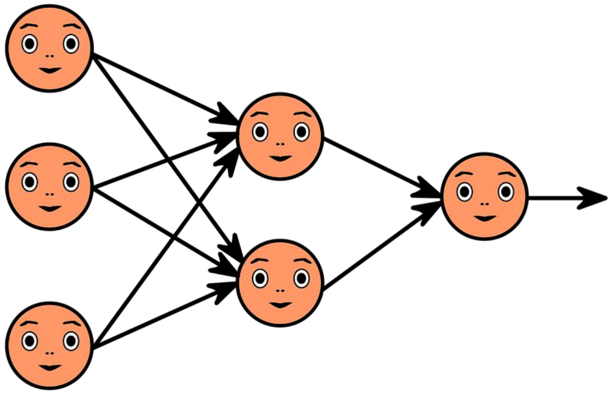
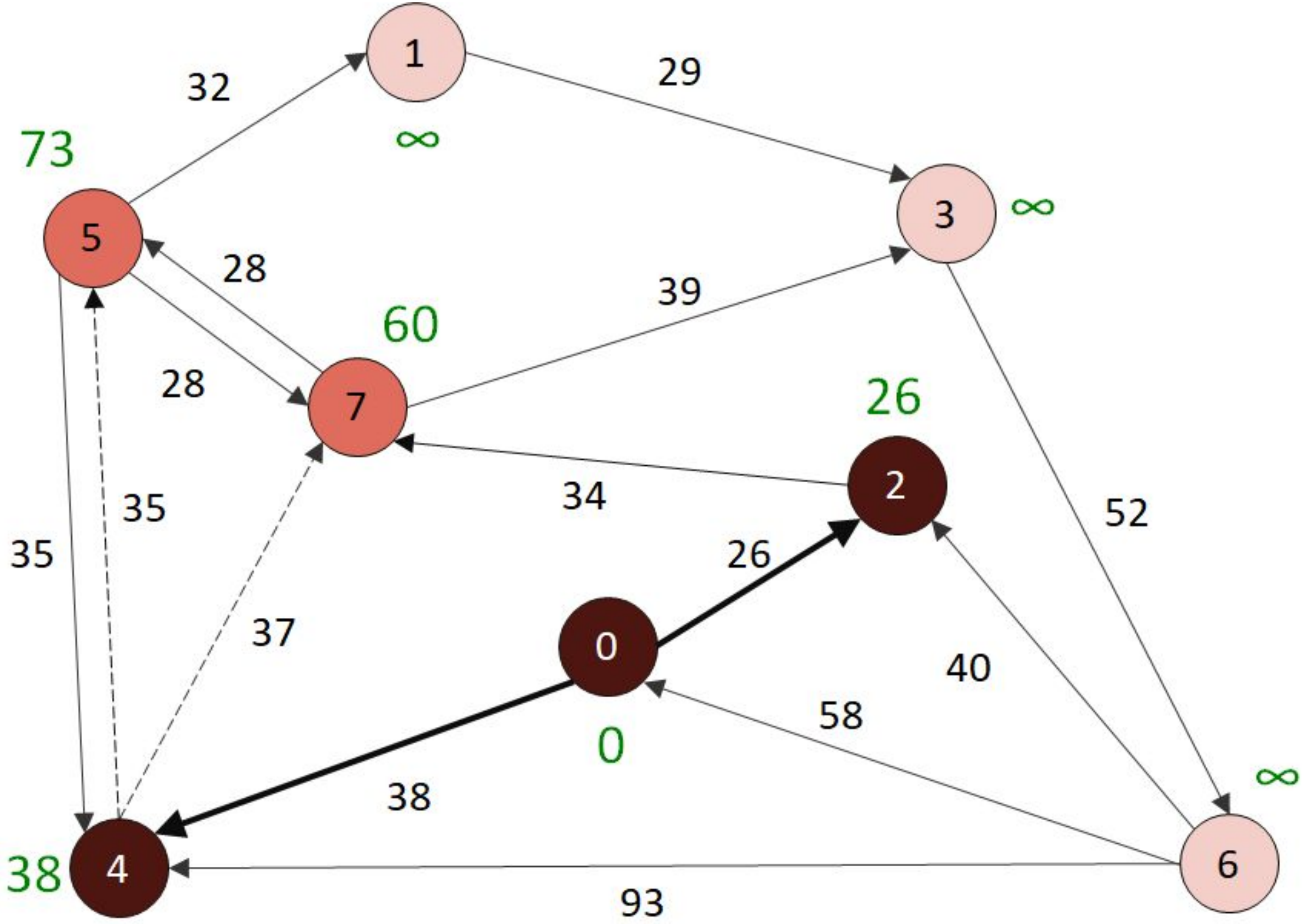
ALGORITMOS CODICIOSOS Y BÚSQUEDA DE SOLUCIONES SUBÓPTIMAS: 非常に長い愛憎関係



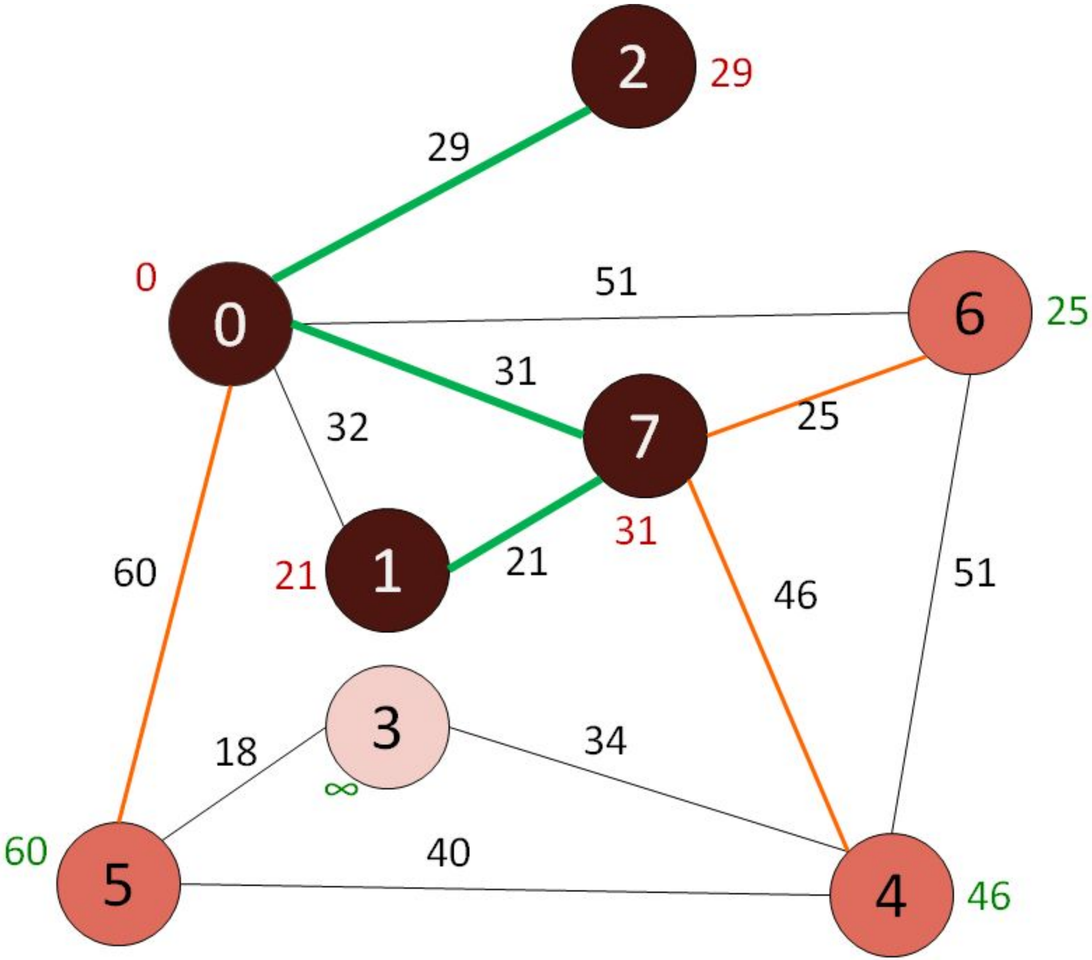
Un algoritmo codicioso intenta encontrar el óptimo del problema tomando la opción más prometedora en cada paso

Un algoritmo codicioso jamás se arrepiente de su decisión

Dijkstra



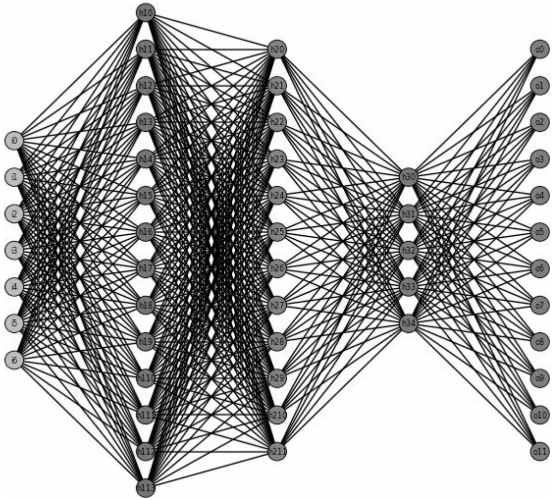
Prim

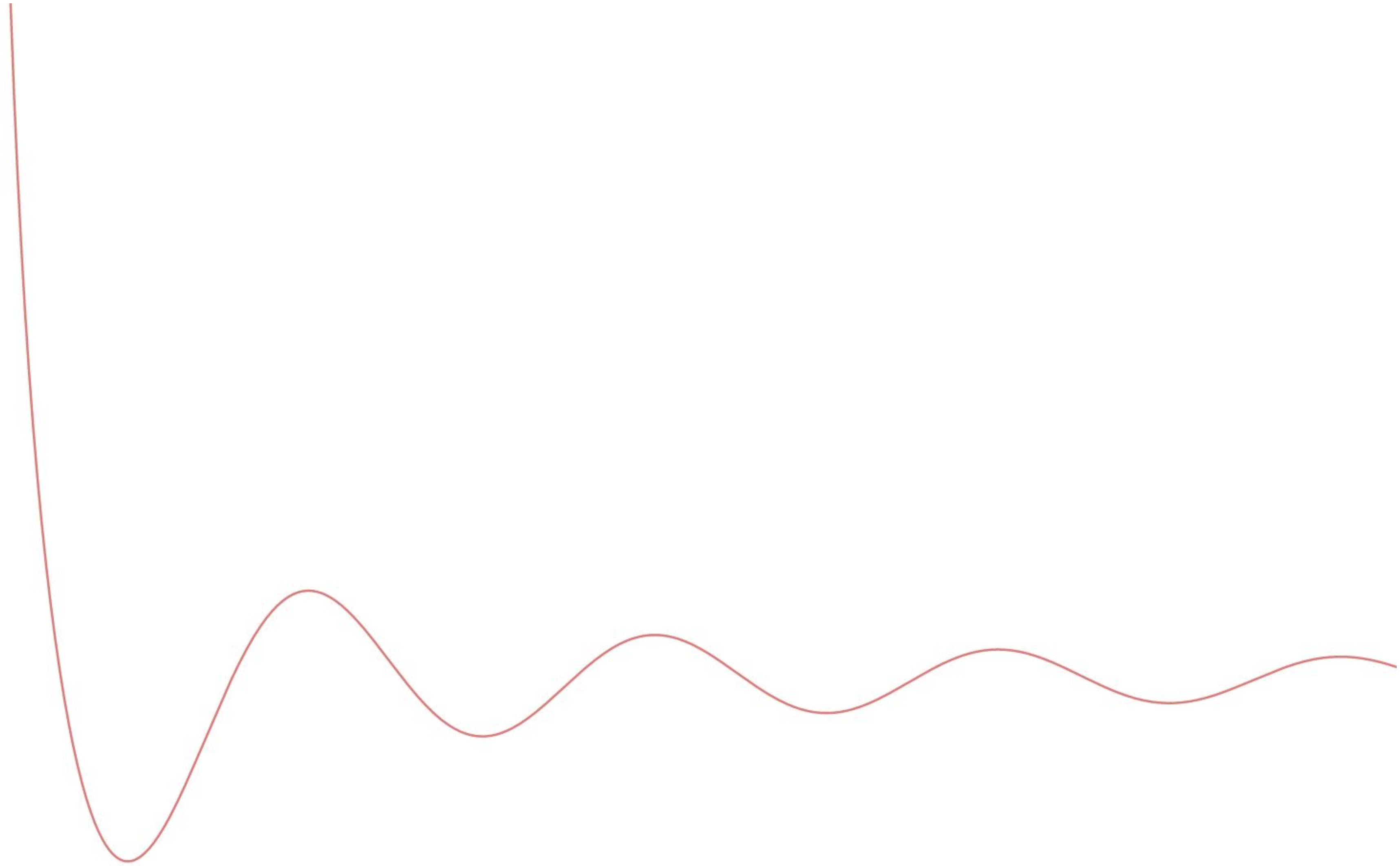


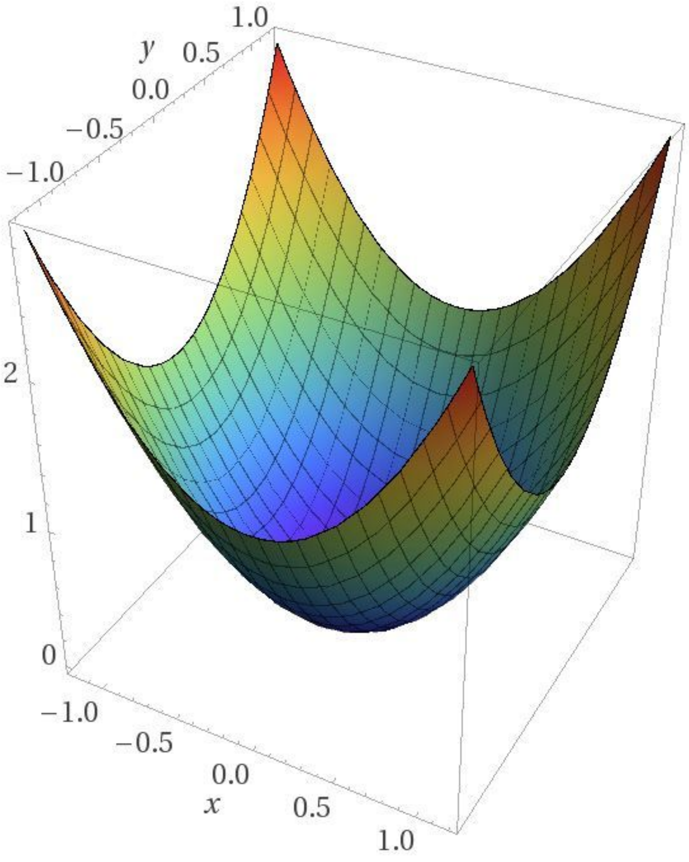
Problema

Encontrar el mínimo en una función no analítica o no diferenciable, pero que varía de manera relativamente continua

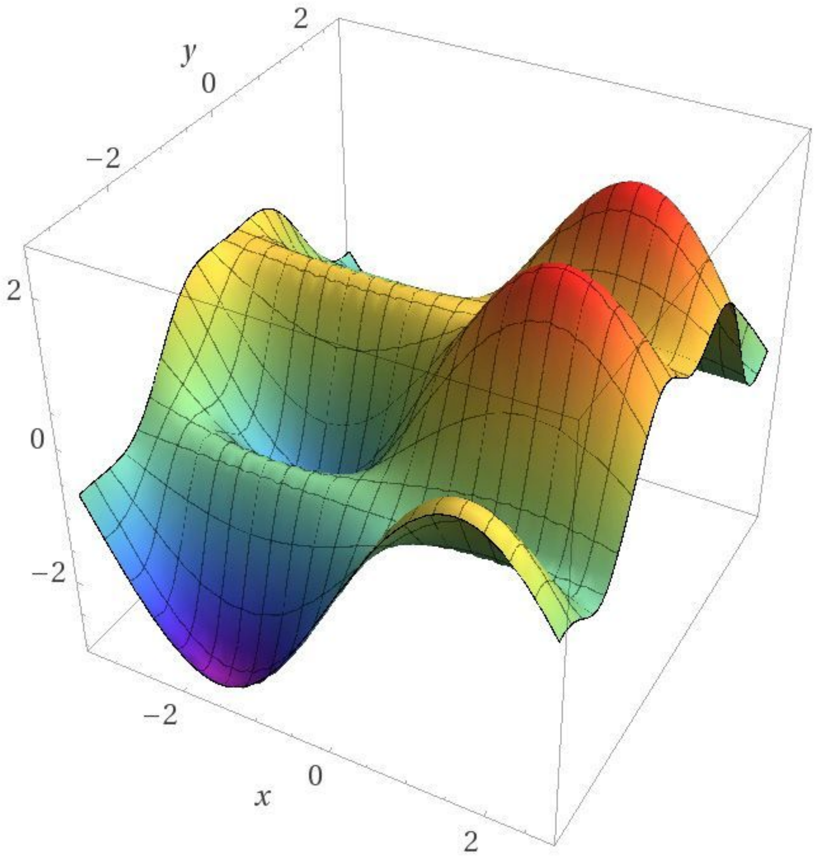
Usos: Fit de modelos de machine learning (particularmente útil en redes neuronales)







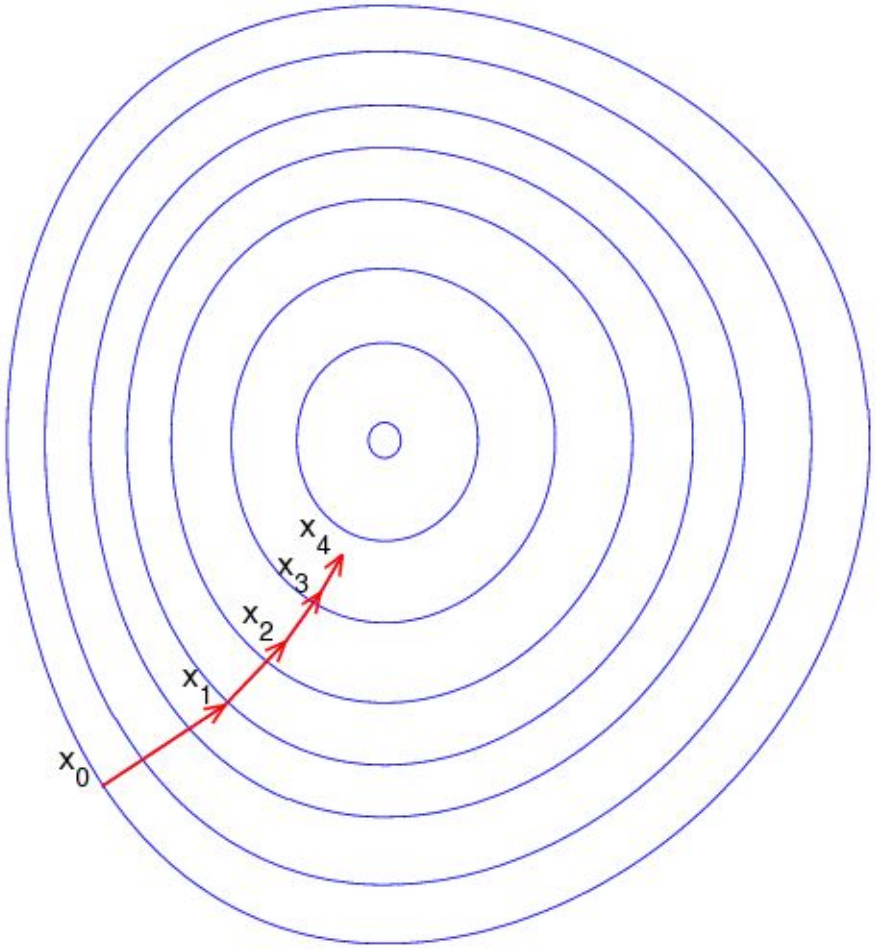
Computed by Wolfram|Alpha



Computed by Wolfram|Alpha

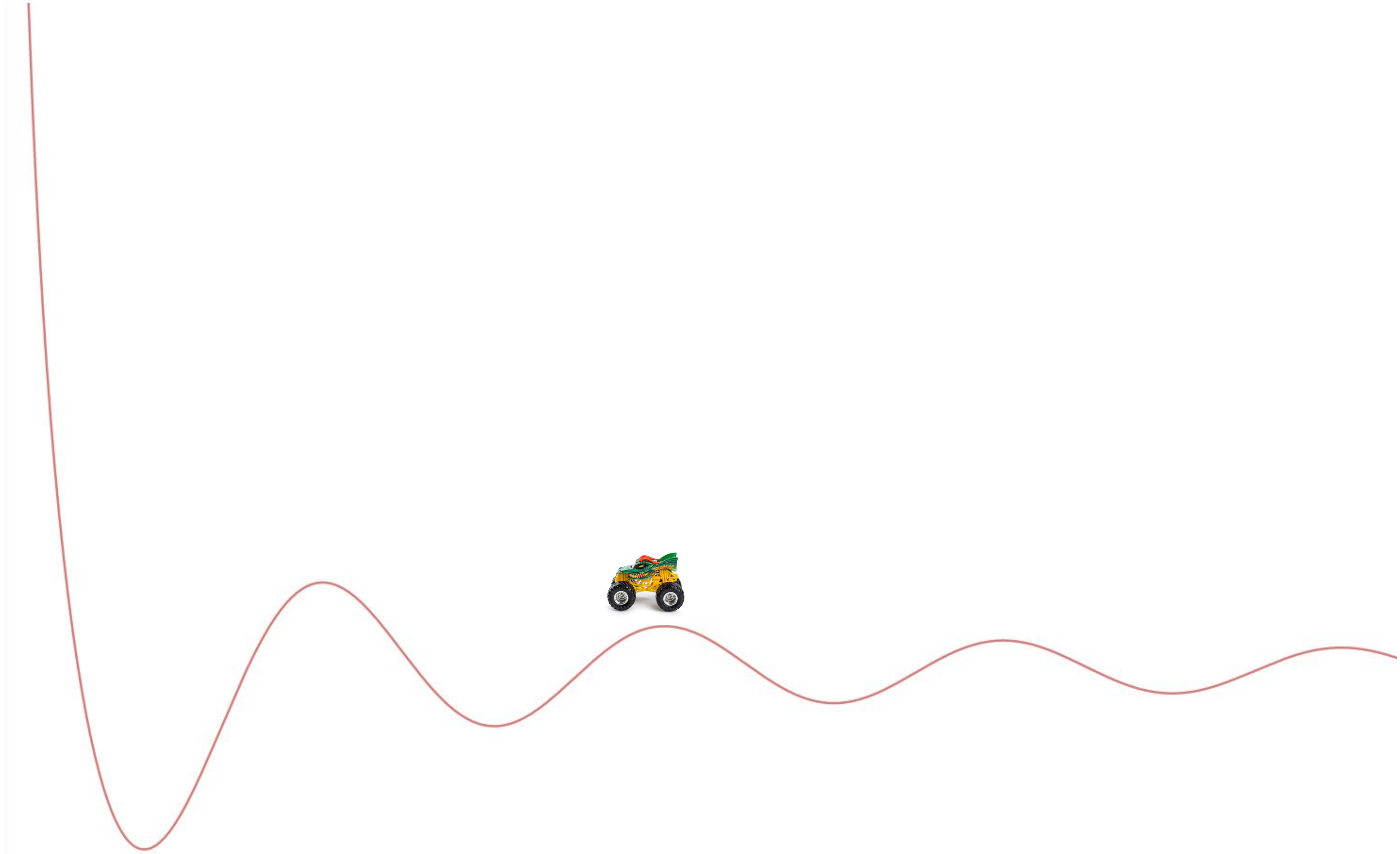
Suposición:

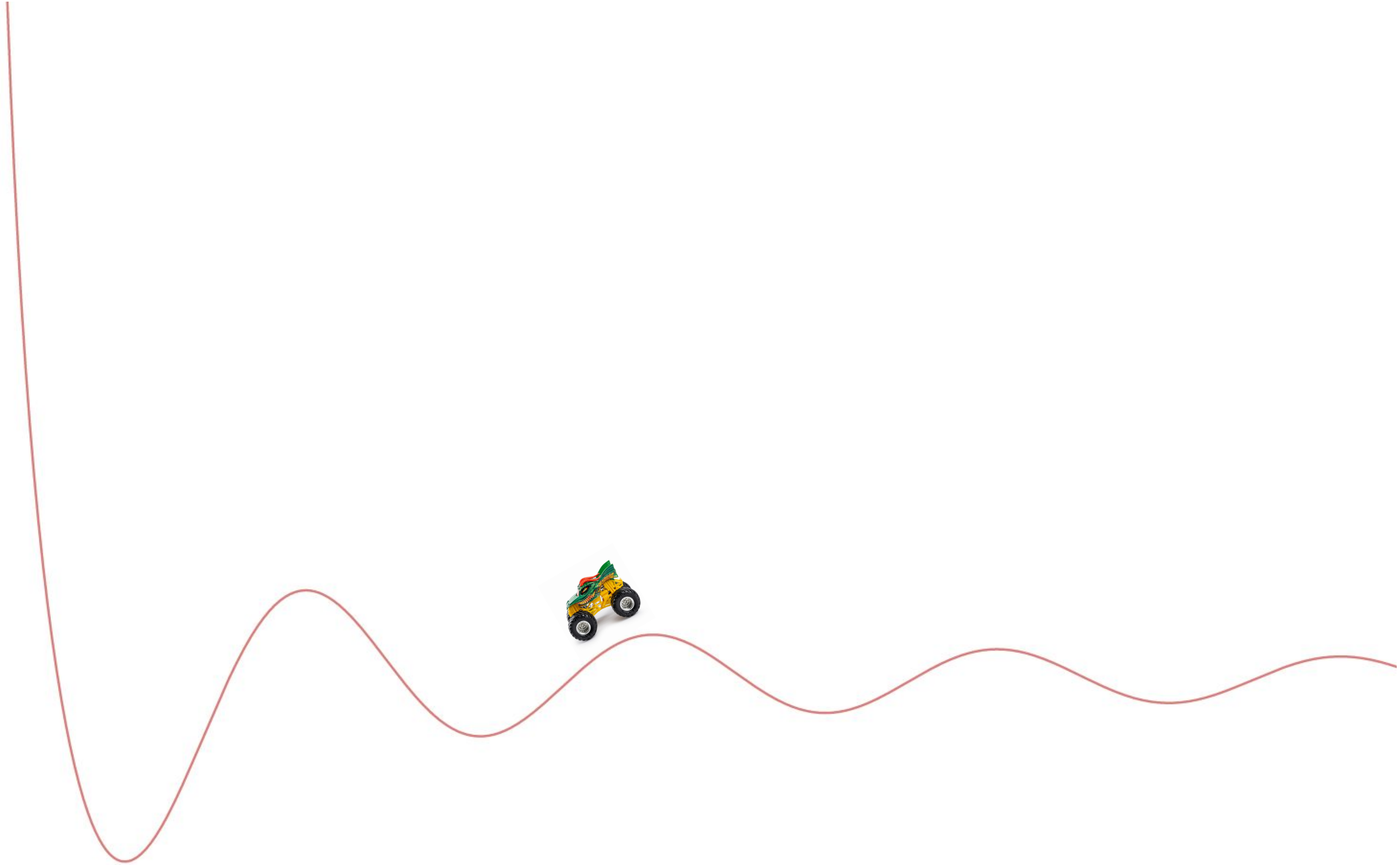
Movernos en la dirección del gradiente más negativo nos llevará al mínimo global

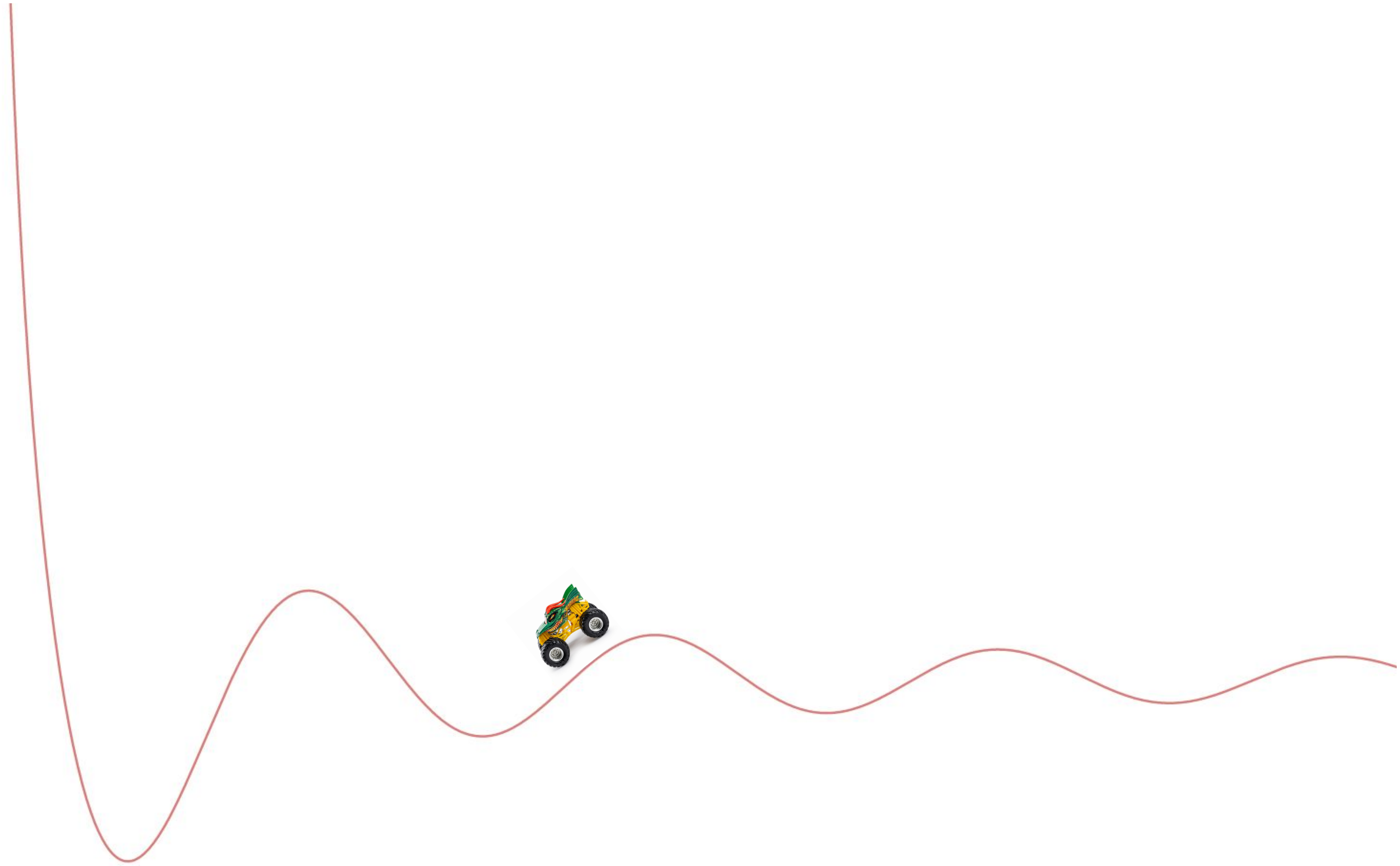


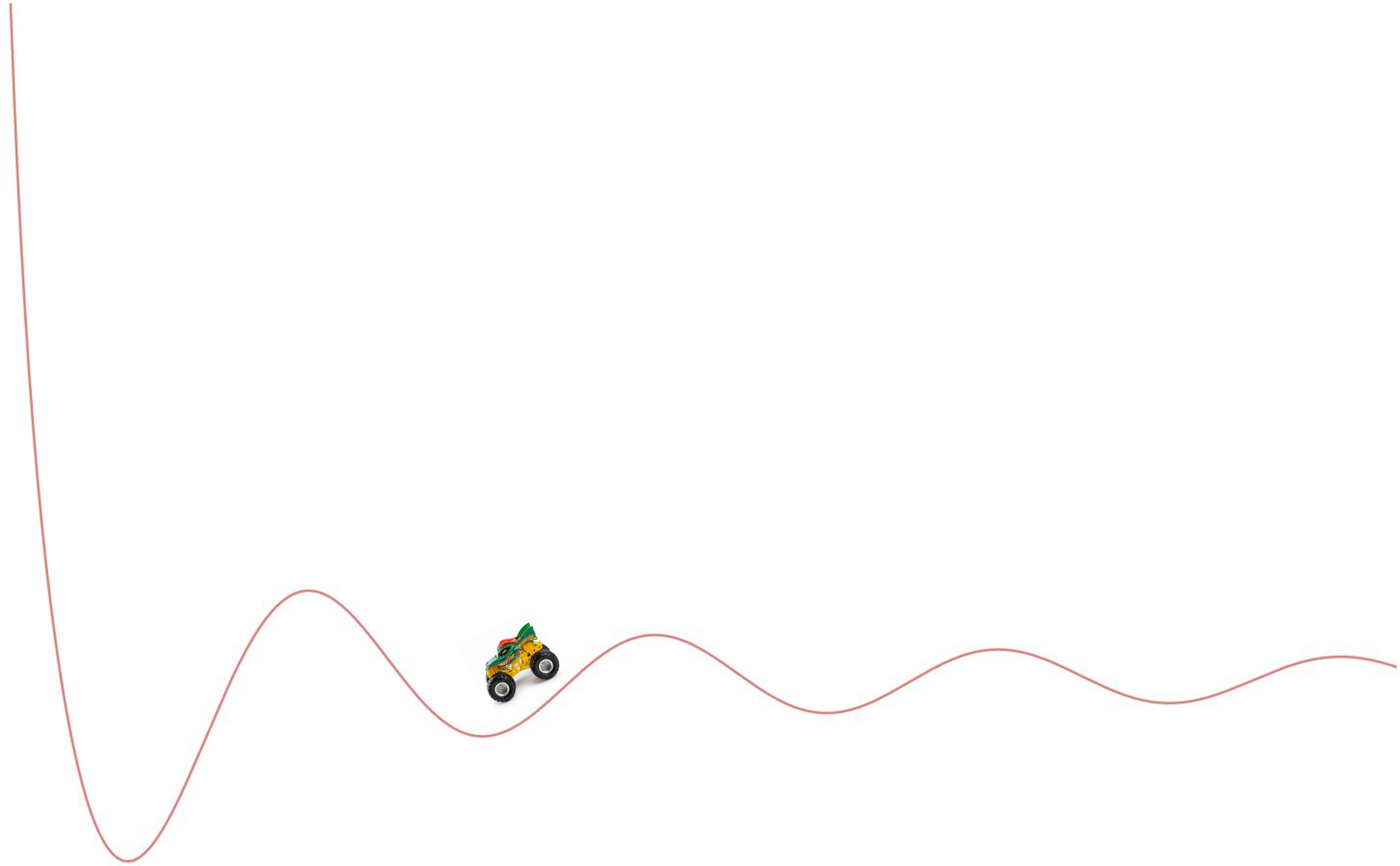
Descenso de gradiente:

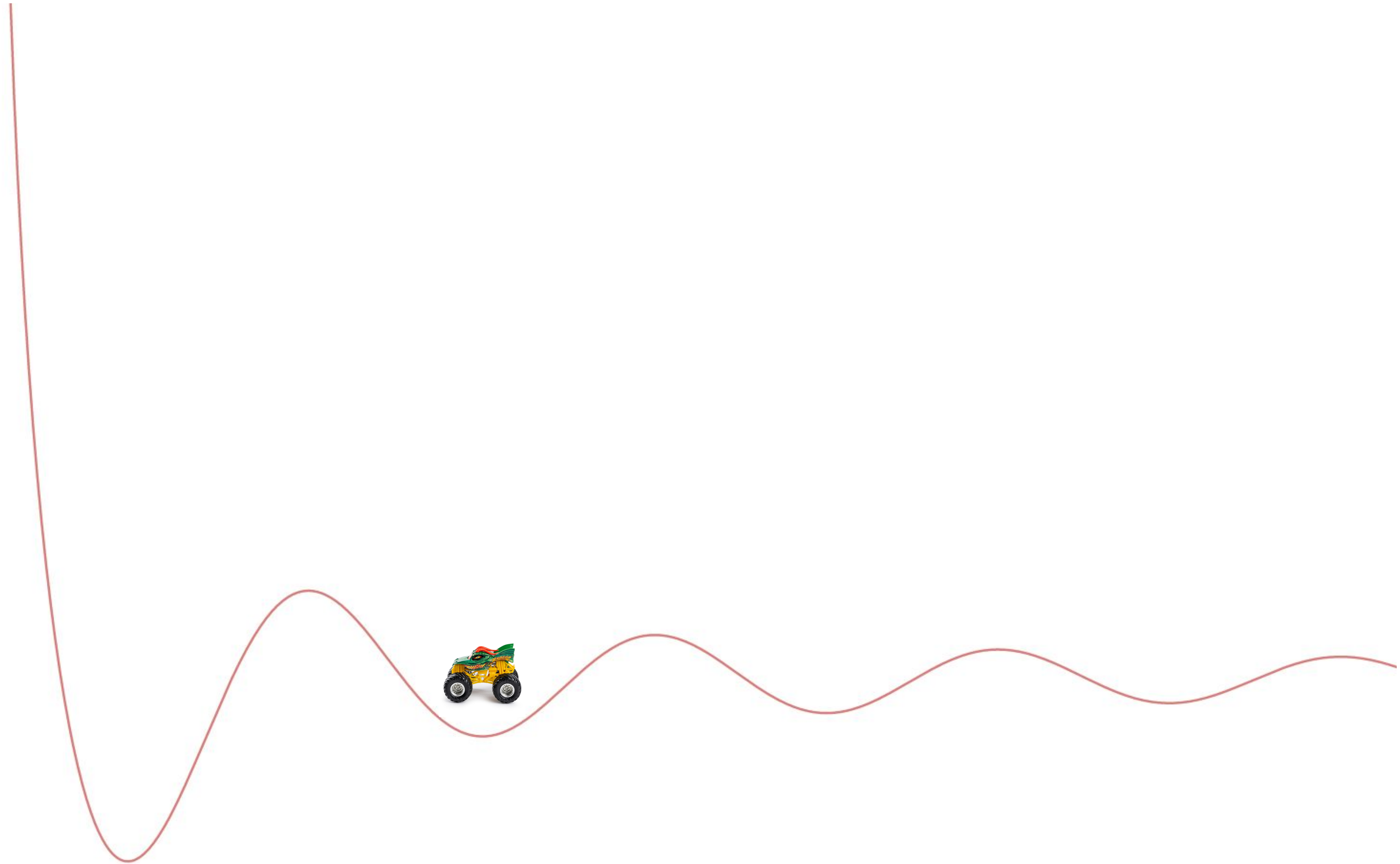
Aseguramos la optimalidad de la función de pérdida?

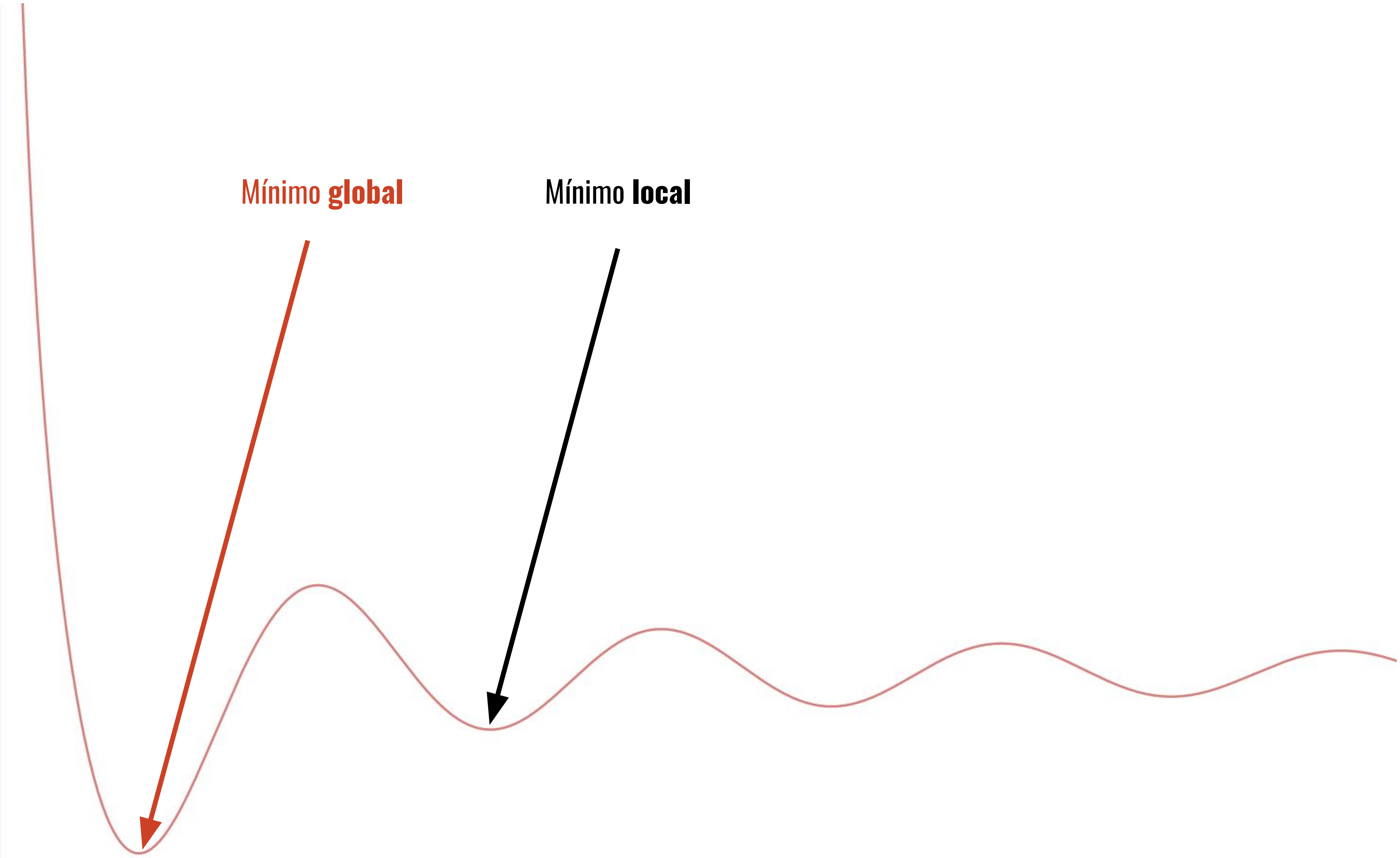






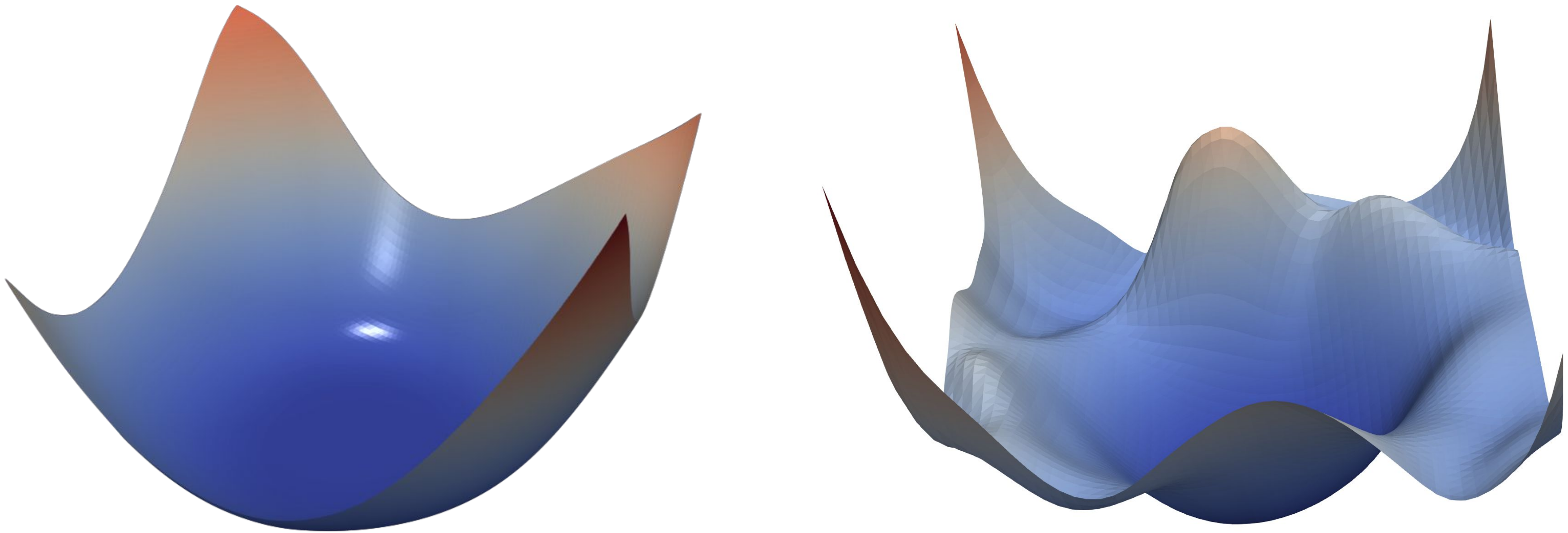


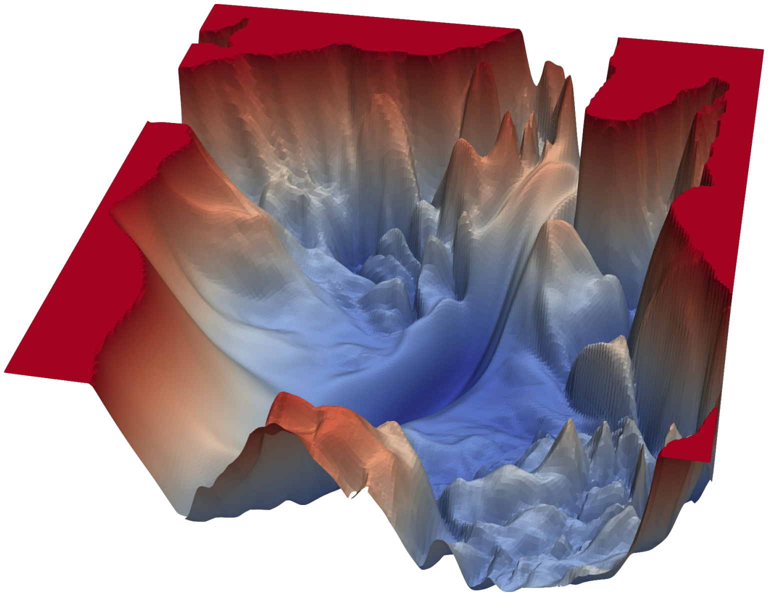


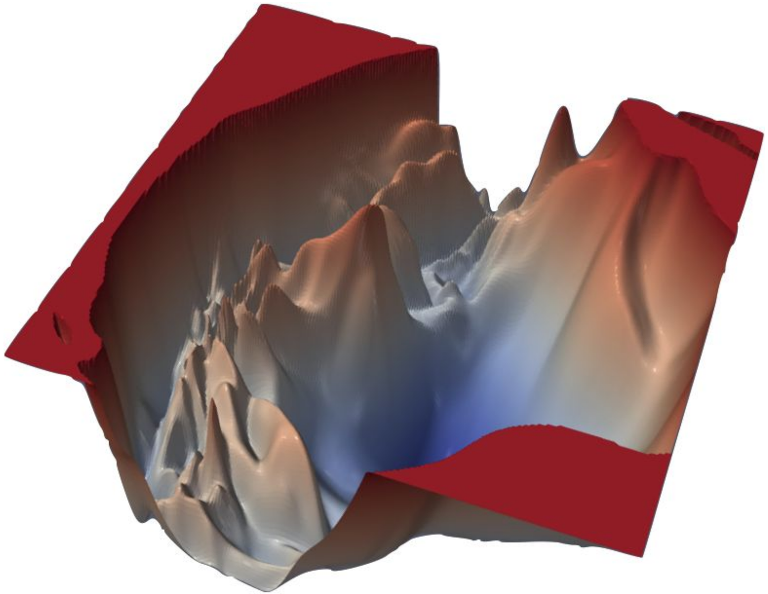

Mínimo **global**
Mínimo **local**

¿Aseguramos la optimalidad de la función de pérdida?

NO.



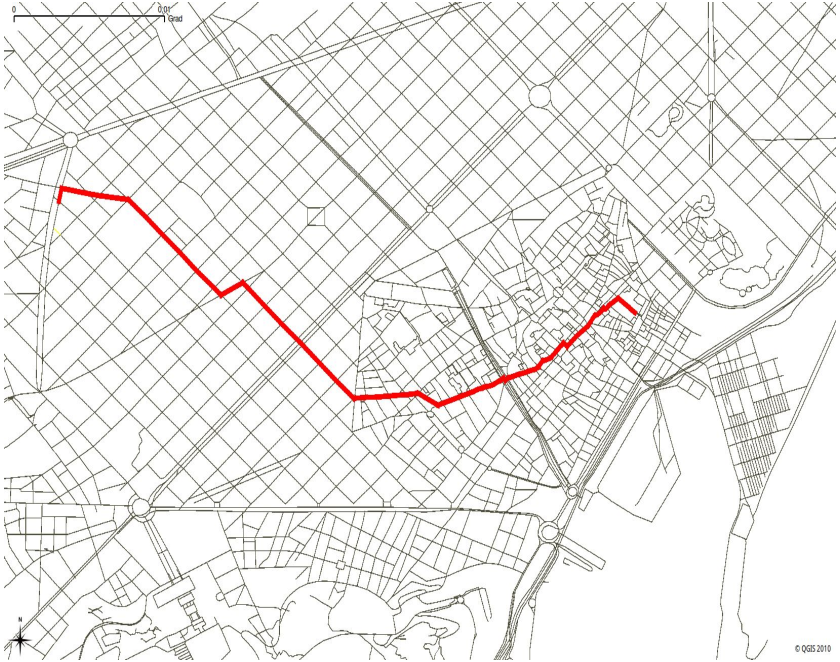




Si un algoritmo no asegura el óptimo, ¿Qué ganamos?

greedy goes brr

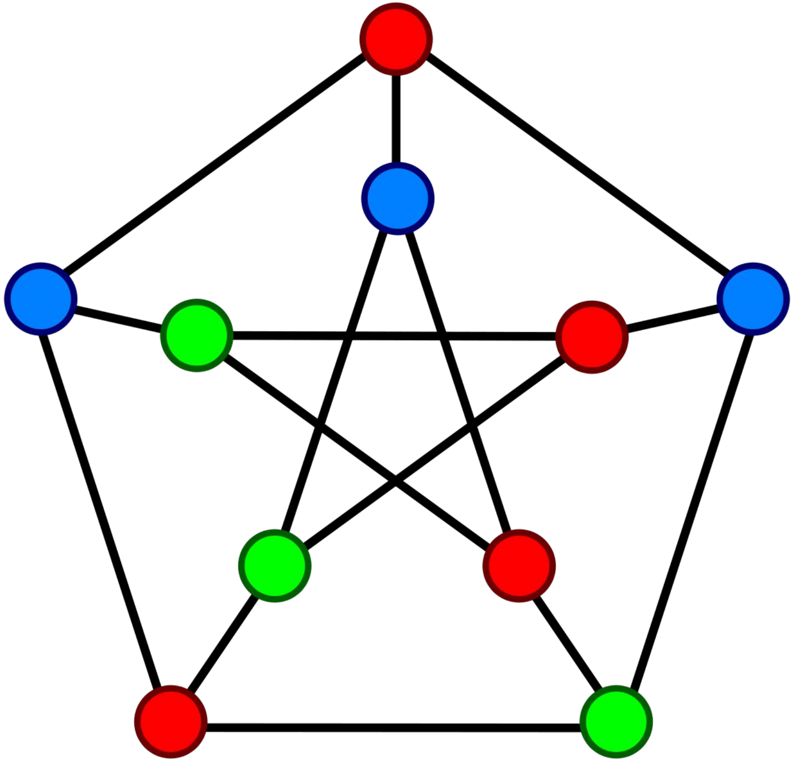
Clases de Complejidad (P, NP, EXP)



P

Problemas que pueden ser resueltos en tiempo polinomial

Ejemplos: ordenación, rutas más cortas, conectividad



NP

Coloquialmente:

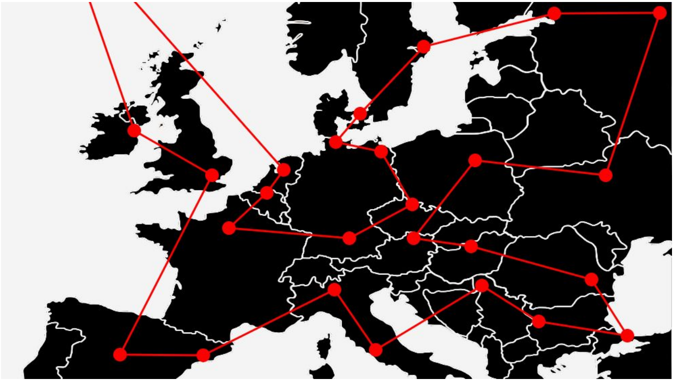
Problemas que no pueden ser resueltos en tiempo polinomial, pero que pueden ser verificados en tiempo polinomial

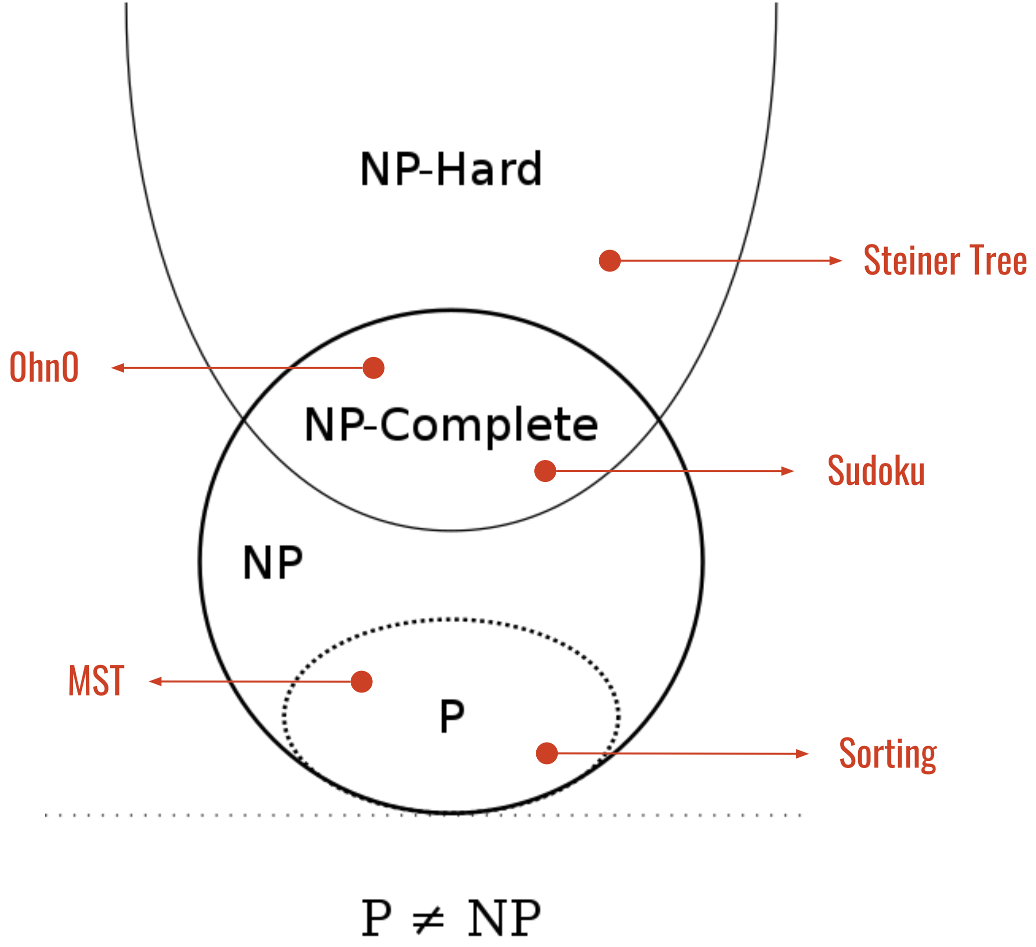
Ejemplos: factorización de enteros (RSA), isomorfismo de grafos (circuitos), coloración de grafos (asignación de recursos)

NP-Hard

Problemas que son al menos tan difíciles como los de clase NP.

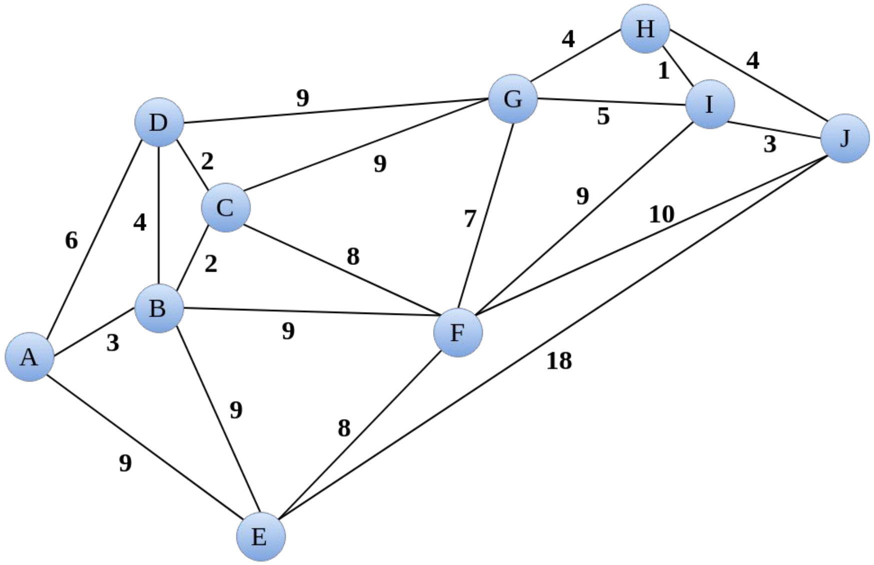
Ejemplos: Traveling Salesman (logística), validación de circuitos (embedded systems), Steiner tree (logística)



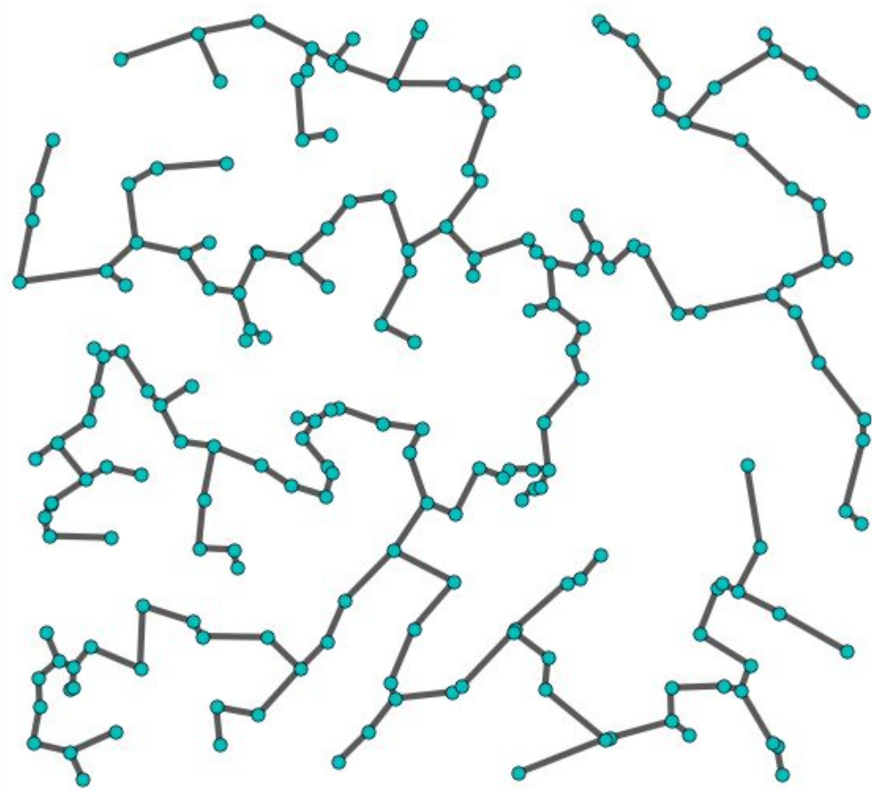

NP-Hard
Steiner Tree
OhnO
NP-Complete
Sudoku
NP
MST
P
Sorting
P ≠ NP

Algoritmos codiciosos al rescate:

Generación de óptimos locales eficientemente.



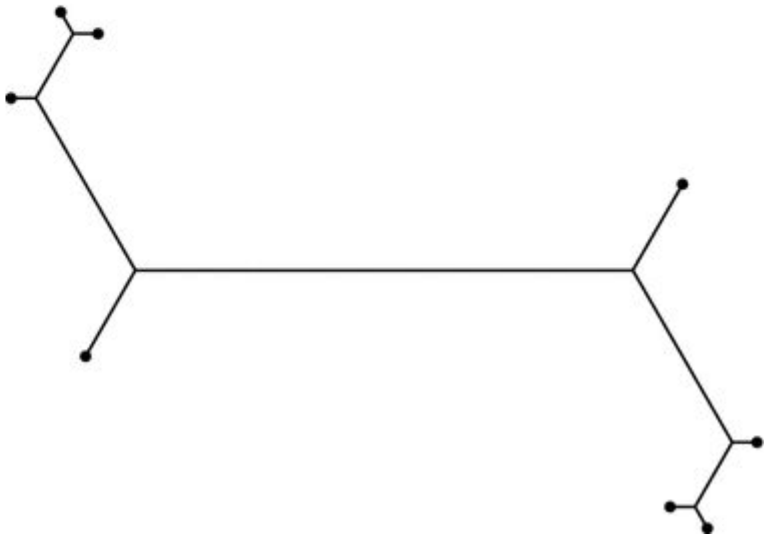
Tarea 4



Euclidean MST

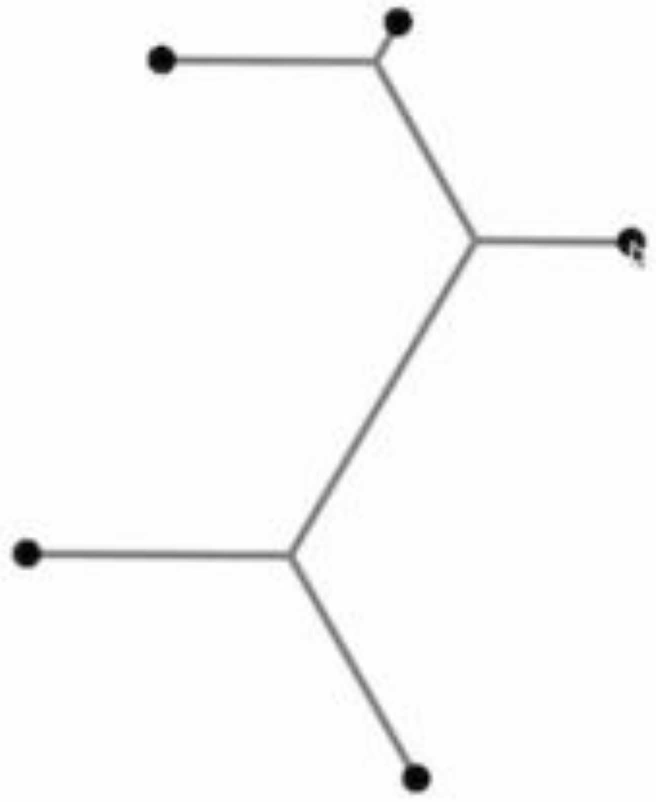
Dado n puntos en un plano R^2, el objetivo es conectar todos los puntos en una sola red tal que el costo total de la red sea mínimo. El costo de conectar dos puntos en el plano es su distancia euclidiana

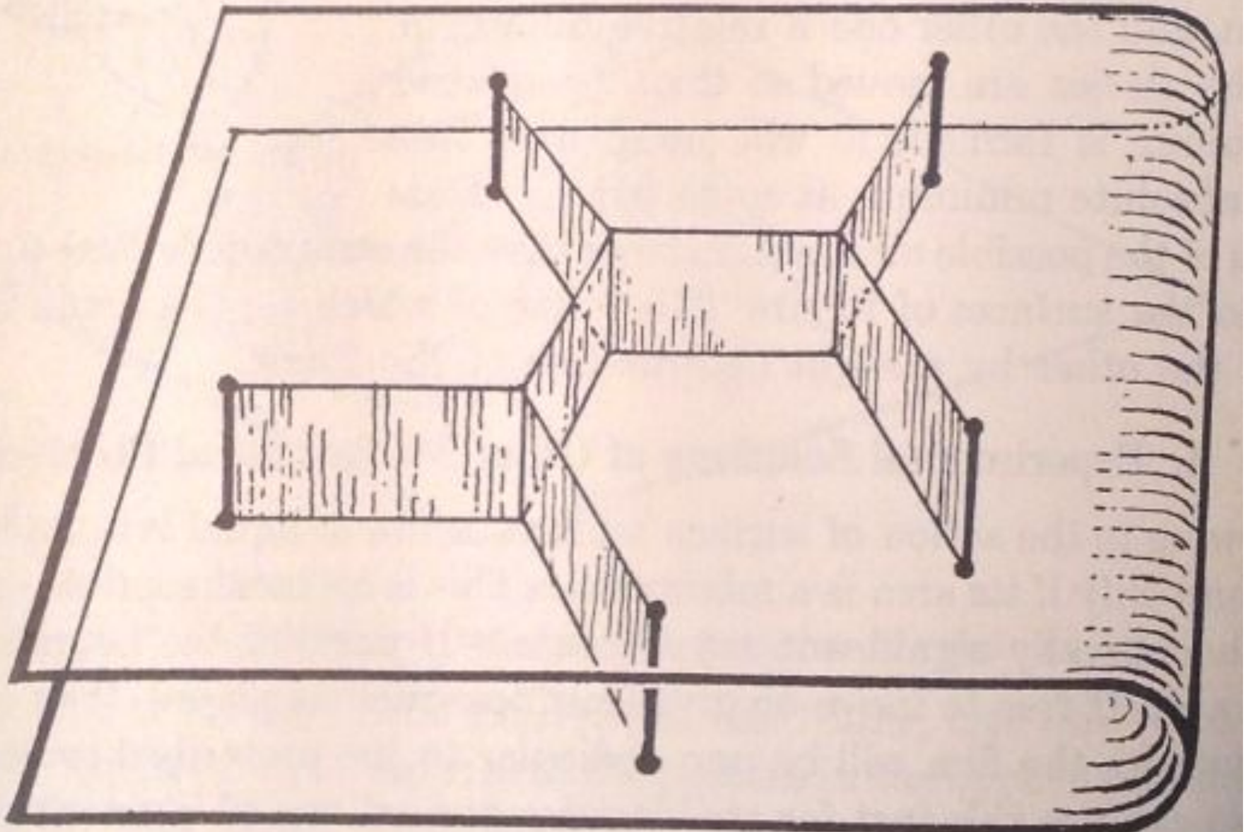
$$d(u, v) = \sqrt{(u_x - v_x)^2 + (u_y - v_y)^2}$$

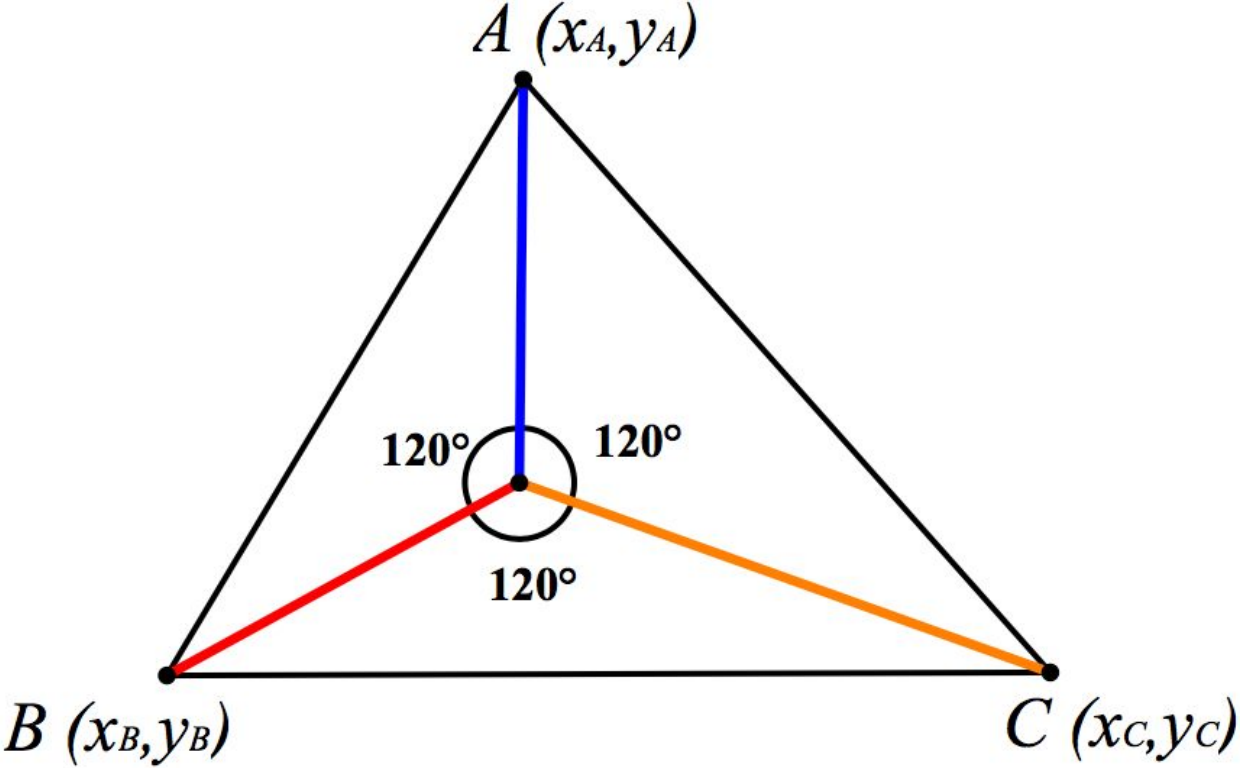


Steiner Tree

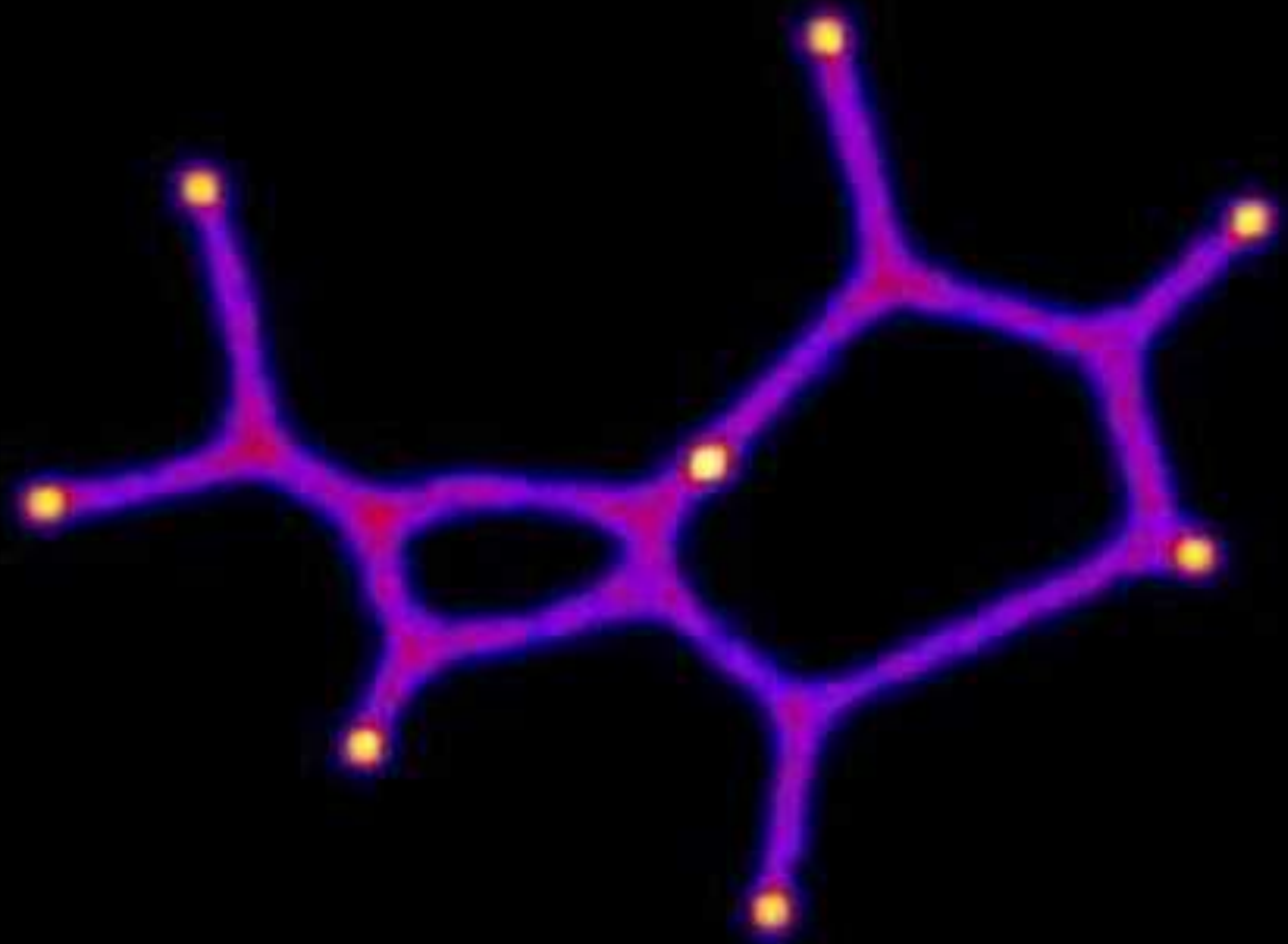
Agregar m puntos para minimizar el costo del Euclidean MST

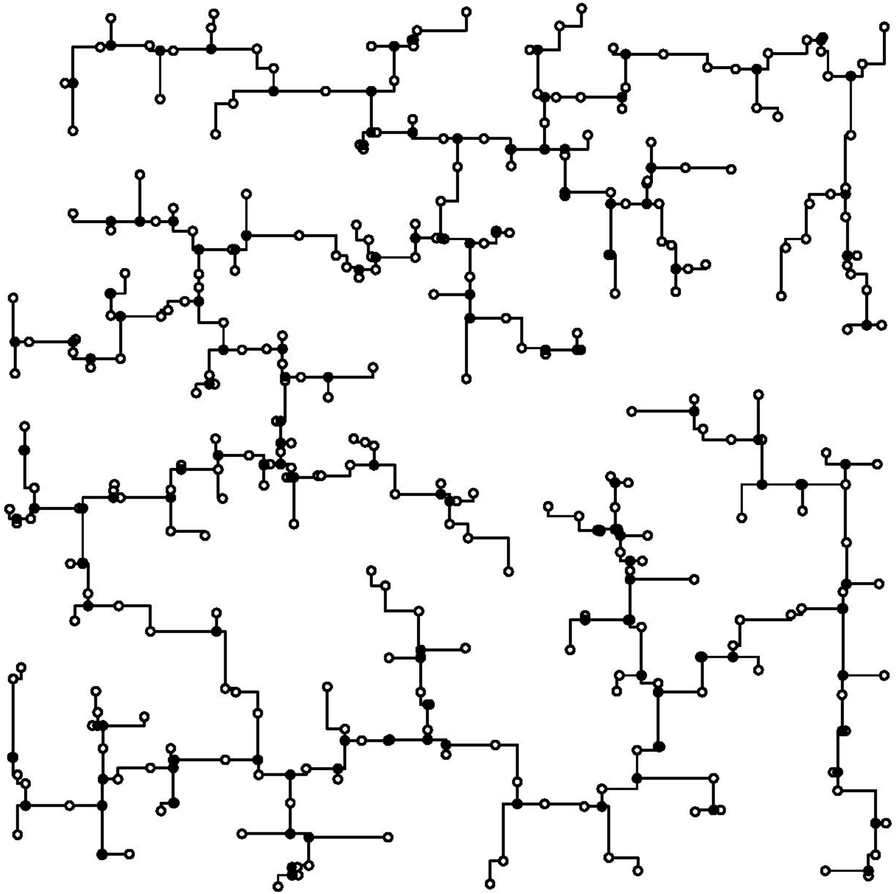






$A\ (x_A, y_A)$
$B\ (x_B, y_B)$
$C\ (x_C, y_C)$
120°
120°
120°





Rectilinear Steiner Tree:

Dado n puntos en un plano Z^2, el objetivo es conectar todos los puntos en una sola red tal que el costo total de la red sea mínimo. El costo de conectar dos puntos en el plano es su distancia Manhattan

$$d(u,v) = |(u_x - v_x)| + |(u_y - v_y)|$$

¿Qué es lo que se espera cuando se habla de un algoritmo codicioso para la tarea?

Un algoritmo que intente encontrar el costo mínimo global tomando el mínimo más prometedor en cada paso

T4

Steiner Tree Problem & Greedy Algorithms

A match made in heaven?