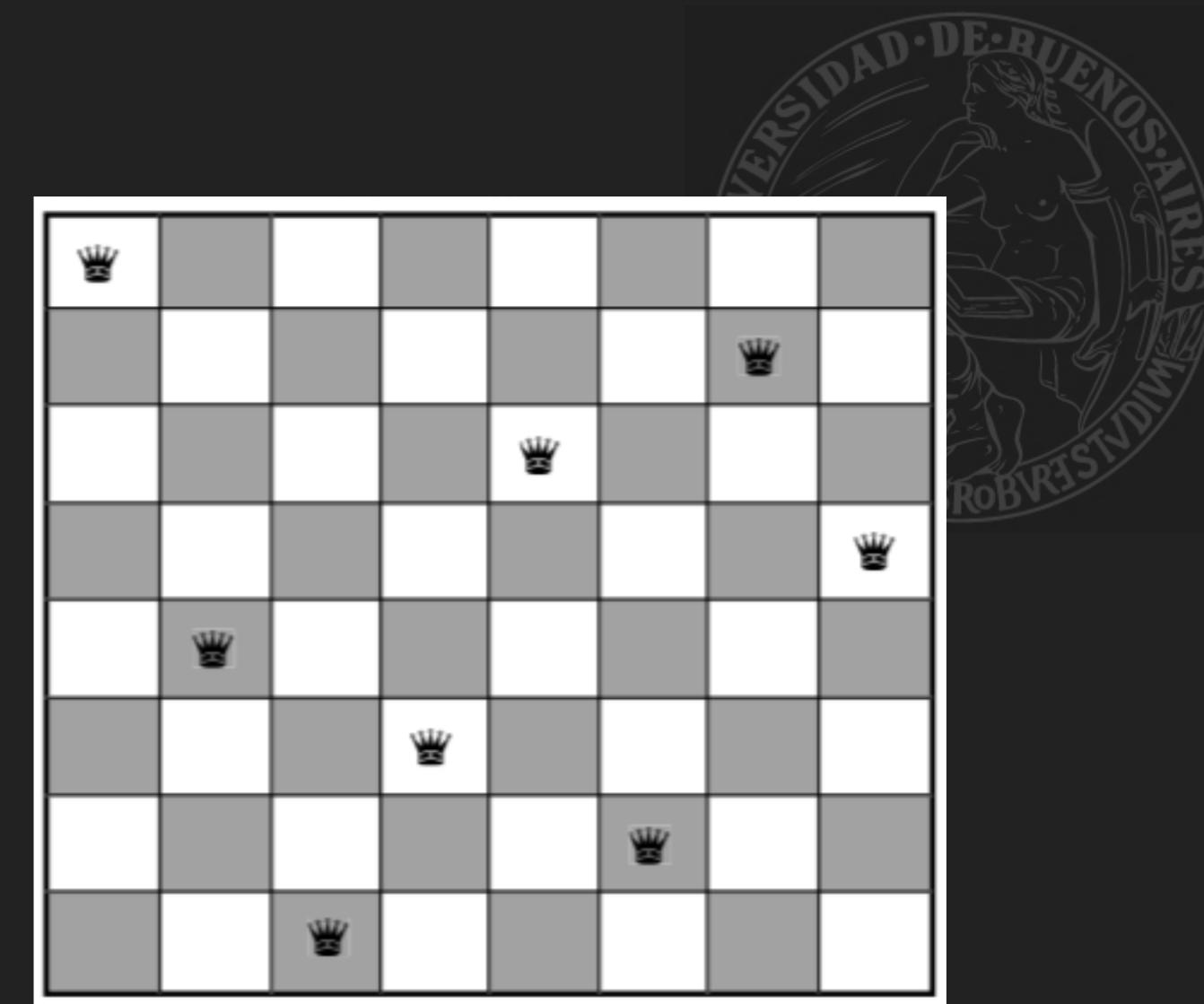
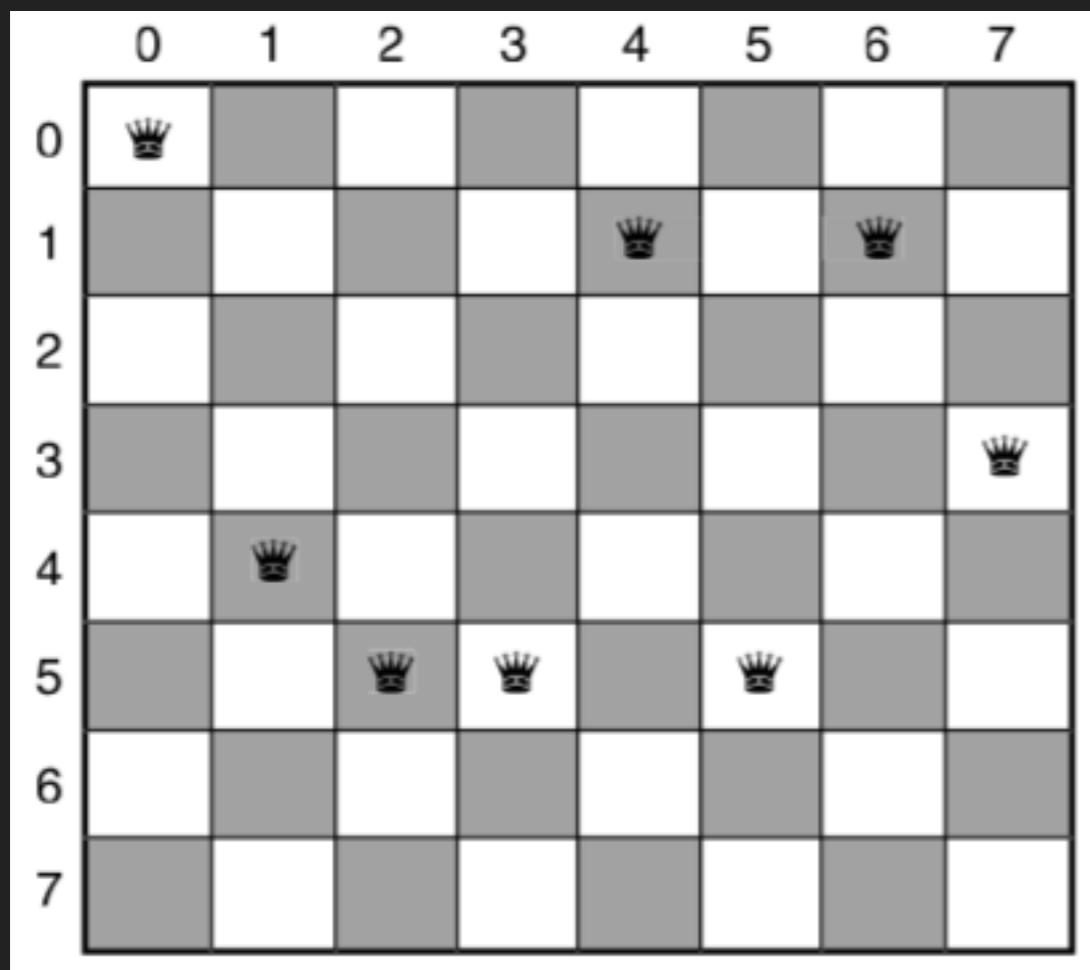
## ¿QUÉ ES LA FUNCIÓN OBJETIVO?

- ▶ Veamos primero algunos ejemplos...



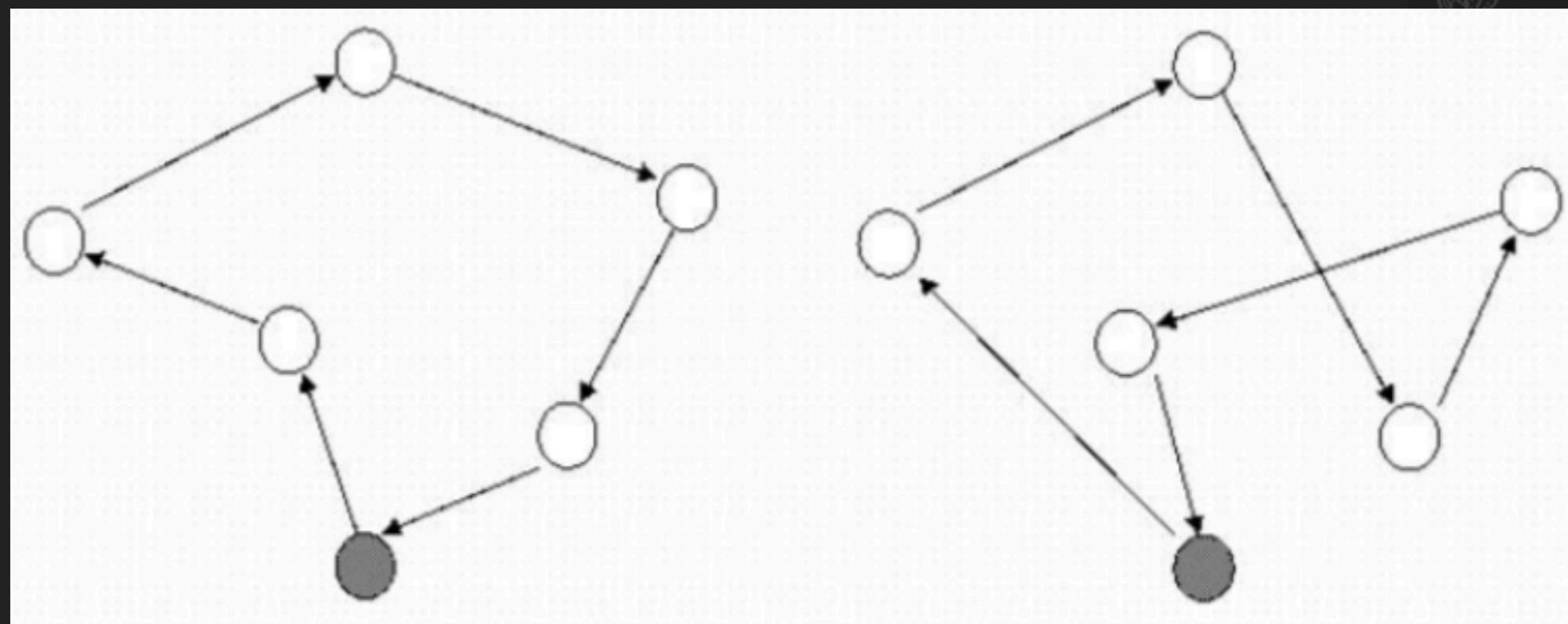
## FUNCIÓN OBJETIVO (EJEMPLO 1)

- ▶ El número de reinas sin jaquear en un tablero de ajedrez (a maximizar)



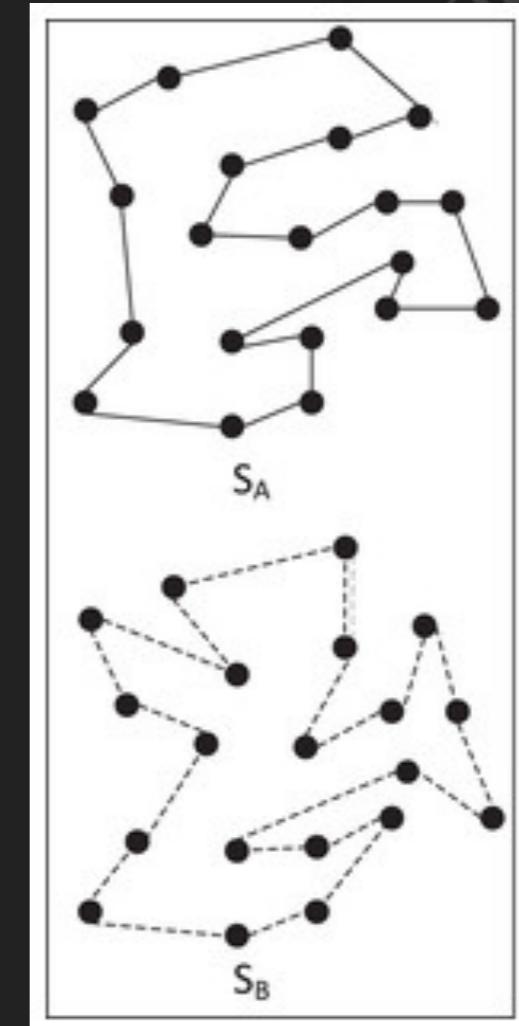
## FUNCIÓN OBJETIVO (EJEMPLO 2)

- ▶ Minimizar la duración (o la distancia) de un recorrido visitando cada ciudad exactamente una vez.
- ▶ Problema del viajante de comercio.



## FUNCIÓN OBJETIVO (EJEMPLO 2)

- ▶ Existen  $20-1!$  diferentes caminos para un problema de 20 ciudades.
- ▶  $20-1! = 1,21645100408832 \times 10^{17}$



## FUNCIÓN OBJETIVO (EJEMPLO 3)

- ▶ Función objetivo lineal genérica Z

Maximizar       $Z = 3x_1 + 5x_2,$

sujeta a

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

y

$$x_1 \geq 0, \quad x_2 \geq 0.$$



## FUNCIÓN OBJETIVO

- ▶ Las soluciones a un problema pueden identificarse en términos de **optimización** con respecto a alguna **función objetivo**.
- ▶ **Función objetivo** se entiende como una forma de asignar un valor a una posible solución que refleja su calidad en una escala



# RESTRICCIONES

- ▶ Una **restricción** es un criterio que debe cumplirse. Representa una evaluación binaria que nos dice si un requisito dado se cumple o no.
- ✓ Ejemplo 4: Encuentre una configuración de ocho reinas en un tablero de ajedrez **tal que no haya dos reinas que se controlen entre sí**.
- ✓ Ejemplo 5: Encuentre un recorrido con una duración mínima para un viajante de comercio **tal que la ciudad X se visite después de la ciudad Y**.



## FUNCIÓN OBJETIVOS Y RESTRICCIONES

- ▶ El ejemplo 2 (viajante) la solución se define puramente en términos de **optimización**.
- ▶ El ejemplo 4 (8 reinas) ilustra el caso en el que una solución se define únicamente en términos de una **restricción**: una configuración dada es buena o no.
- ▶ El ejemplo 3 (maximizar Z) expresa la solución tanto en términos de **optimización** como de **restricciones**.



# FUNCIÓN OBJETIVOS RESTRICCIONES

		Objective function	
Constraints		Yes	No
Yes	Constrained optimisation problem	Constraint satisfaction problem	
	Free optimisation problem		No problem



# FORMALIZACIÓN DE PROBLEMAS (I)

- ▶ Para desarrollar un algoritmo que resuelva problemas de optimización es necesario **formalizarlo**.



## FORMALIZACIÓN DE PROBLEMAS (II)

- ▶ Ejemplos:
- ▶ Problema de satisfacción de restricciones (**informal**):  
Coloque 8 reinas en un tablero de ajedrez de tal manera que ninguna de ellas se controle entre sí.
- ▶ Problema (**formal**): Dado un espacio de búsqueda  $S$ , definir una restricción  $\varphi$  /  $\varphi(s) = \text{verdadero} \Leftrightarrow$  no hay 2 reinas que se controlen entre sí para la configuración  $s$ .



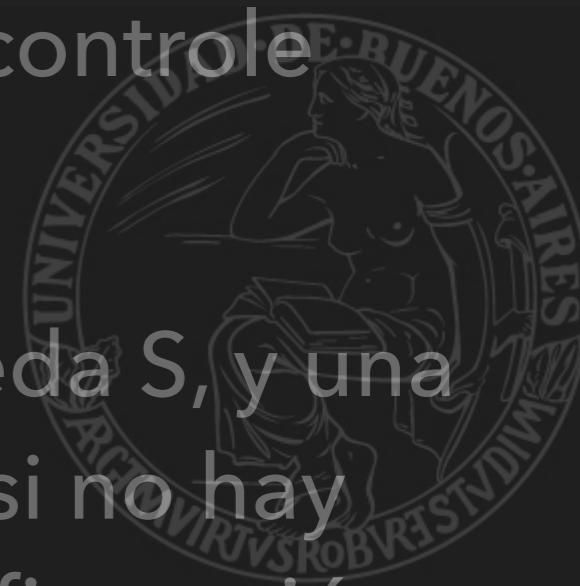
## FORMALIZACIÓN DE PROBLEMAS (III)

- ▶ Problema de optimización (**informal**): Coloque 8 reinas en un tablero de ajedrez de tal manera que ninguna de ellas se controle entre sí.
- ▶ Problema (**formal**): Dado un espacio de búsqueda  $S$  (siendo  $S$  el conjunto de todas las configuraciones del tablero con ocho reinas) con una función objetivo  $f$  (la cual informa el número de reinas libres para una configuración dada), definir un solución  $s / s \in S$  con  $f(s) = 8$ .



# FORMALIZACIÓN DE PROBLEMAS

- ▶ Problema (**informal**): Coloque ocho reinas en un tablero de ajedrez de tal manera que ninguna de ellas se controle entre sí.
- ▶ Problema (**formal**): Dado un espacio de búsqueda  $S$ , y una restricción  $\varphi$  tal que  $\varphi(s) = \text{verdadero si y sólo si no hay dos reinas que se controlen entre sí para la configuración } s.$

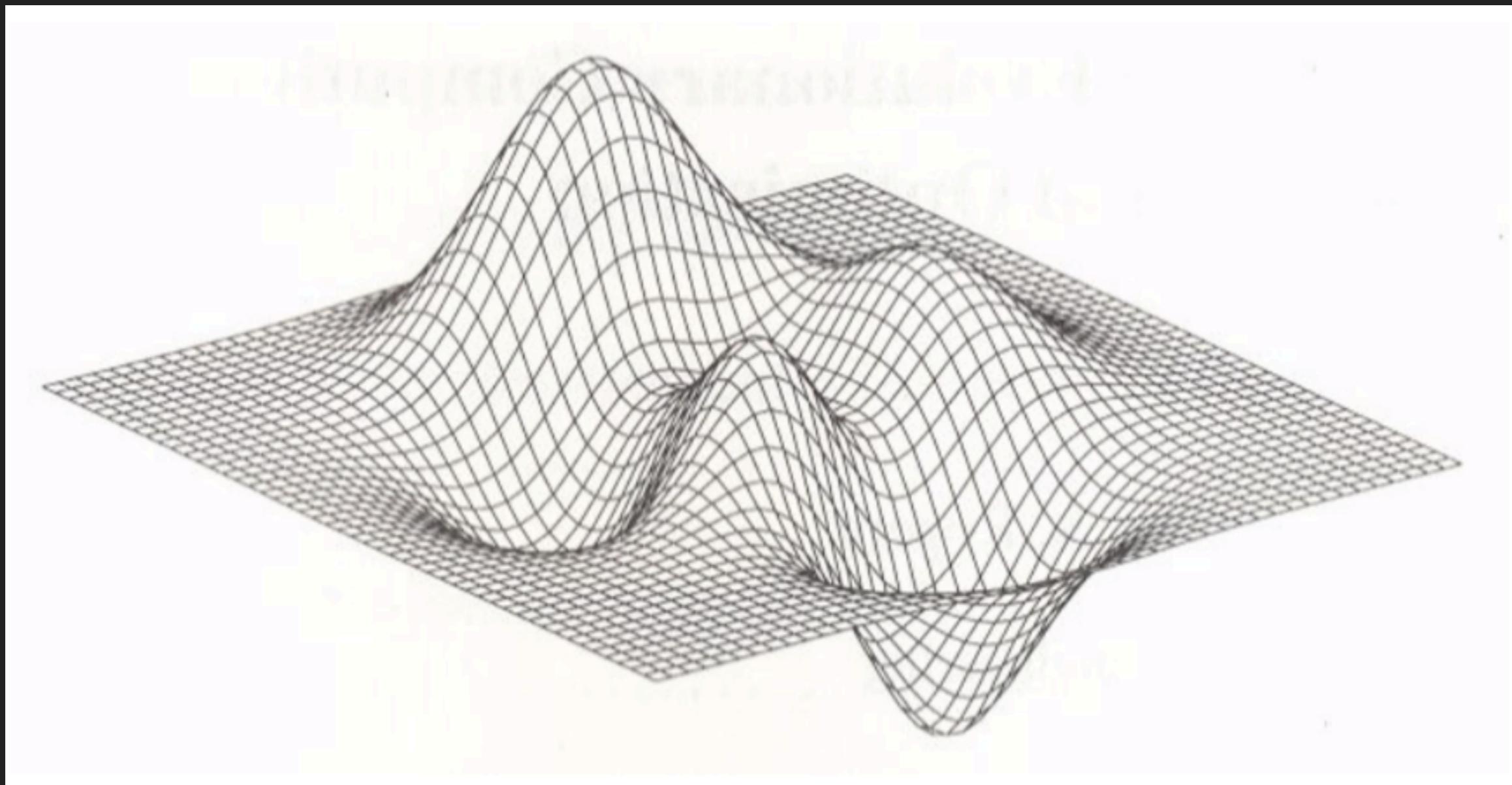


# FORMALIZACIÓN DE PROBLEMAS

- ▶ Si  $S \in \mathbb{R} \Rightarrow$  problema de **optimización numérica**.
- ▶ Si  $S \in \mathbb{Z} \Rightarrow$  problema de **optimización combinatoria**.
- ▶ Si  $S \in \{\text{true, false}\} \Rightarrow$  problema de **optimización combinatoria**.

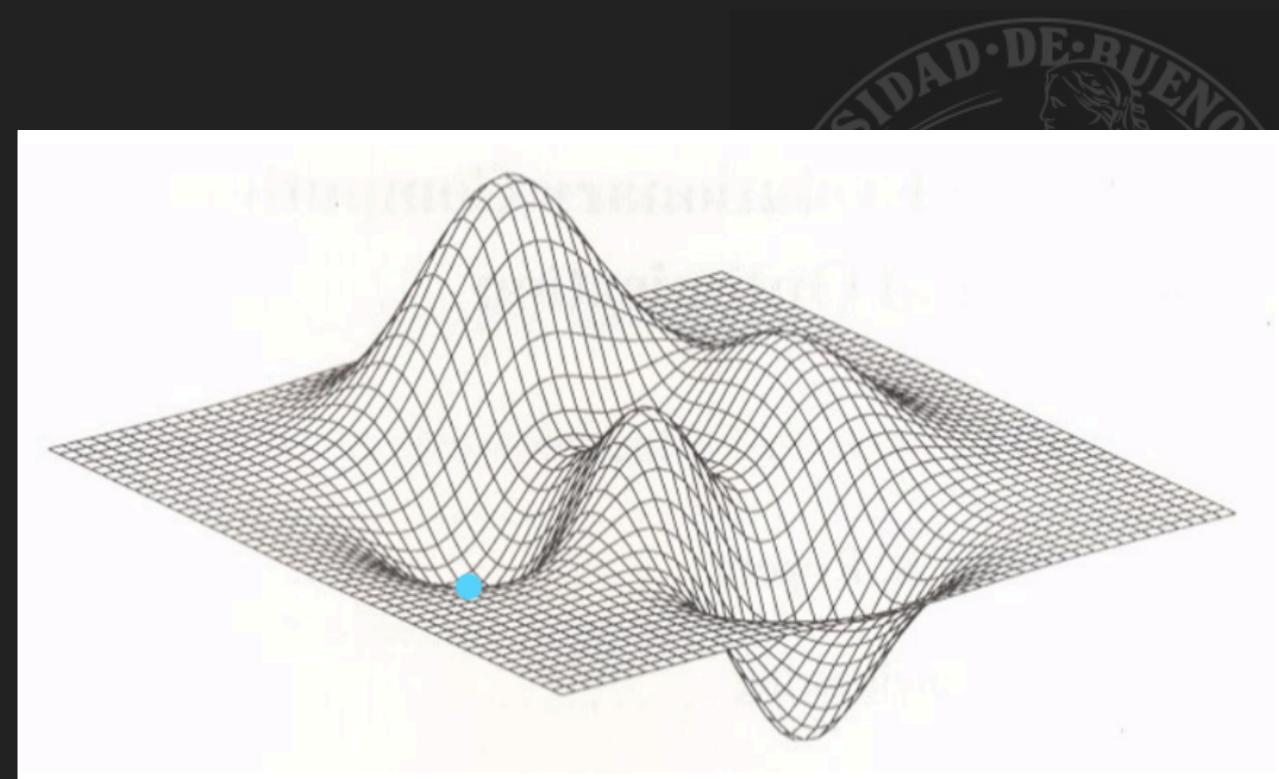


# ÓPTIMOS LOCALES Y GLOBALES



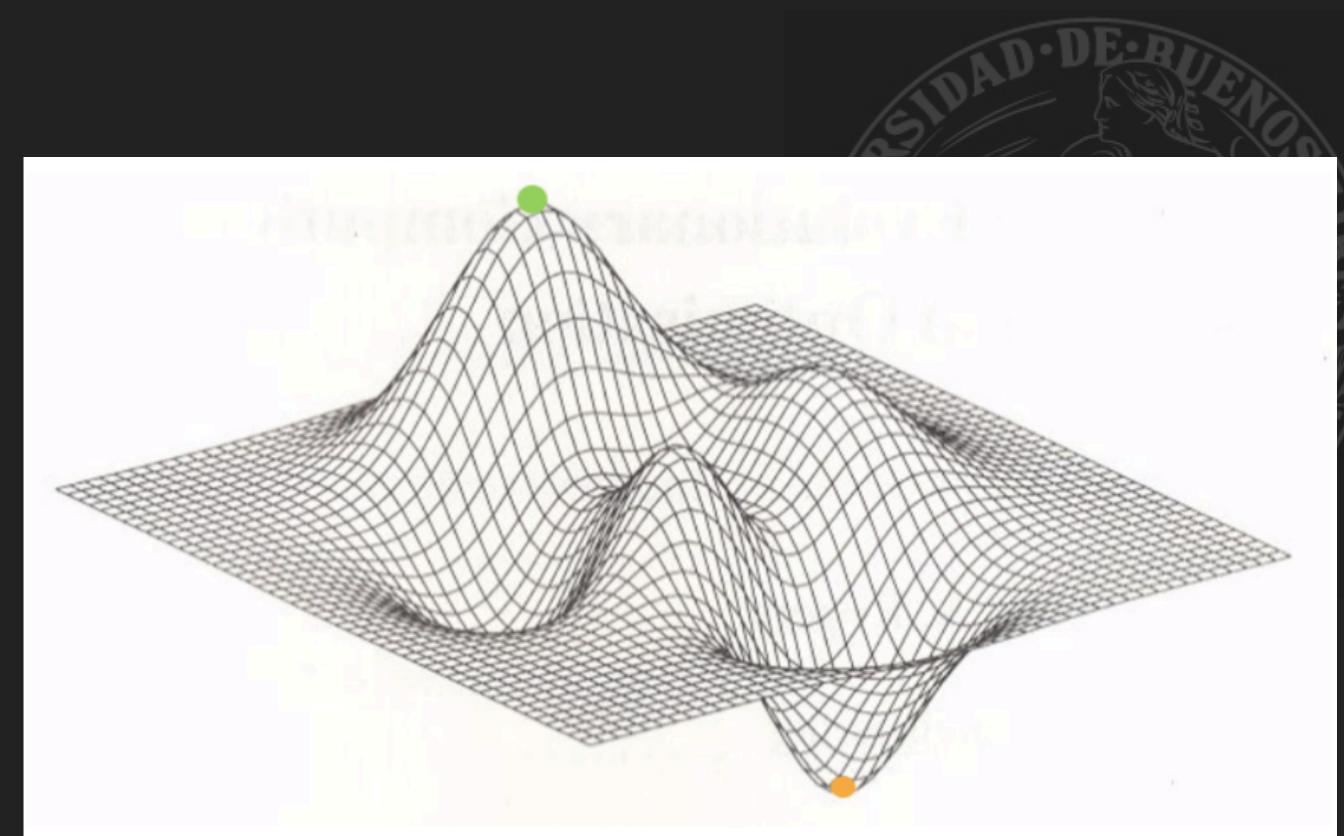
# ÓPTIMOS LOCALES Y GLOBALES

- ▶ Un óptimo local (o mínimo/máximo local) es un punto en el espacio de búsqueda donde la función objetivo tiene el mejor valor posible en una vecindad específica de ese punto.
- ▶ No es necesariamente el mejor valor posible para todo el espacio de búsqueda, pero es óptimo dentro de su entorno local.



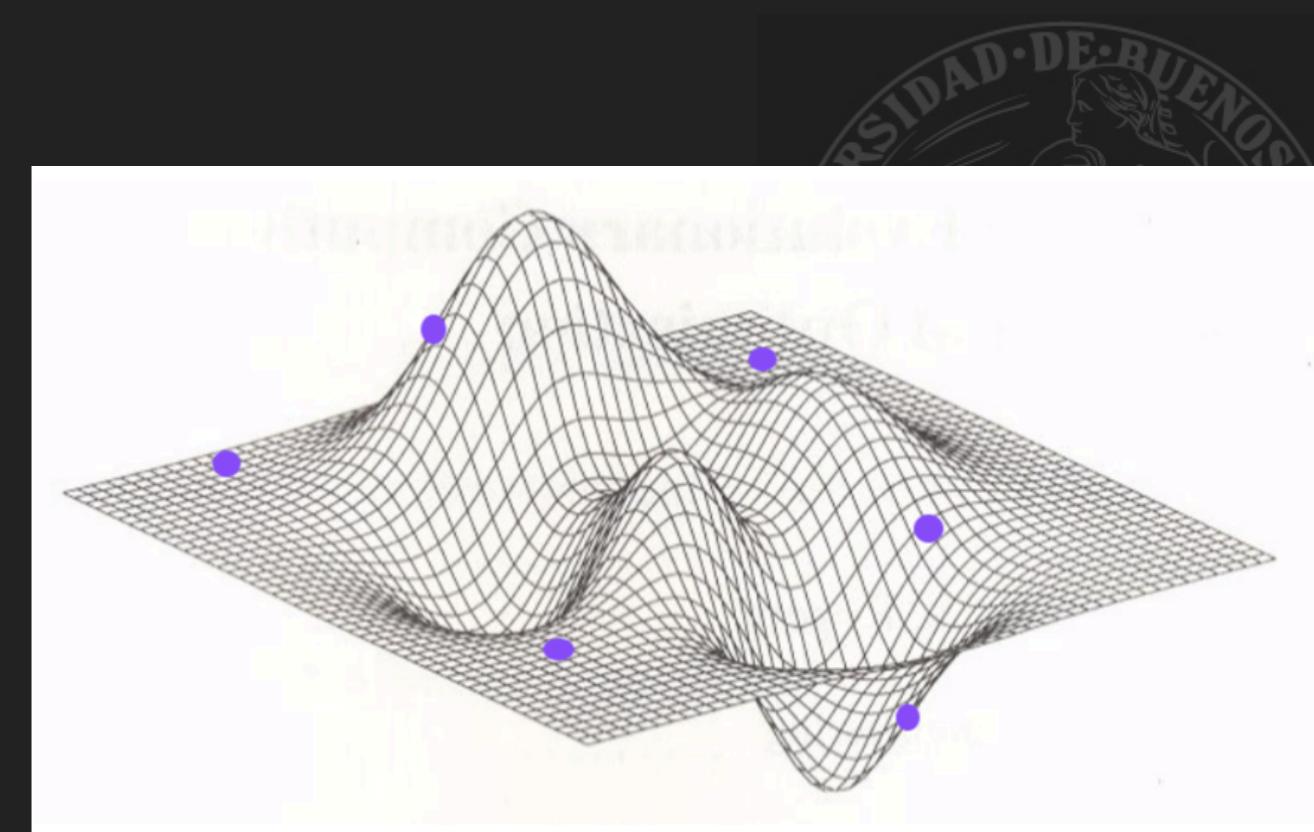
## ÓPTIMOS LOCALES Y GLOBALES

- ▶ El óptimo global (o mínimo/máximo global) es el punto en el espacio de búsqueda donde la función objetivo tiene el mejor valor posible en todo el dominio de la función.
- ▶ Representa la solución óptima para todo el problema.



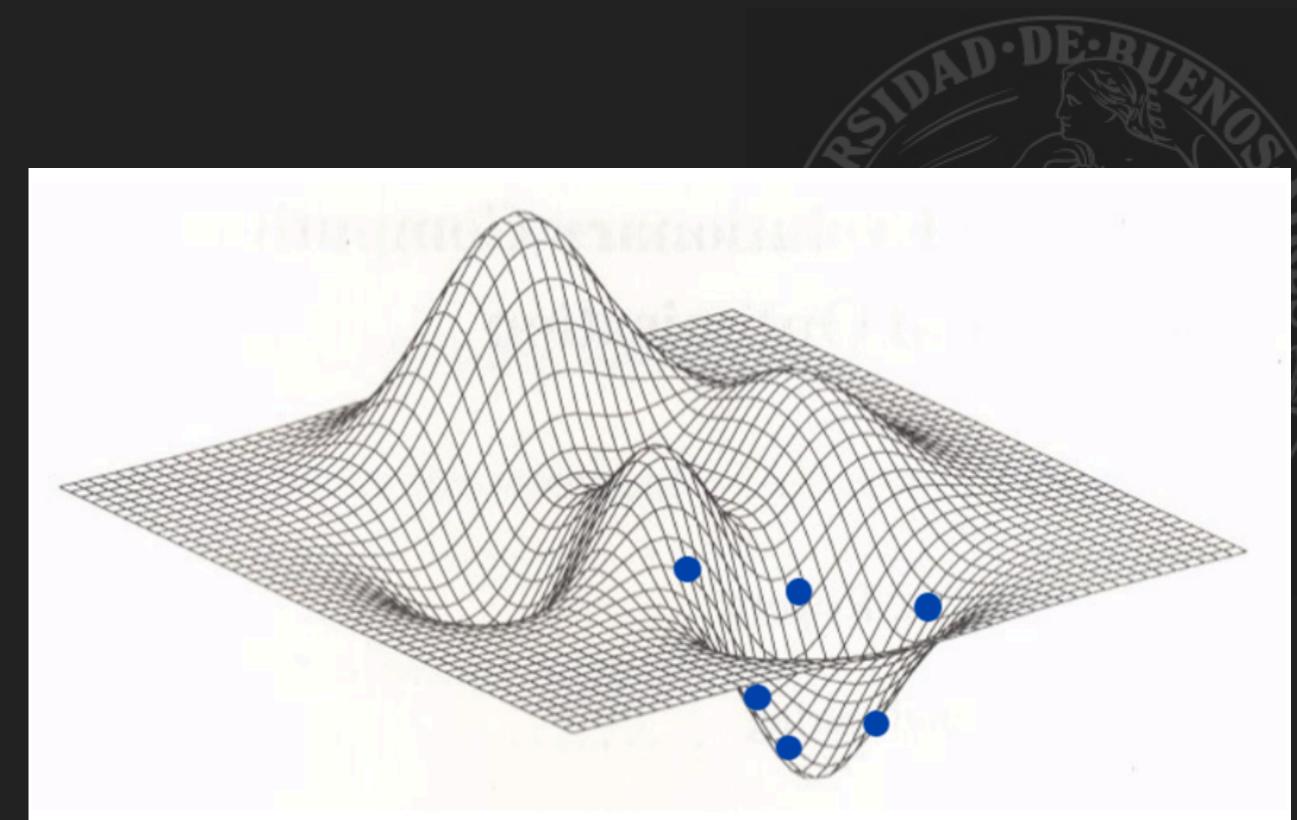
# EXPLORACIÓN Y EXPLOTACIÓN

- ▶ La **exploración** es el proceso de búsqueda amplia en el espacio de búsqueda para descubrir diferentes regiones que pueden contener soluciones prometedoras.
- ▶ Implica diversificar la búsqueda para encontrar nuevas áreas que puedan contener mejores soluciones, incluidos **óptimos locales y globales**.



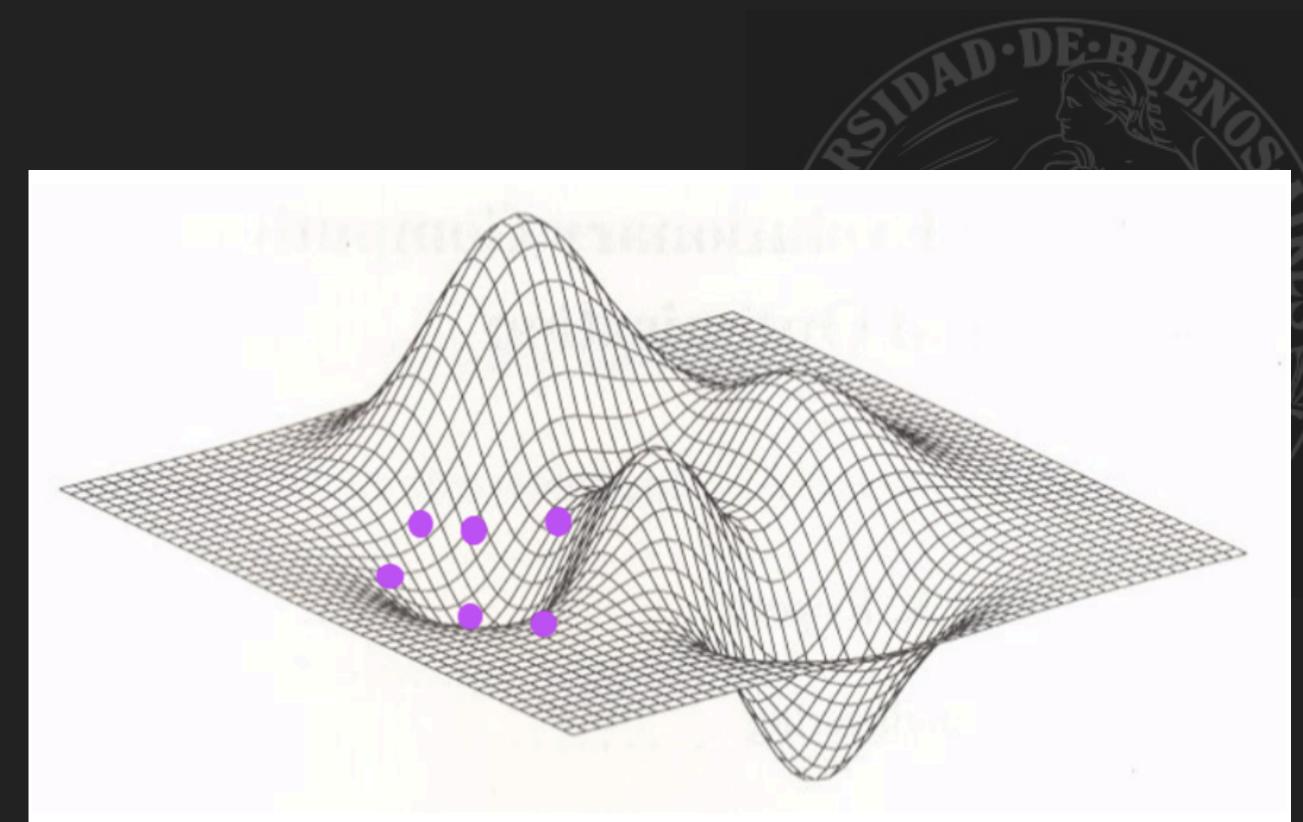
# EXPLORACIÓN Y EXPLOTACIÓN

- ▶ La **explotación** es el proceso de centrarse y mejorar soluciones que ya se sabe que son buenas o prometedoras.
- ▶ Implica intensificar la búsqueda de soluciones conocidas, refinándolas para mejorar potencialmente su calidad y su cercanía al óptimo global.



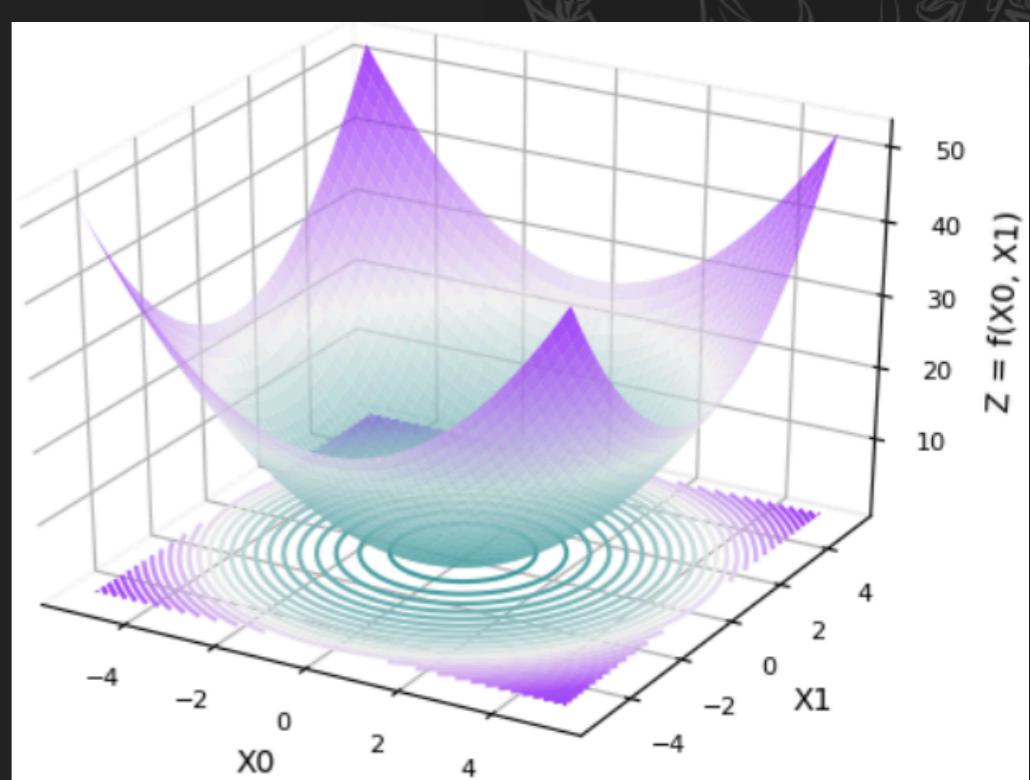
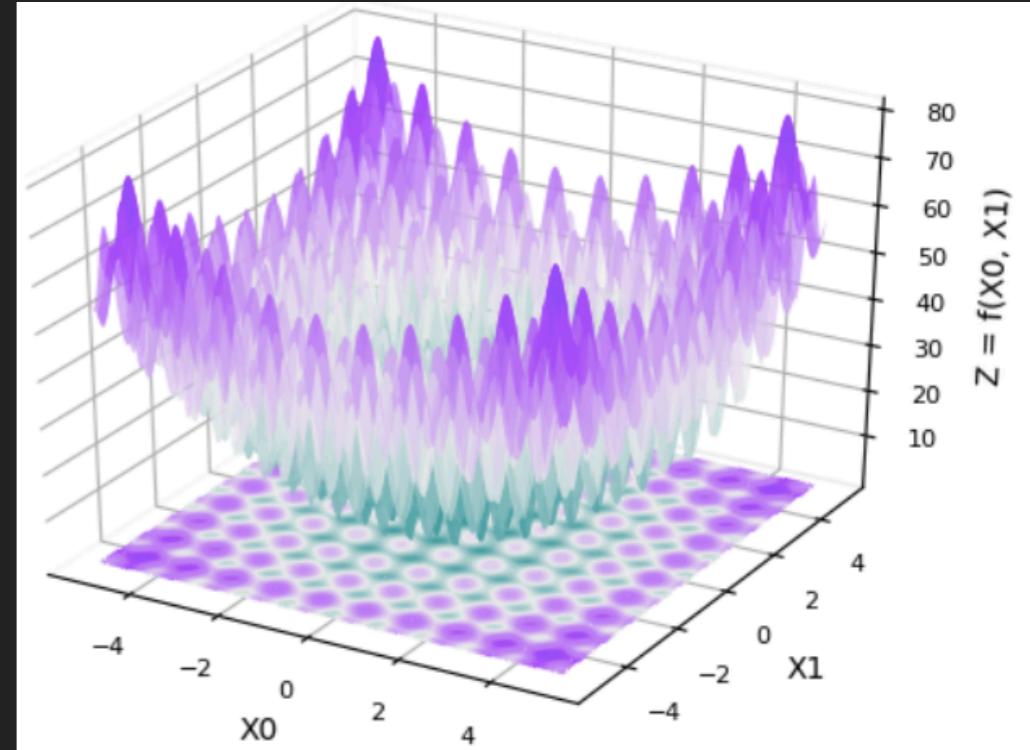
# CONVERGENCIA TEMPRANA

- ▶ La **convergencia temprana** ocurre cuando un algoritmo de optimización se decide prematuramente por una solución sin explorar completamente el espacio de búsqueda.
- ▶ Esto puede suceder cuando el algoritmo converge rápidamente a un **óptimo local** y no logra explorar otras soluciones potencialmente mejores en otras partes del espacio de búsqueda.



# ÓPTIMOS

- ▶ Los problemas multimodales son aquellos problemas de optimización donde la función objetivo tiene **múltiples óptimos** (o modos) locales distribuidos por todo el espacio de búsqueda.
- ▶ Los problemas unimodales tienen un **único óptimo global** y ningún óptimo local.



# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

► Usando métodos de optimización tradicionales:

- ✓ Derivada primera
- ✓ Búsqueda exhaustiva (Exhaustive Search)
- ✓ Caminata aleatoria (Random walk)
- ✓ Descenso por Gradiente (Gradient Descent)



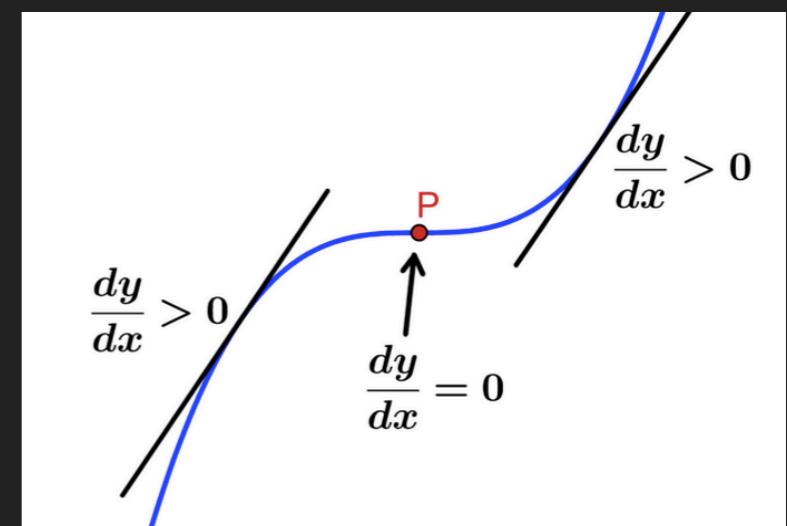
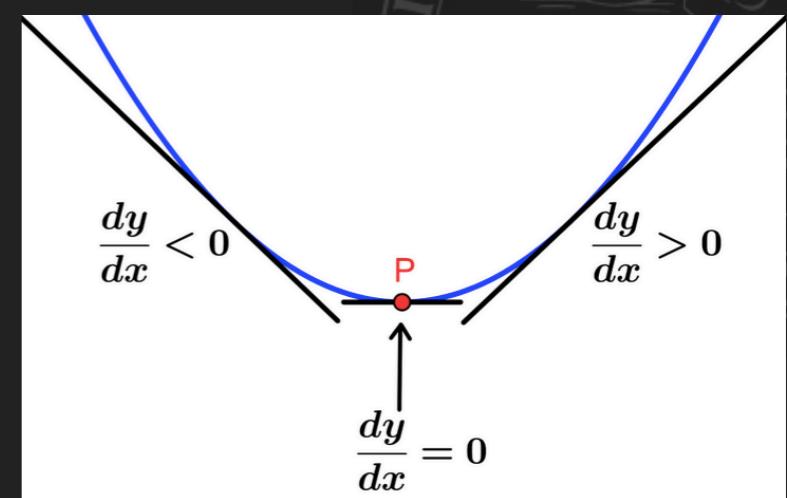
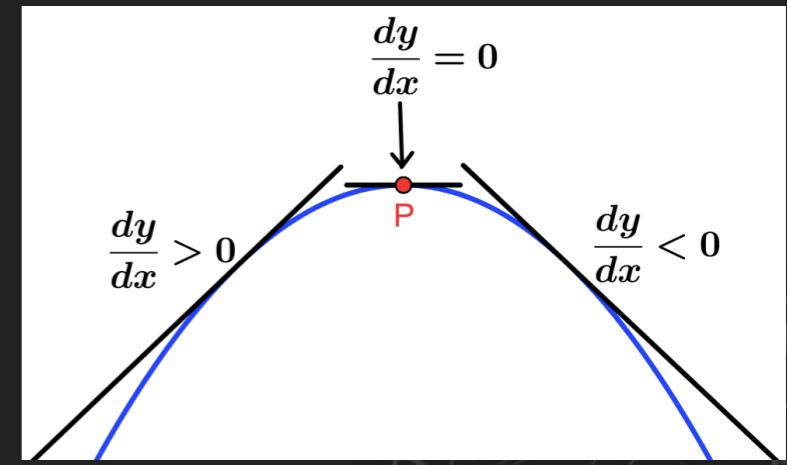
# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- ▶ Derivada primera
- ▶ Búsqueda exhaustiva (Exhaustive Search)
- ▶ Caminata aleatoria (Random walk)
- ▶ Descenso por Gradiente (Gradient Descent)



# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- ▶ **Opción 1: Derivada primera**
- ▶ Consideremos que "y" es la función de una sola variable "x", es decir,  $y=f(x)$ .
- ▶ Si la **primera derivada** de esta función, es decir,  $f'(x)$  se vuelve igual a cero, es decir,  $f'(x)=0$ , en un punto  $x = x^*$ , decimos que **puede** existir un:
  - ✓ Óptimo (mínimo o máximo)
  - ✓ Punto de inflexión que se encuentra en ese punto.



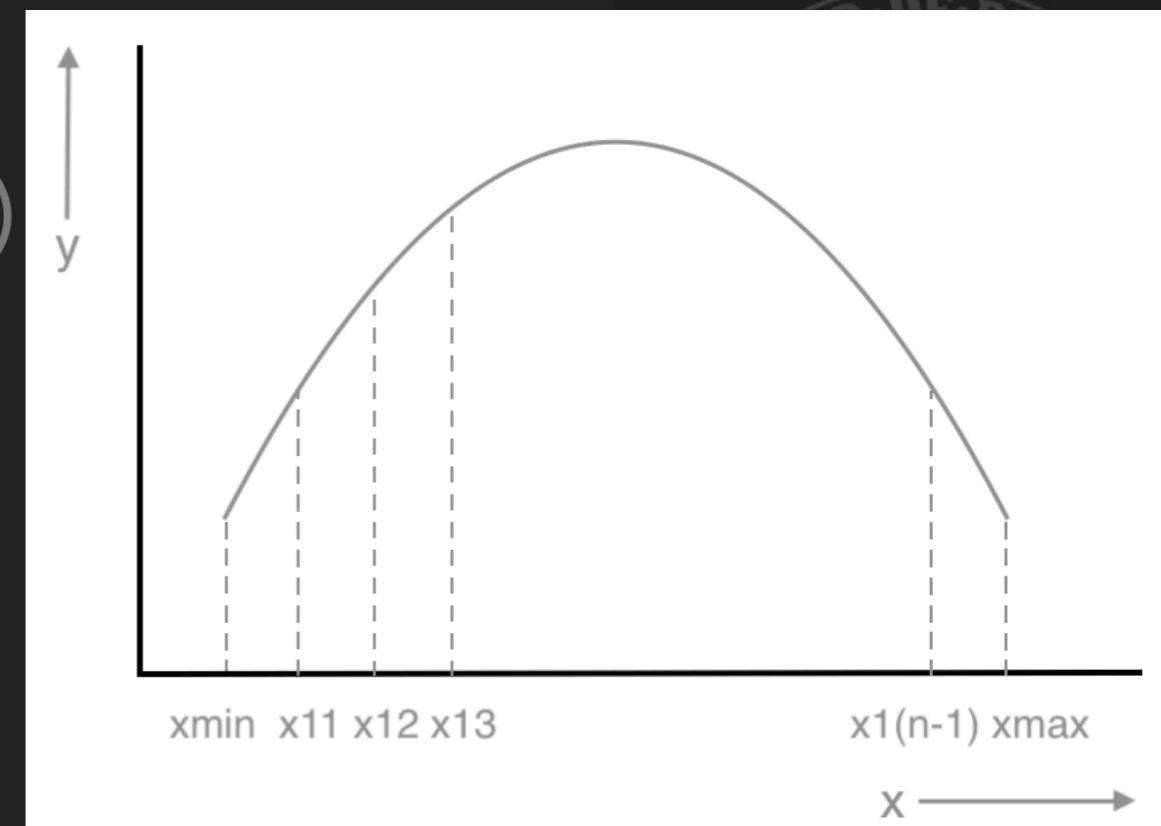
# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- ▶ Derivada primera
- ▶ Búsqueda exhaustiva (Exhaustive Search)
- ▶ Caminata aleatoria (Random walk)
- ▶ Descenso por Gradiente (Gradient Descent)



# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- ▶ **Opción 2: Búsqueda exhaustiva**
- ▶ Consideremos una función unimodal "y" (tiene un solo pico) de una variable "x" definida en el rango  $[x_{\min}, x_{\max}]$  donde  $x_{\min}$  y  $x_{\max}$  son los límites inferior y superior de la variable x, respectivamente.



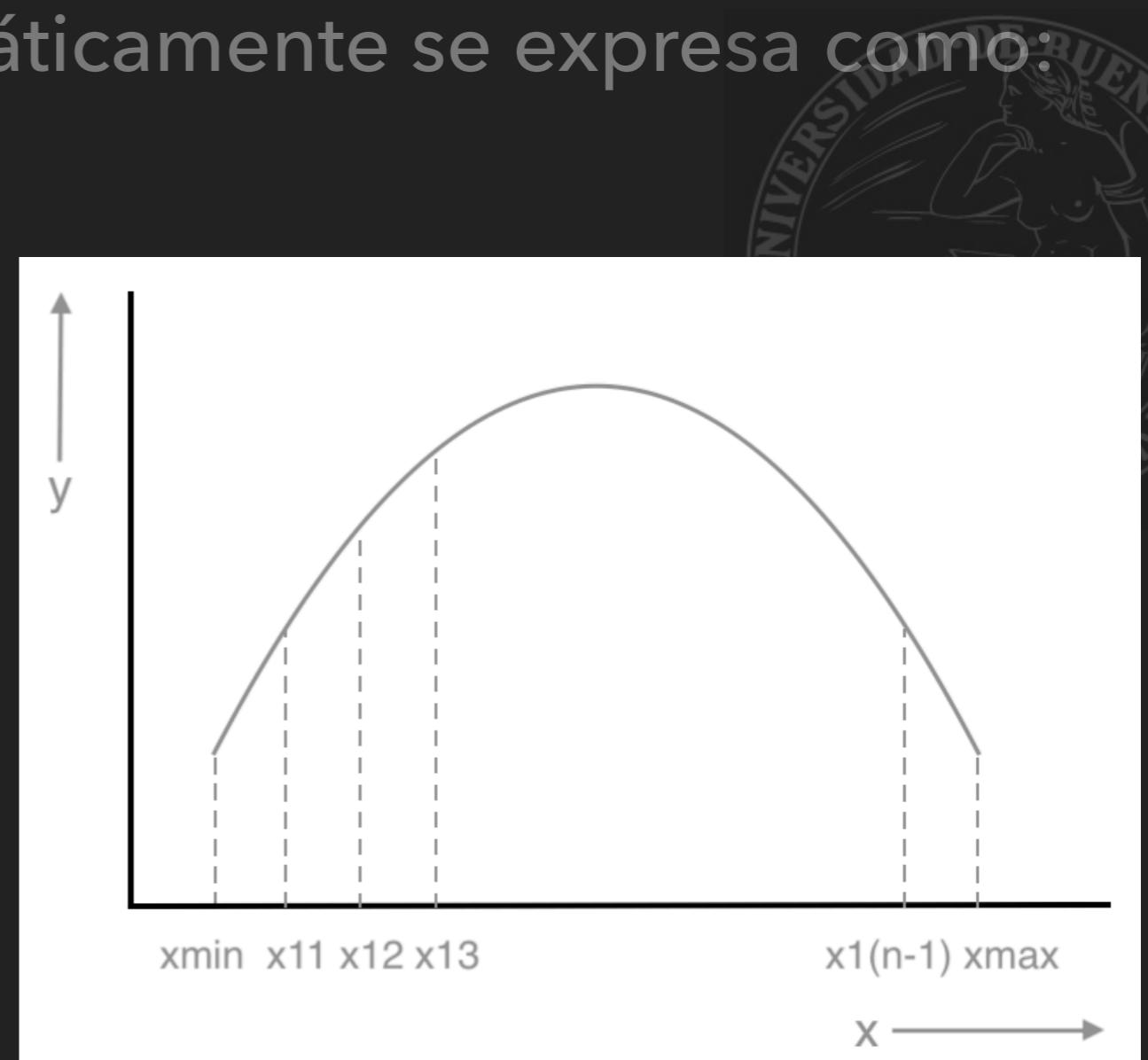
# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- Si el objetivo es maximizar la función  $y = f(x)$  en el rango citado, el problema matemáticamente se expresa como:

**Maximizar  $y = f(x)$**

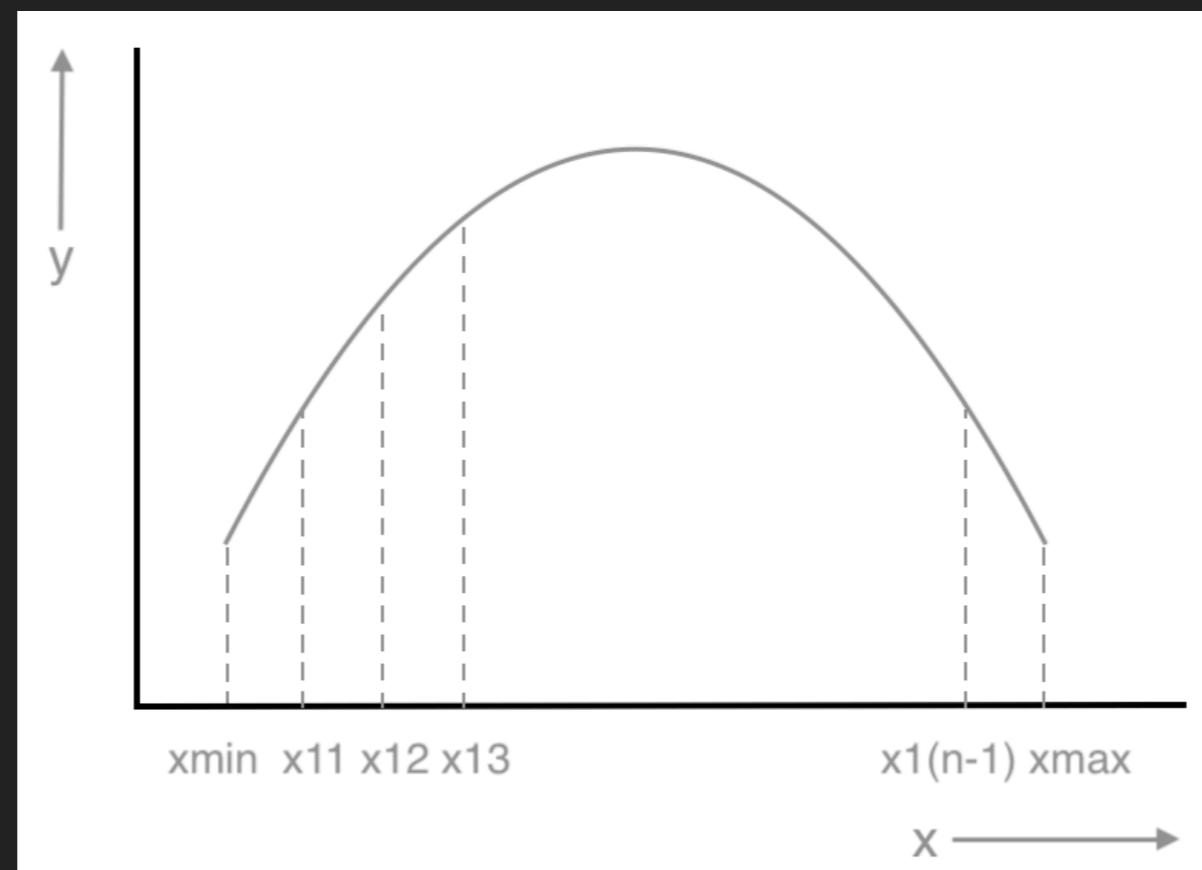
Sujeto a:

$$x_{\min} \leq x \leq x_{\max}$$



# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- Si el rango de la variable  $x$  que es  $(x_{\max} - x_{\min})$  es dividido en  $n$  partes iguales, un pequeño cambio en  $x$ , esto es;  $\Delta x$  estará dado por la expresión  $(x_{\max} - x_{\min})/n$



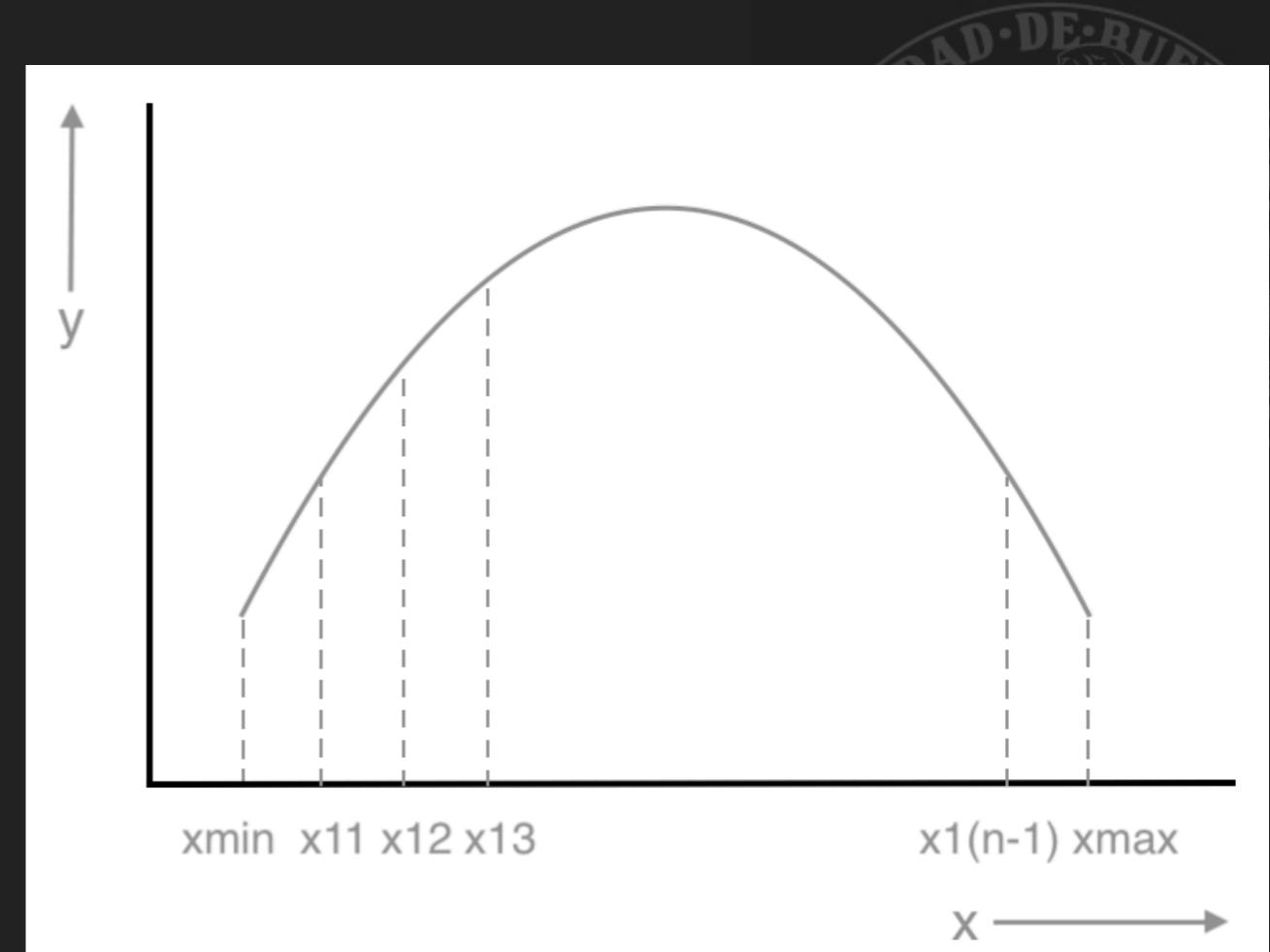
# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- ▶ Pasos para encontrar el máximo de la función:
- ▶ **Paso 1:** Obtener  $x_1, x_2$  y  $x_3$

$$x_1 = x_{\min}$$

$$x_2 = x_1 + \Delta x = x_{11}$$

$$x_3 = x_2 + \Delta x = x_{12}$$



# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- ▶ **Paso 2:** Calculamos los valores de la función, esto es,  $f(x_1)$ ,  $f(x_2)$ ,  $f(x_3)$  y verificamos el punto máximo

Si  $f(x_1) \leq f(x_2) \Rightarrow f(x_3)$

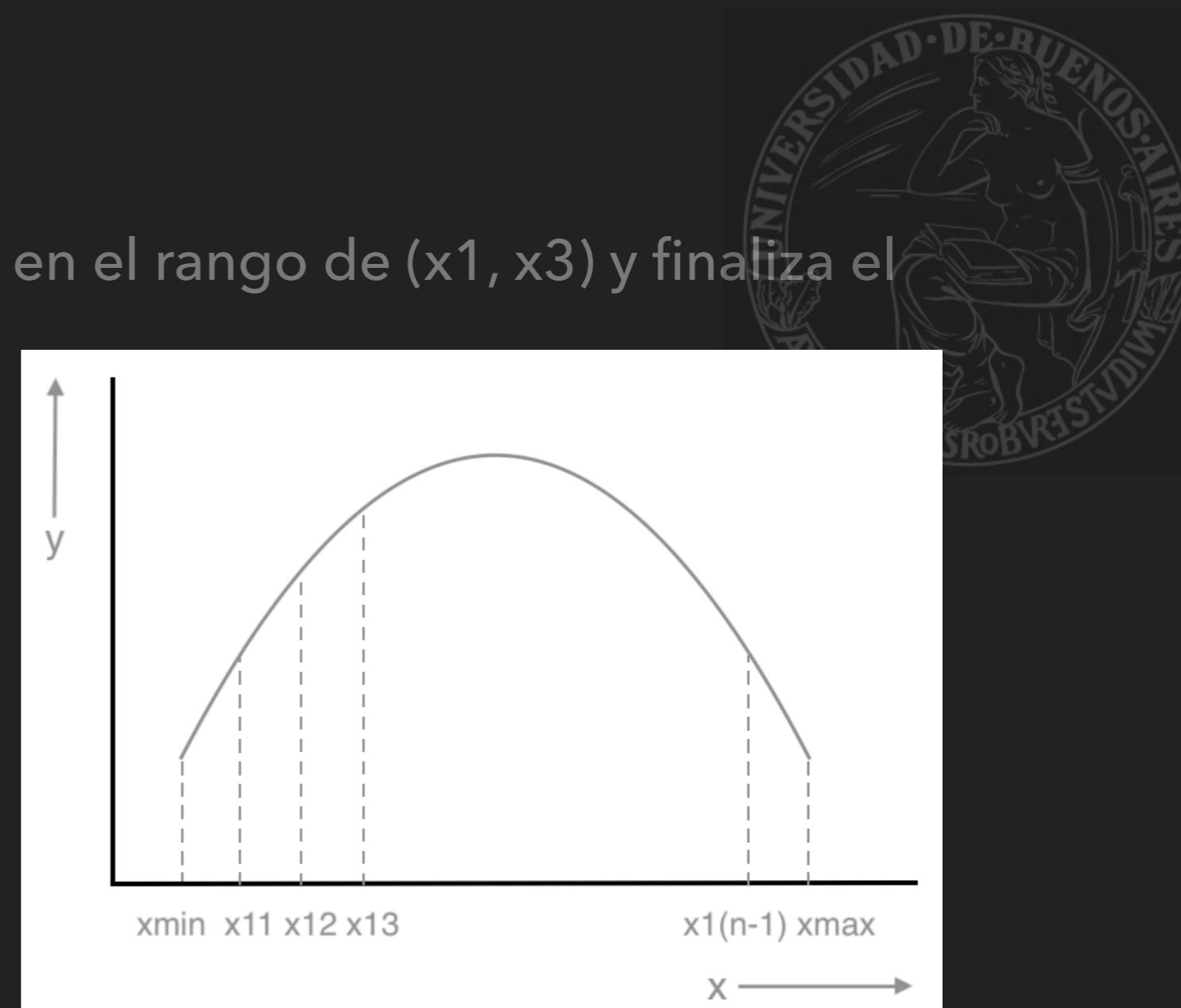
El punto máximo se encuentra en el rango de  $(x_1, x_3)$  y finaliza el algoritmo

Sino

$x_1 = x_2$  (anterior)

$x_2 = x_3$  (anterior)

$x_3 = x_2$ (actual) +  $\Delta x$



# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

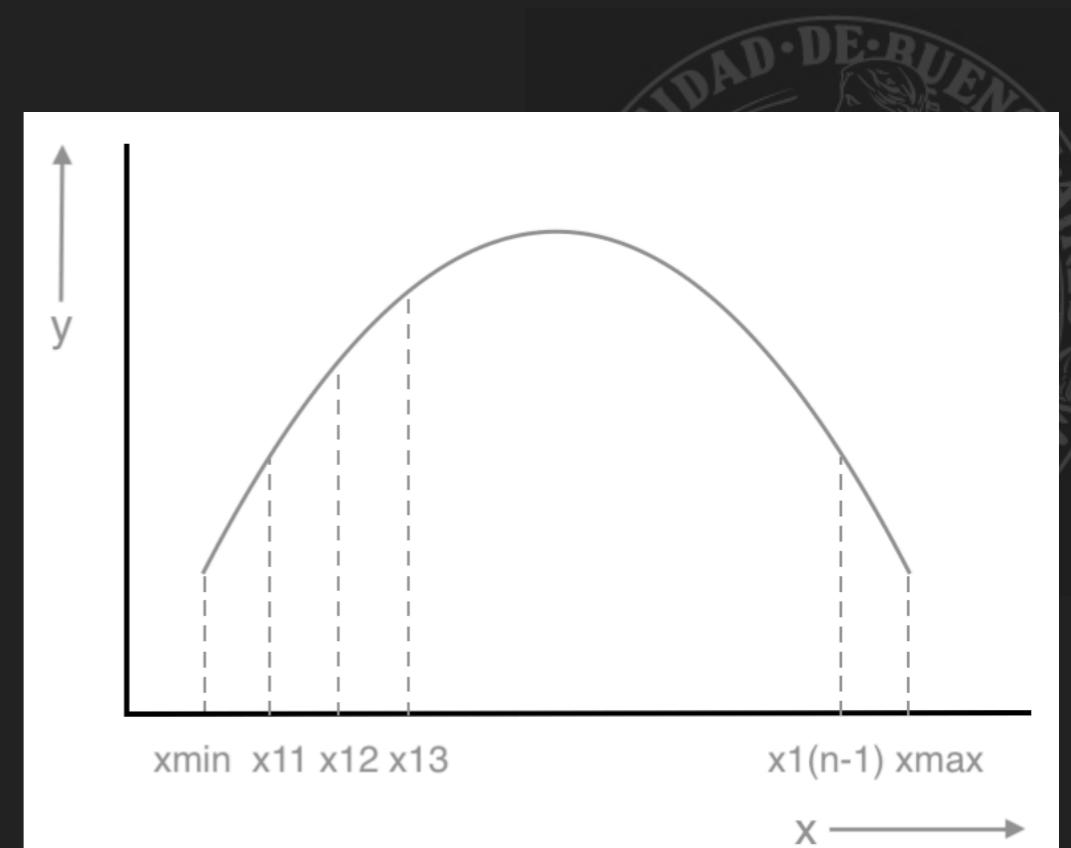
- ▶ **Paso 3:** Comprobamos si  $x_3$  excede  $x_{\max}$

Si  $x_3 \leq x_{\max}$

Ir al paso 2

Sino

Decimos que el máximo no está en el rango de  $(x_{\min}, x_{\max})$



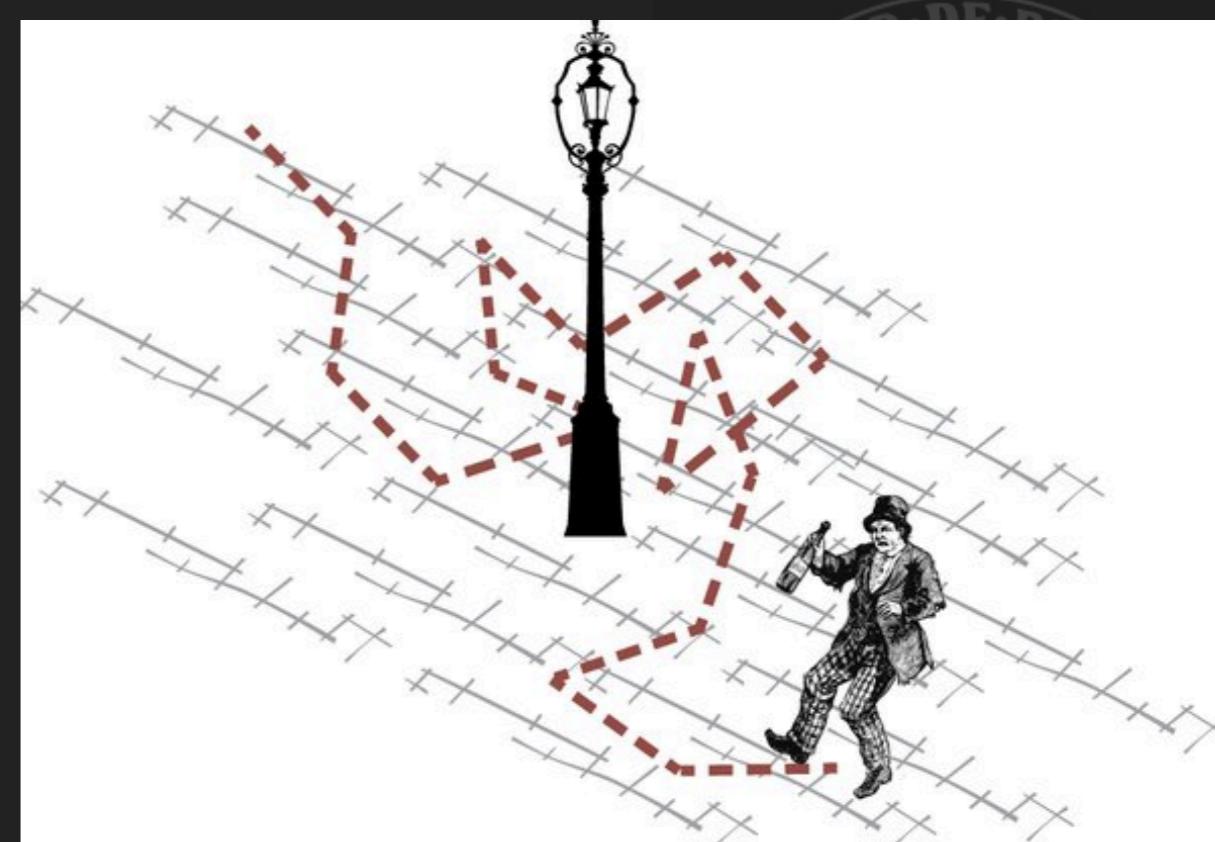
# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- ▶ Derivada primera
- ▶ Búsqueda exhaustiva (Exhaustive Search)
- ▶ Caminata aleatoria (Random walk)
- ▶ Descenso por Gradiente (Gradient Descent)



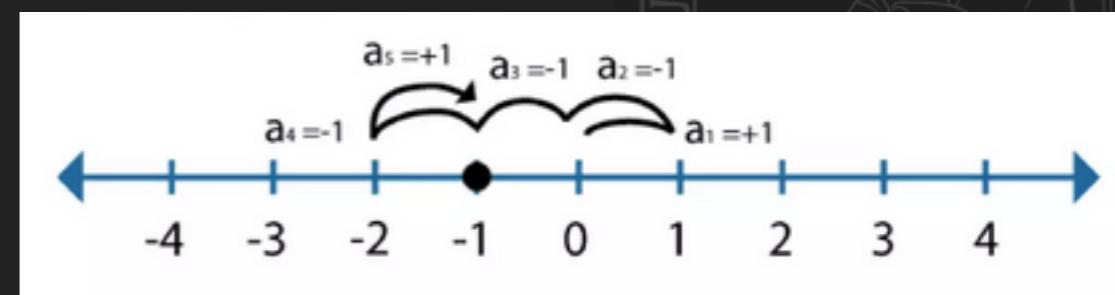
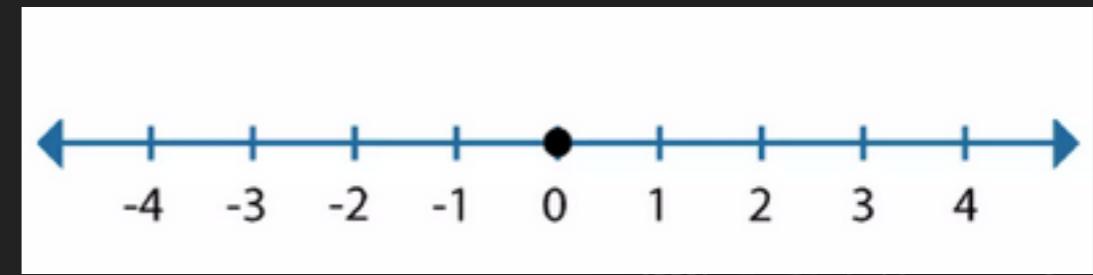
# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- ▶ **Opción 3: Caminata aleatoria**
- ▶ Es un proceso estocástico simple que implica una serie de pasos aleatorios tomados en secuencia
- ▶ Cada paso se determina de manera probabilística e independiente de los pasos anteriores.



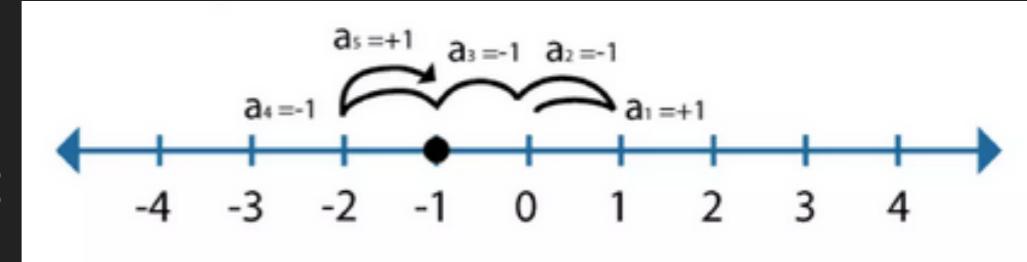
# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- ▶ El camino se construye comenzando desde una posición inicial (a menudo en el origen) y luego avanzando paso a paso según decisiones aleatorias.
- ▶ La **dirección** y la **distancia** de cada paso suelen estar determinadas por una distribución de probabilidad (uniforme o normal).



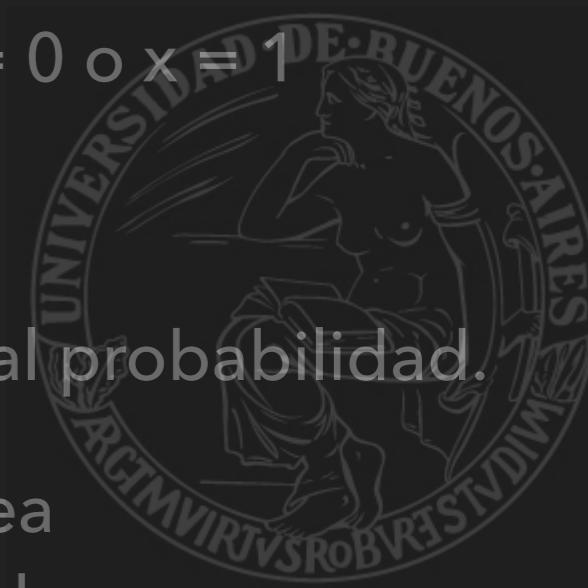
# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- ▶ Pasos para encontrar máx o mín de una función:



- ▶ **Paso 1:** Inicializar en una posición específica, generalmente  $x = 0$  o  $x = 1$
- ▶ **Paso 2:** Repetir los siguientes pasos varias veces:

- ✓ Decide la dirección (izquierda o derecha) al azar con igual probabilidad.
- ✓ Determinar el tamaño del paso (distancia recorrida) ya sea aleatoriamente o de acuerdo con una distribución definida.
- ✓ Actualice la posición actual según la dirección y el tamaño del paso.
- ✓ Evaluar si  $f(x_{actual}) > f(x_{max})$ , almacenar  $f(x_{actual})$  si el problema es de maximización, sino repetir **Paso 2** hasta completar N iteraciones.



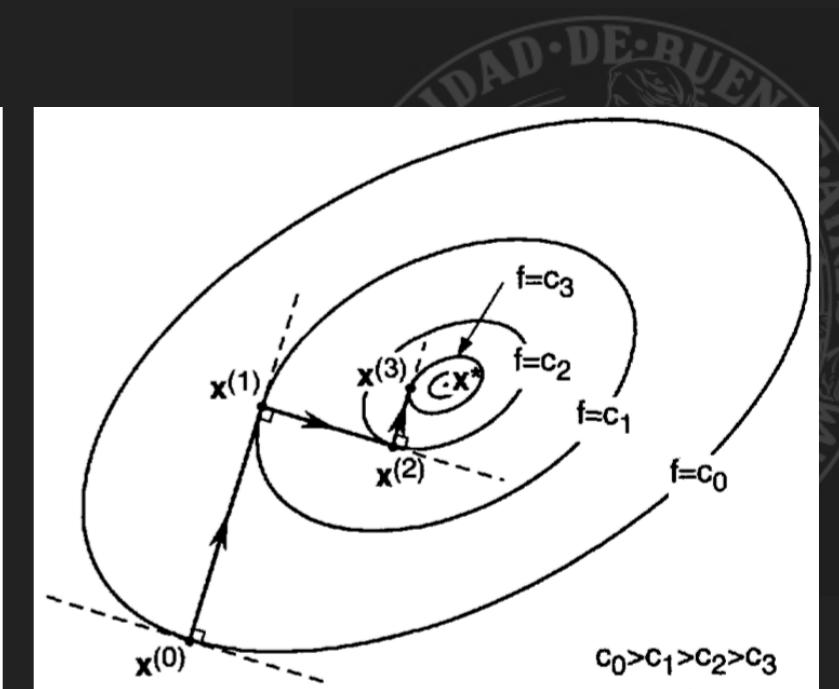
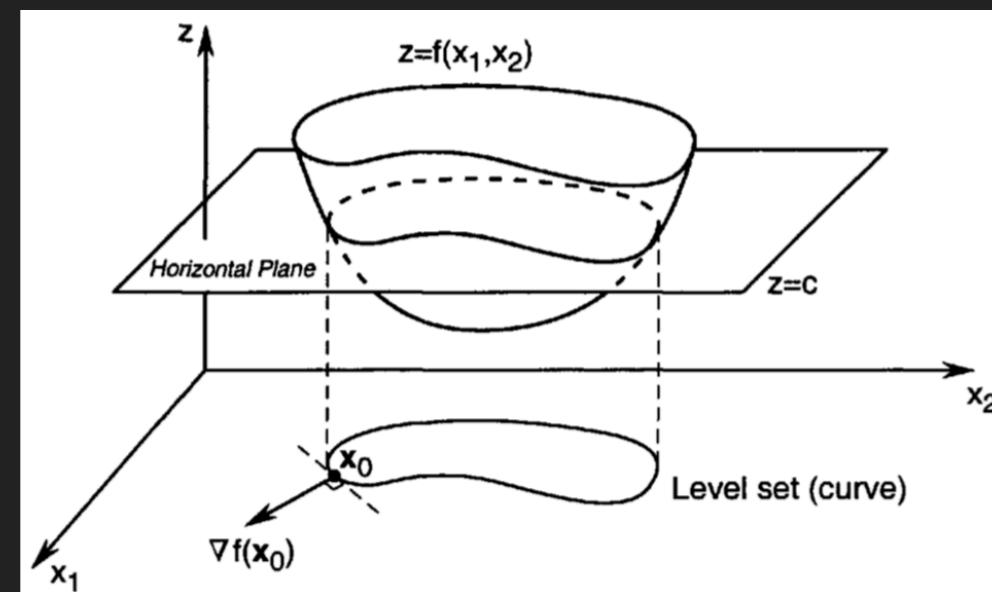
# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- ▶ Derivada primera
- ▶ Búsqueda exhaustiva (Exhaustive Search)
- ▶ Caminata aleatoria (Random walk)
- ▶ Descenso por gradiente (Gradient Descent)



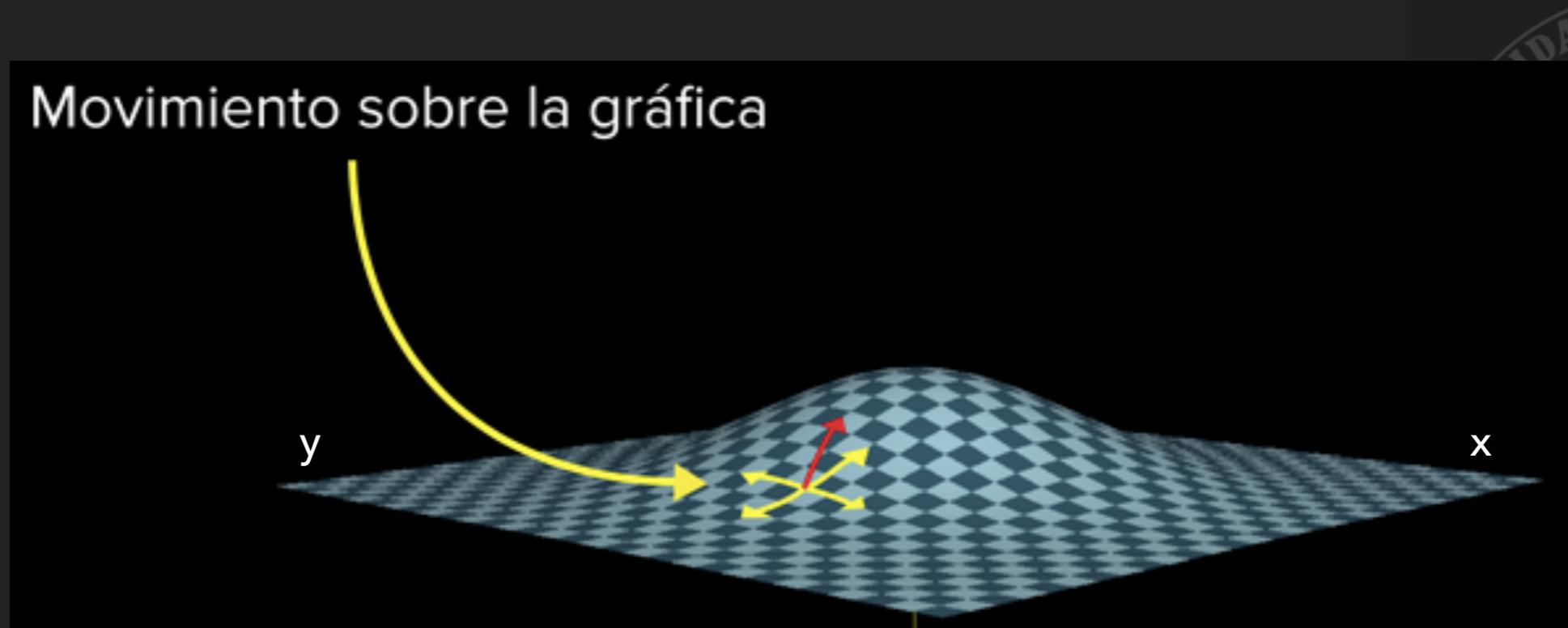
# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

## ▶ Descenso por Gradiente



# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- ▶ Descenso por Gradiente



# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- ▶ Cauchy (1847)
- ▶ ¿Qué es el gradiente?
  - ✓ "Un gradiente mide cuánto cambia la salida de una función si cambias un poco las entradas".

Lex Fridman (MIT)

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \vdots \end{bmatrix}$$



# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

Función multivariable con valores escalares

$$\nabla f(x_0, y_0, \dots) = \underbrace{\nabla f}_{\nabla f \text{ tiene el mismo tipo de entradas que } f} (x_0, y_0, \dots)$$

Notación para el vector gradiente

$$\begin{bmatrix} \frac{\partial f}{\partial x}(x_0, y_0, \dots) \\ \frac{\partial f}{\partial y}(x_0, y_0, \dots) \\ \vdots \end{bmatrix}$$

$\nabla f$  es un vector con todas las derivadas parciales posibles de  $f$

# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- ▶ Método de descenso por gradiente:

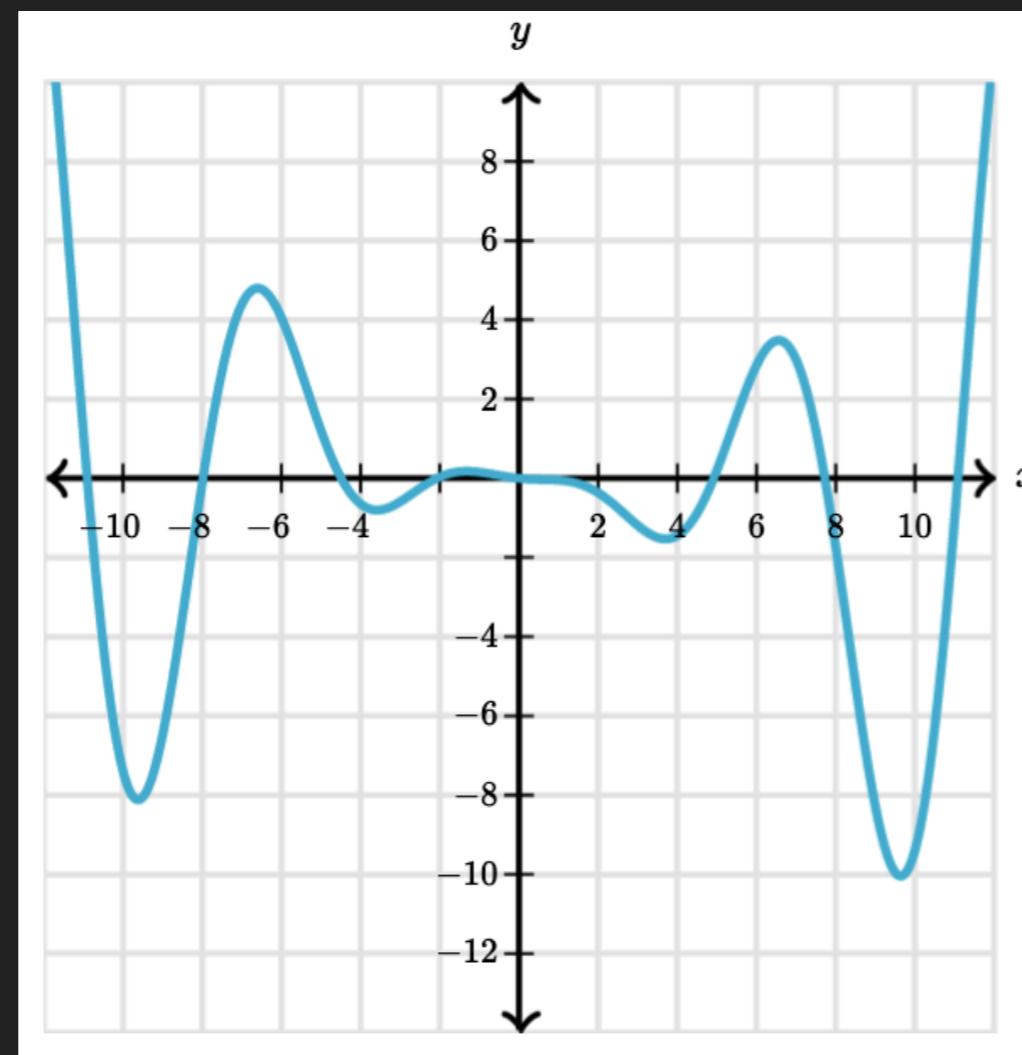
$$x_{n+1} = x_n - \alpha \nabla f(x_n)$$



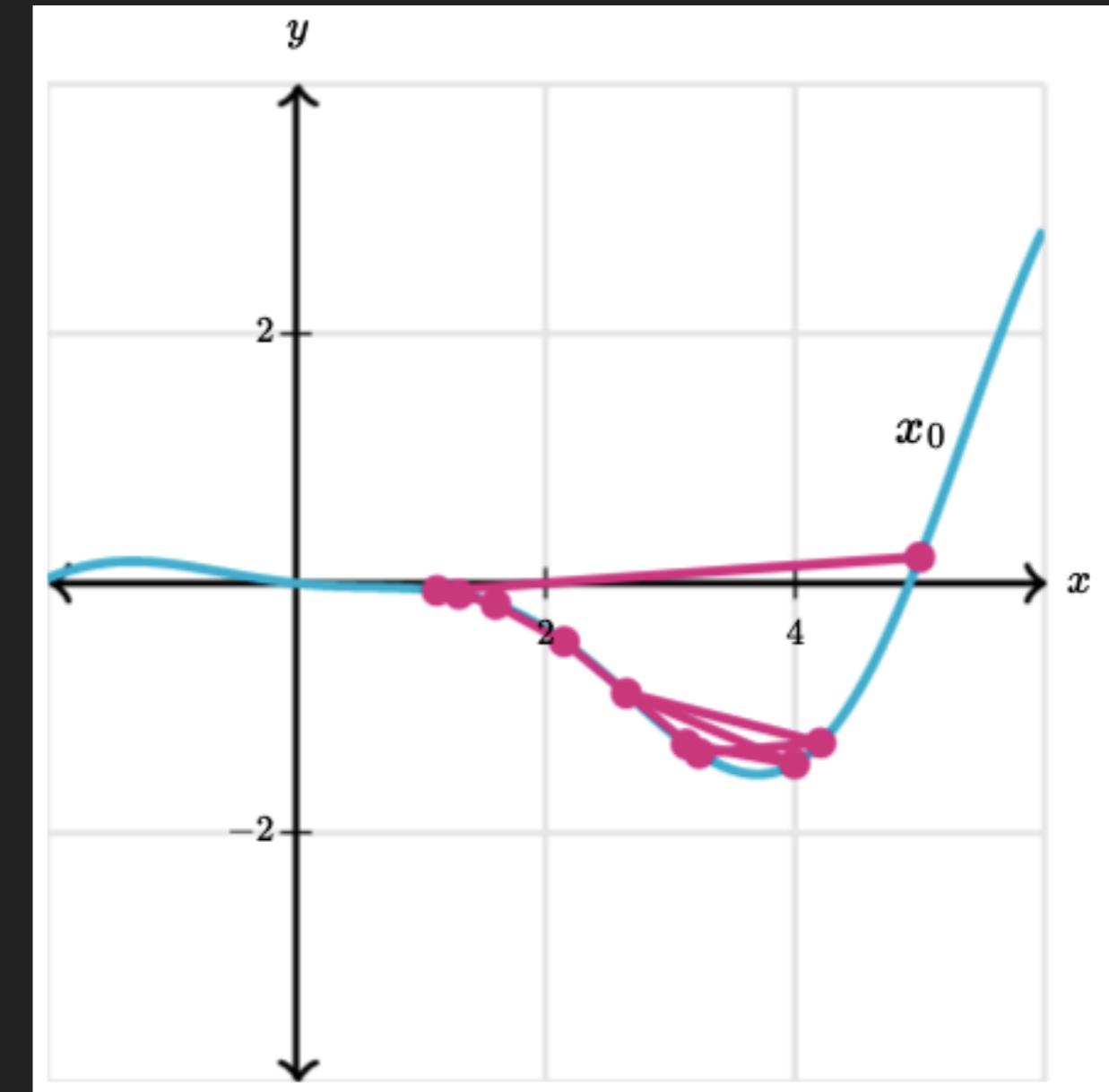
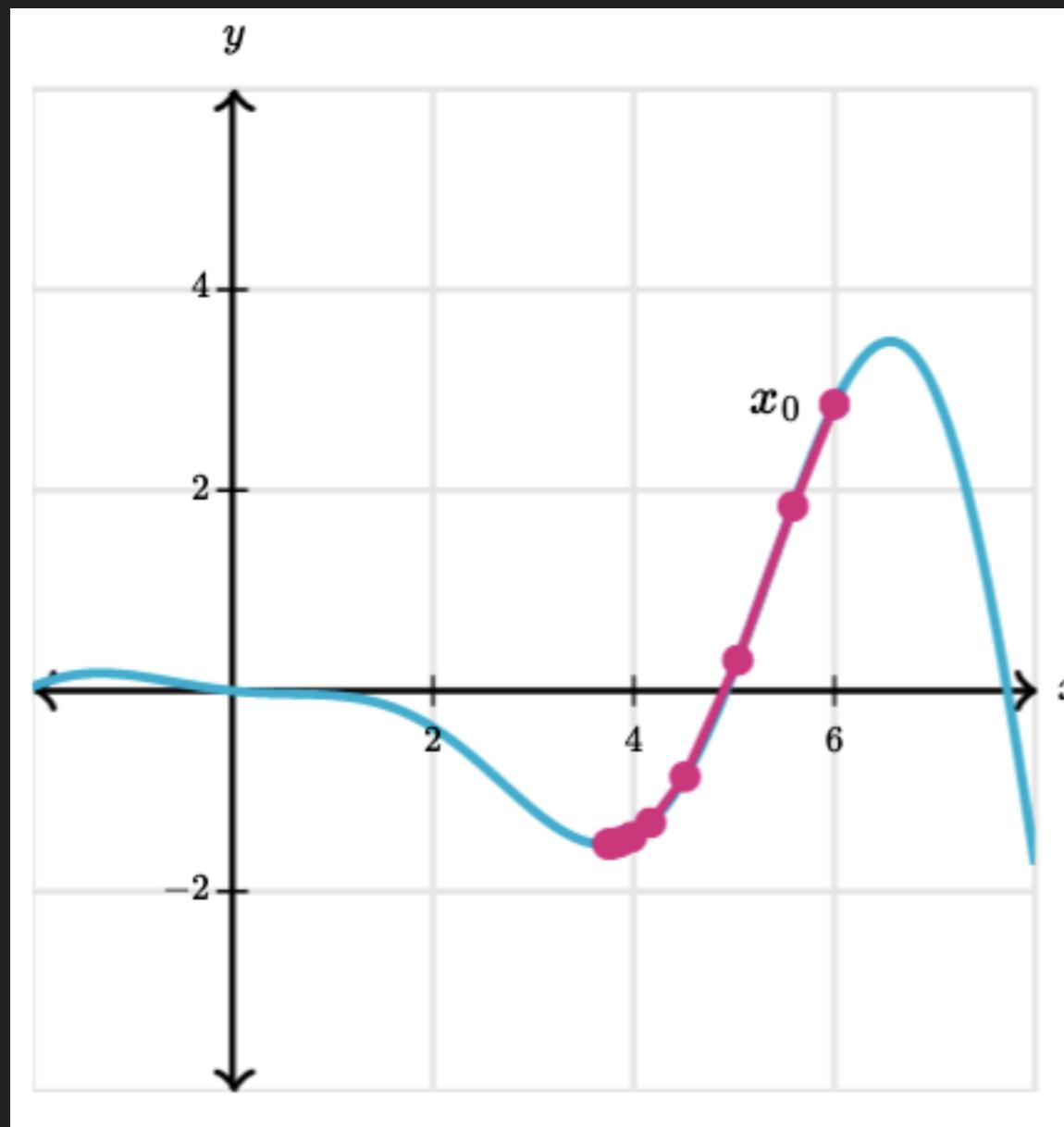
# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

- ▶ Ejemplo:

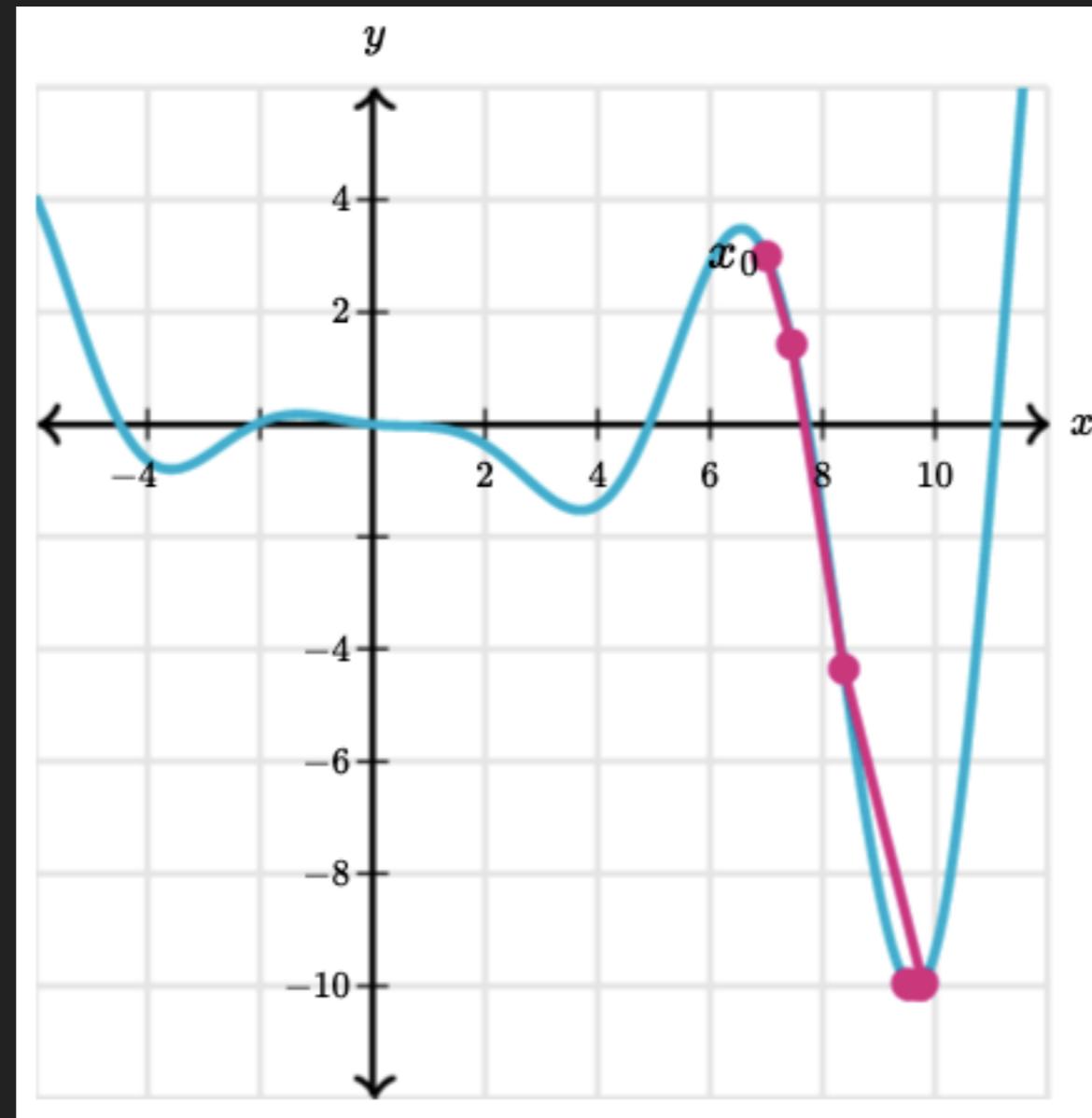
$$f(x) = \frac{x^2 \cos(x) - x}{10}$$



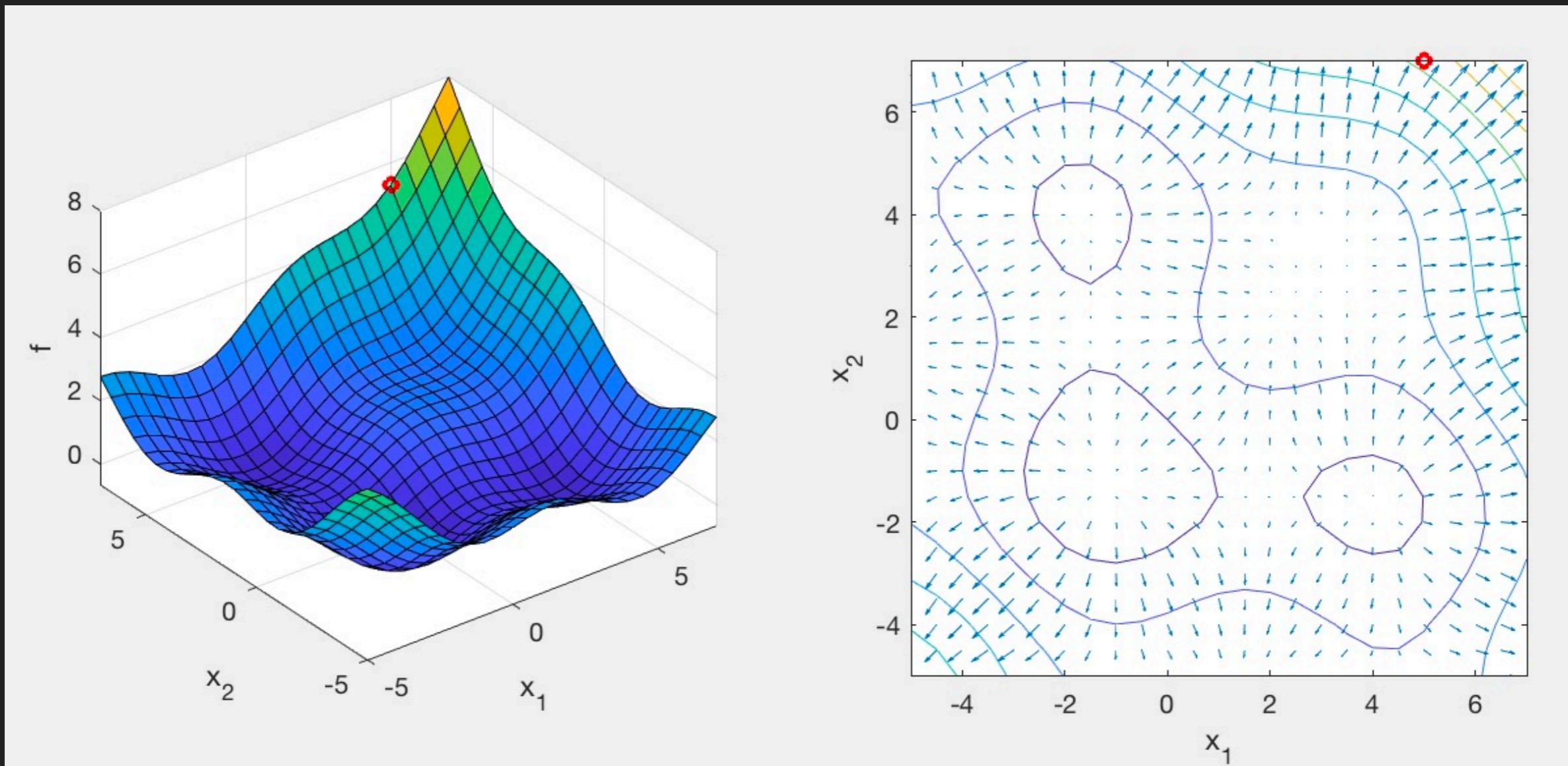
# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?



# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?

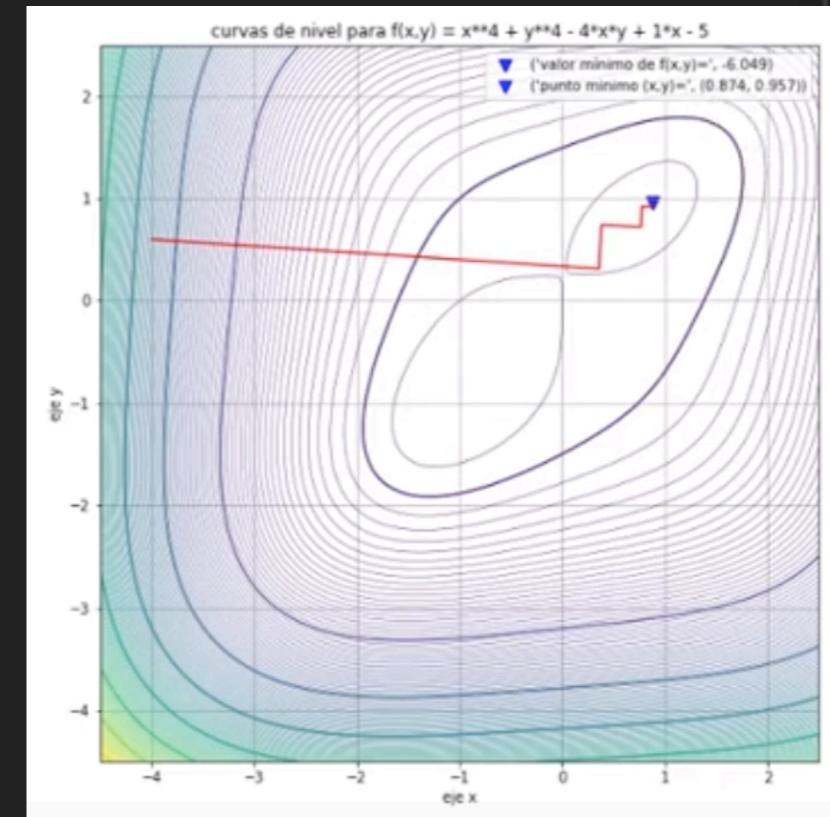
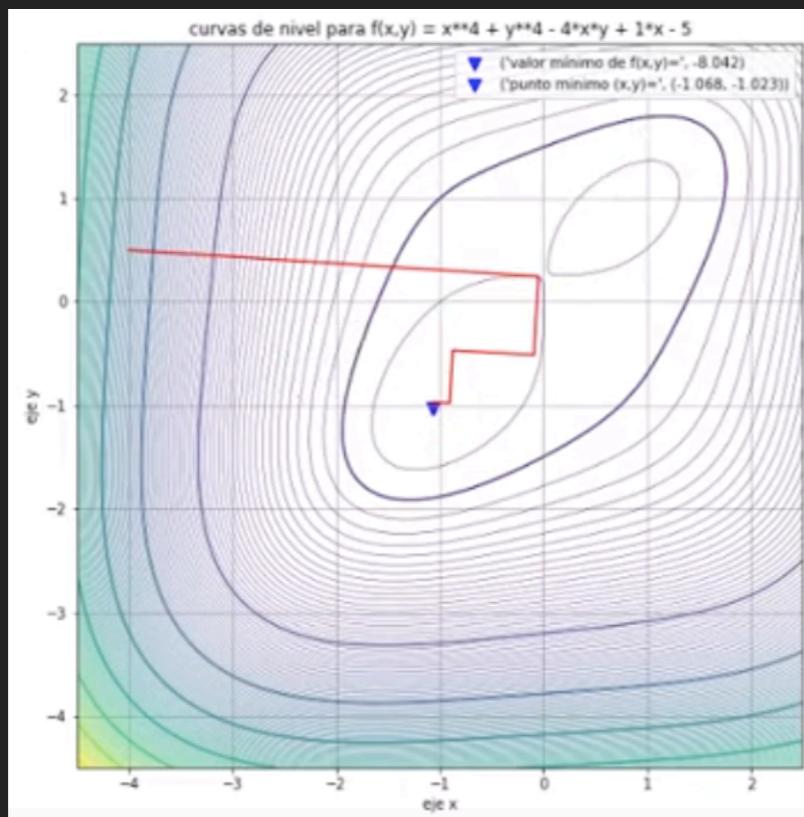


# ¿CÓMO ENCONTRAR EL ÓPTIMO (MÁX O MÍN) DE UNA FO?



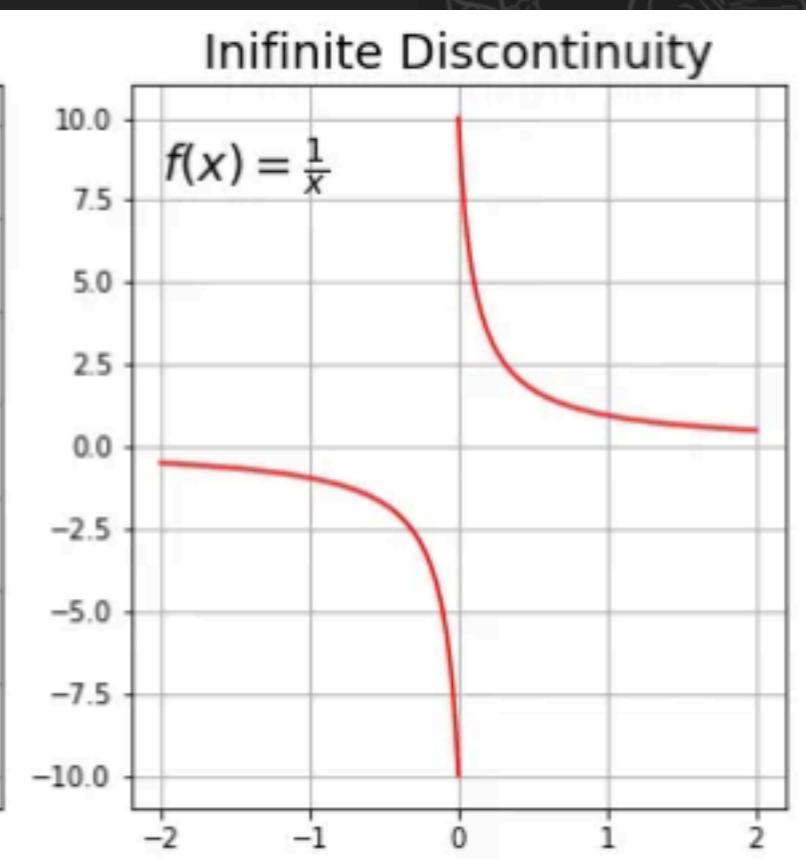
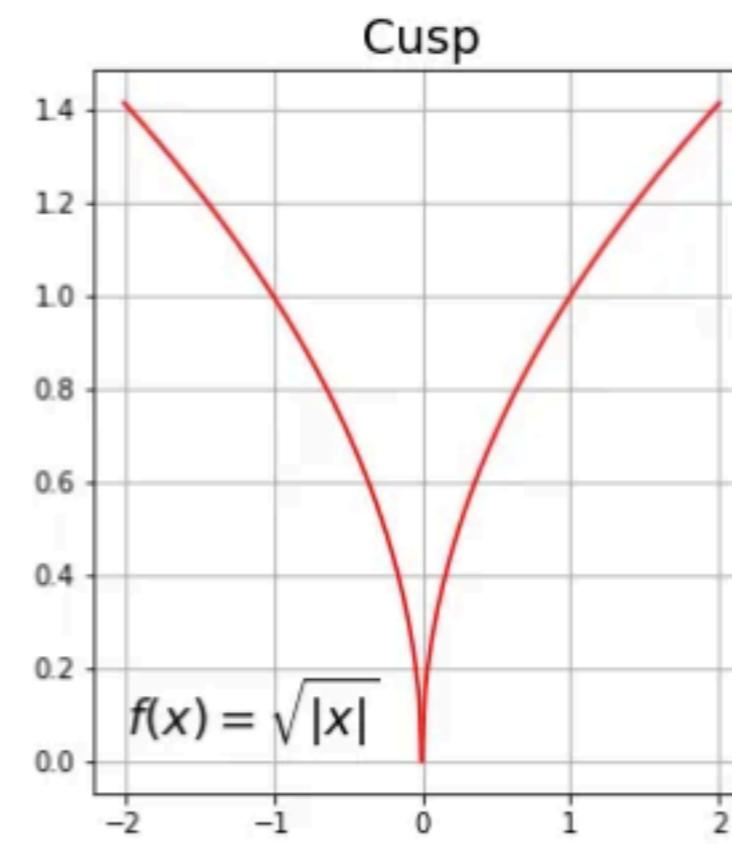
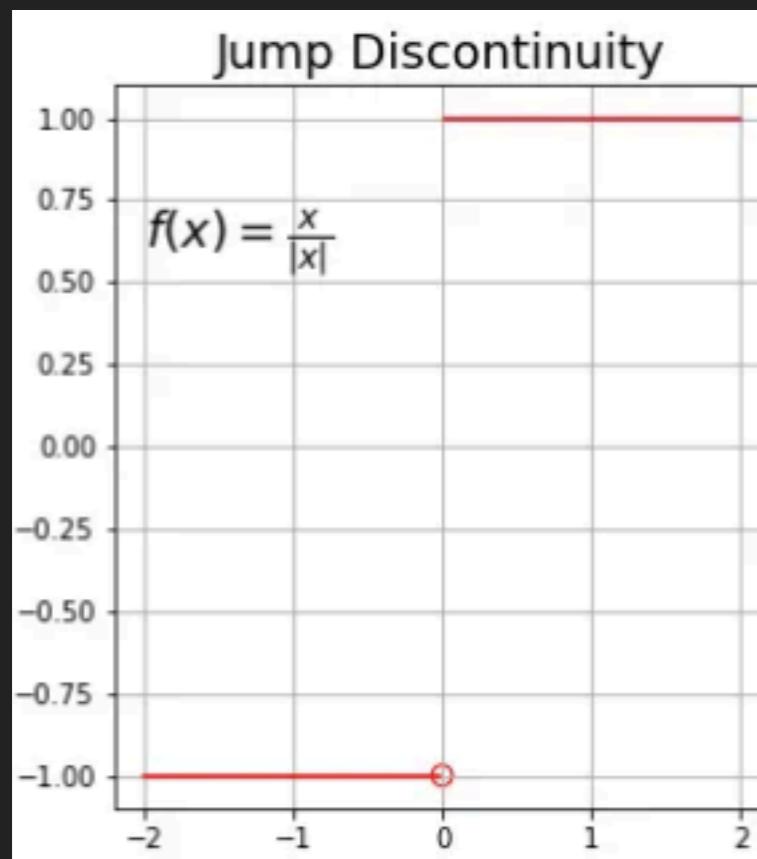
## DESVENTAJAS DE LOS MÉTODOS TRADICIONALES (I)

- ▶ 1. La **solución final** de un problema de optimización resuelto mediante un método tradicional depende de la **solución inicial** elegida al azar. (Pratihar, 2007).
- ▶ Si la solución inicial posee un gradiente en dirección a un mínimo local, la solución final se quedará **estancada en el óptimo local**. (Pratihar, 2007).



## DESVANTAJAS DE LOS MÉTODOS TRADICIONALES (II)

- 2. Para una función objetivo **discontinua**, el gradiente **no se puede determinar en el punto de discontinuidad**. Por lo tanto, los métodos basados en gradientes no se pueden utilizar para dicha función.



## DESVANTAJAS DE LOS MÉTODOS TRADICIONALES (II)

- ▶ 3. Las **variables discretas** (enteras) son difíciles de manejar utilizando los métodos tradicionales de optimización.
- ▶ 4. No son adecuados para **computación paralela**.



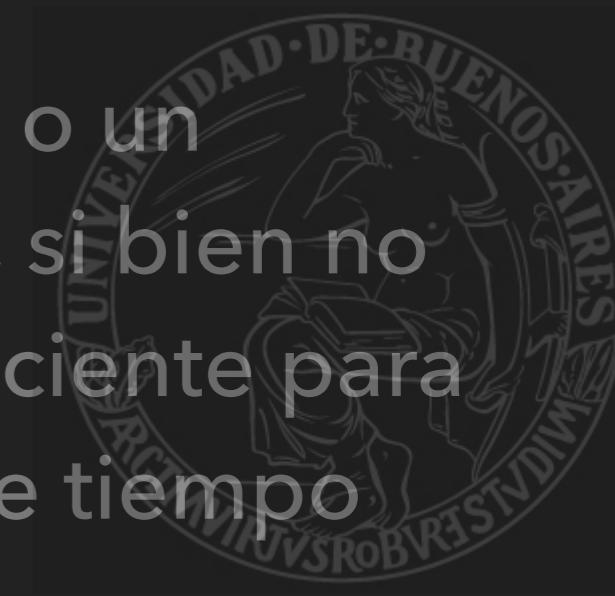
# HEURÍSTICAS (MÉTODOS TRADICIONALES)

- ▶ ¿Qué son las **heurísticas**?



# HEURÍSTICAS (MÉTODOS TRADICIONALES)

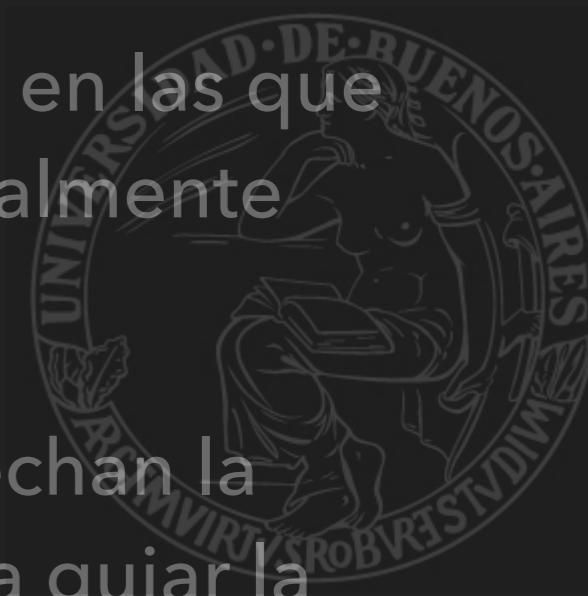
- ▶ Heurística -> *Heuriskein* = "encontrar", "descubrir"
- ▶ **Heurística:** Una heurística es una regla general o un método práctico para resolver problemas que, si bien no se garantiza que sea óptimo o perfecto, es suficiente para alcanzar una solución aceptable en un plazo de tiempo razonable.



# HEURÍSTICAS

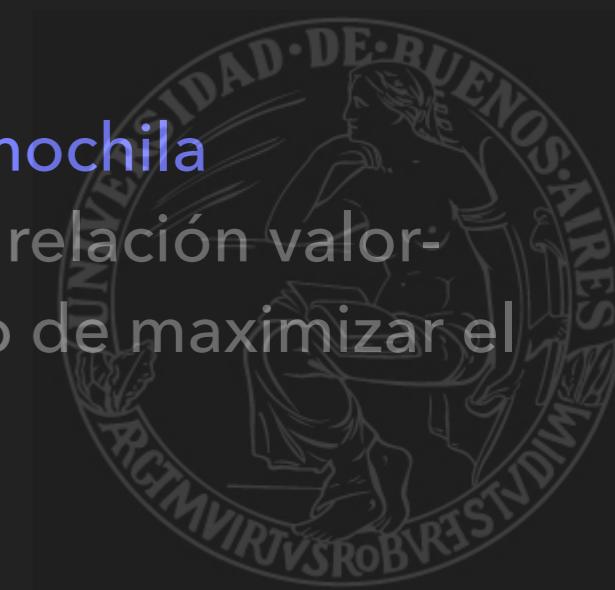
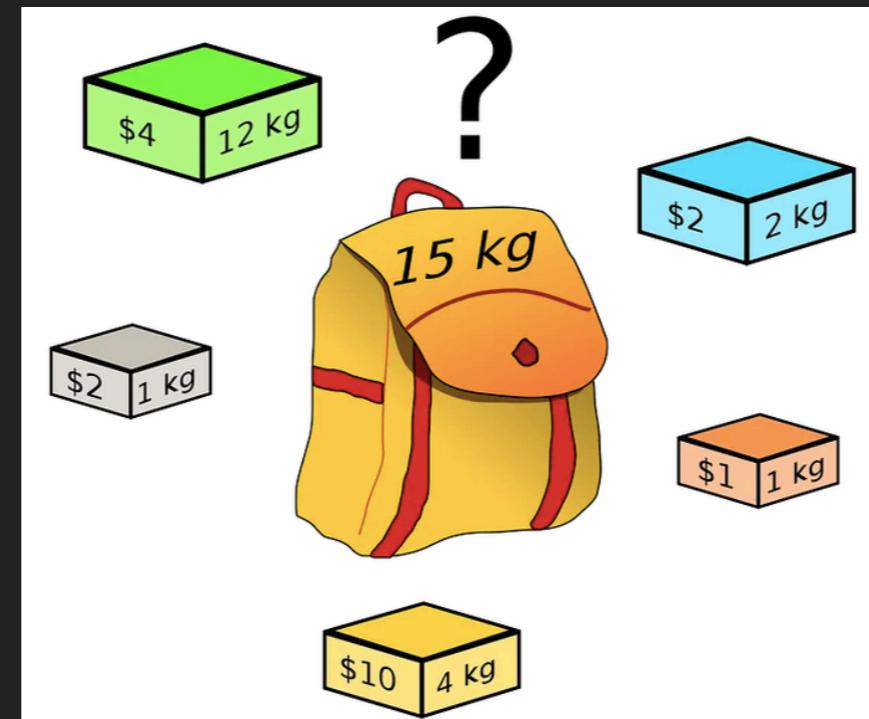
## ► Características:

- ✓ La heurística se utiliza a menudo en situaciones en las que encontrar una solución óptima es computacionalmente costoso o poco práctico.
- ✓ Por lo general, son simples e intuitivos y aprovechan la experiencia o el conocimiento del dominio para guiar la toma de decisiones.
- ✓ La heurística no garantiza la mejor solución, pero tiene como objetivo encontrar rápidamente una solución satisfactoria.



## HEURÍSTICAS CLÁSICAS (OTROS EJEMPLOS)

- ▶ **Heurística codiciosa (greedy):** toma decisiones localmente óptimas en cada paso con la esperanza de encontrar un óptimo global.
  - ✓ Ejemplo : el algoritmo codicioso para el **problema de la mochila (knapsack problem)** selecciona artículos en función de su relación valor-peso sin considerar consecuencias futuras, con el objetivo de maximizar el valor sin exceder la capacidad de peso.



## HEURÍSTICAS CLÁSICAS (OTROS EJEMPLOS)

- ▶ **Heurística del vecino más cercano** : esta heurística selecciona la opción disponible más cercana en cada paso de una secuencia, a menudo utilizada en problemas de enrutamiento.
- ✓ Ejemplo : en el problema del viajante de comercio (TSP), la heurística del vecino más cercano construye un recorrido comenzando en una ciudad aleatoria y visitando repetidamente la ciudad más cercana no visitada hasta que se incluyan todas las ciudades.



## HEURÍSTICAS CLÁSICAS (OTROS EJEMPLOS)

- ▶ **Heurística basada en reglas** : la heurística basada en reglas utiliza un conjunto de reglas o condiciones predefinidas para guiar la toma de decisiones.
- ✓ Ejemplo : en ajedrez, la heurística para la evaluación de piezas podría incluir reglas como asignar valores más altos a las piezas en función de su posición en el tablero, amenazas potenciales o importancia estratégica.



# METAHEURÍSTICAS (ALGORITMOS EVOLUTIVOS)

- ▶ ¿Qué son las metaheurísticas?



# METAHEURÍSTICAS (ALGORITMOS EVOLUTIVOS)

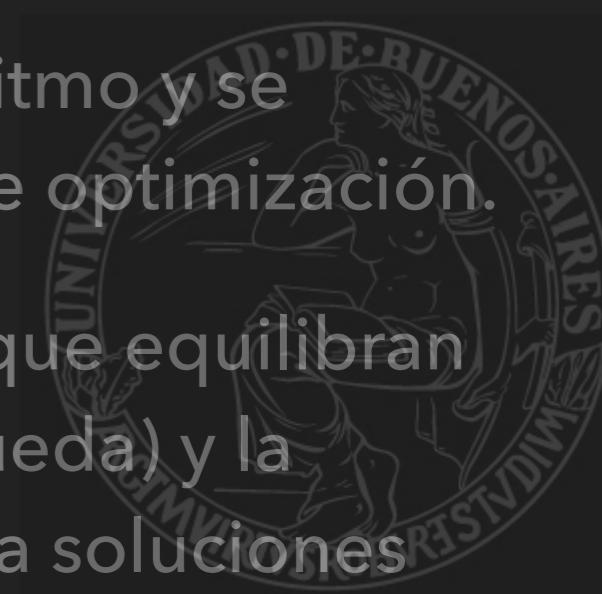
- ▶ Metaheurística -> *Meta* = “mas allá de”, “por encima de”
- ▶ Una **metaheurística** (Glover, 1986) es una estrategia o marco de nivel superior que guía la búsqueda de soluciones a problemas de optimización.
- ▶ A diferencia de los algoritmos clásicos, las metaheurísticas proporcionan un enfoque general aplicable a una amplia gama de dominios de problemas.



# METAHEURÍSTICAS

## ► Características :

- ▶ Las metaheurísticas son independientes del algoritmo y se pueden aplicar a diferentes tipos de problemas de optimización.
- ▶ A menudo implican técnicas de mejora iterativas que equilibran la exploración (diversificando el espacio de búsqueda) y la explotación (intensificando la búsqueda en torno a soluciones prometedoras).
- ▶ Ejemplos de metaheurísticas incluyen **algoritmos genéticos**, **optimización por colonias de hormigas**, **optimización por enjambres de partículas**, entre otras.



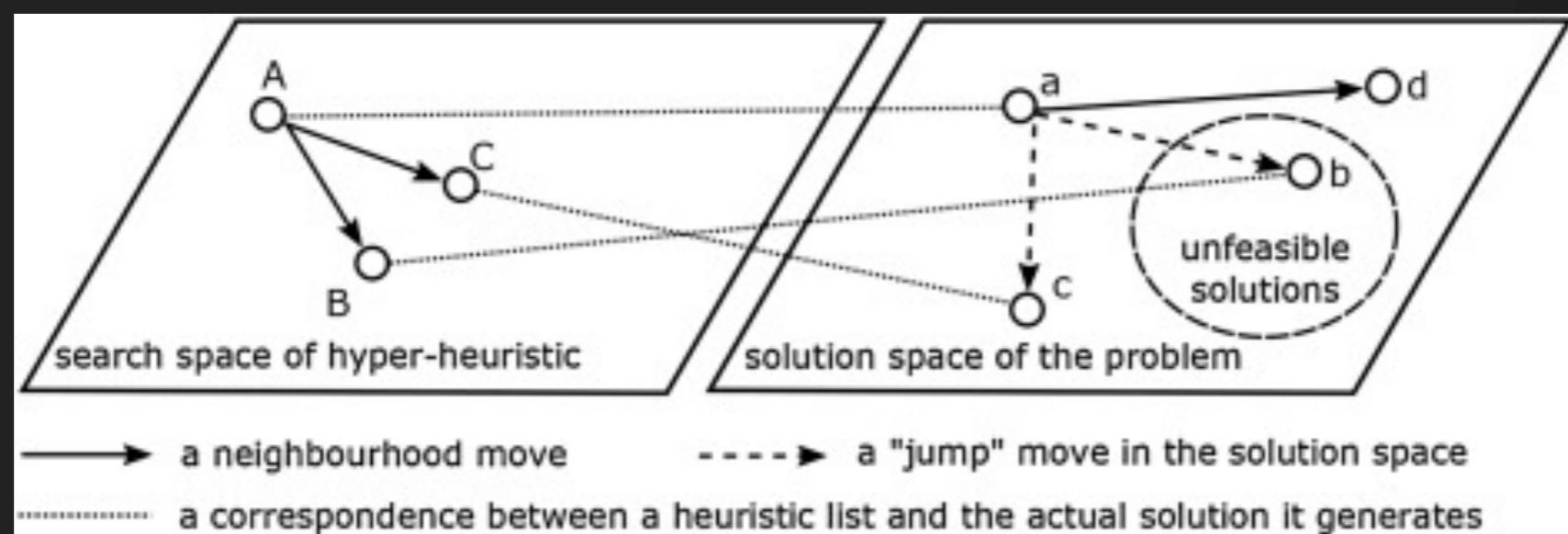
# METAHEURÍSTICAS (ALGORITMOS EVOLUTIVOS)

- ▶ ¿Qué son las hiperheurísticas?



# HIPERHEURÍSTICAS

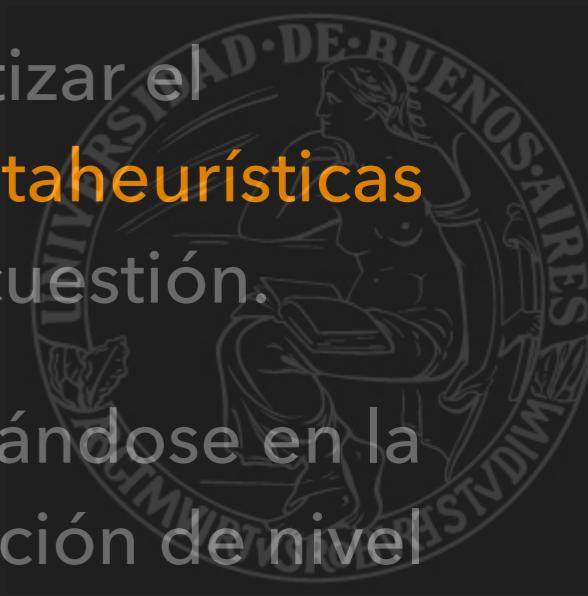
- ▶ Una **hiperheurística** es una metodología o enfoque que selecciona, combina, adapta o genera **heurísticas** o **metaheurísticas** para resolver problemas de optimización.
- ▶ Mejora la eficiencia y eficacia de la resolución de problemas.



# HIPERHEURÍSTICAS

## ► Características:

- ✓ Las hiperheurísticas tienen como objetivo automatizar el proceso de selección o diseño de **heurísticas** o **metaheurísticas** en función de las **características del problema** en cuestión.
- ✓ Proporcionan un mayor nivel de abstracción, centrándose en la **selección** o **generación** de estrategias de optimización de nivel inferior.
- ✓ Se utilizan en problemas de **optimización** complejos donde diferentes heurísticas o metaheurísticas pueden funcionar mejor dependiendo de restricciones o casos de problemas específicos.



## TÉCNICAS INTELIGENTES DE OPTIMIZACIÓN (METAHEURÍSTICAS)

- ▶ Algoritmos Genéticos (GA)

Holland, 1975

- ▶ Optimización por Colonia de Hormigas (ACO)

Dorigo, 1992

- ▶ Optimización por Enjambre de Partículas (PSO)

Kennedy y Eberhart, 1995

- ▶ Colonia de Abejas Artificiales (ABC)

Karaboga, 2005

- ▶ Optimización Basada en Enseñanza Aprendizaje (TLBO)

Rao y colegas, 2011



## REFERENCIAS BIBLIOGRÁFICAS Y WEB

- ▶ Roy, S., & Chakraborty, U. (2013). *Introduction to soft computing: neuro-fuzzy and genetic algorithms*. Pearson.
- ▶ Eiben, A. E., & Smith, J. E. (2015). *Introduction to evolutionary computing*. Springer-Verlag Berlin Heidelberg.
- ▶ Hillier, F. S. Lieberman, G. J. (2010). *Introducción a la Investigación de Operaciones*.
- ▶ Lu Y, Hao J, Wu Q. (2022). Solving the clustered traveling salesman problem via traveling salesman problem methods. *PeerJ Computer Science* 8:e972
- ▶ Pratihar, D. K. (2007). *Soft computing*. Alpha Science International, Ltd.
- ▶ A. Cauchy. (1847). Méthode générale pour la résolution des systèmes d'équations simultanées, *Comp. Rend. Sci. Paris*, 25. 536-538.
- ▶ Chong, E. K., & Żak, S. H. (2013). *An introduction to optimization* (Vol. 75). John Wiley & Sons.
- ▶ Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5), 533-549.
- ▶ <https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21>
- ▶ <https://www.youtube.com/watch?v=EmyoaQGAkyw>

