Algoritmos → Cotas (tiempo)
medir desempeño
(términos op. básicas)
entrada de tamaño n en ejecutarse

Cotas asintóticas → $O(n)$ ← Medir el tiempo abo x para una entrada n
Notación " Notación O grande en el peor caso
Big-O

$O(n^2)$ $O(n^3)$
↳ menor

Bonito / Facil

$O(n)$
$O(n^2)$
$O(\log n)$
$O(n^3)$
$O(n^5)$
} polinomial

Feo / Difícil

$O(n!)$ ← factorial

$O(2^n)$ ← exponencial

Coloración → Mínimo cromático ⇒ $n \geq 3$ ⇒ NP
NP-Completo

64 ⇒ $O(2^n)$

2^{64} ≈ 72 veces vida universo

Complejidad

- Clasificar en los problemas en distintos conjuntos (o clases de complejidad)

P (polinomia) NP (no determinista polinomia))
↳ NP hard
NP-completos

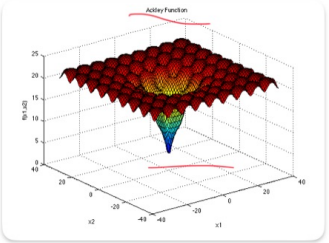
P. Decisión → Si/No

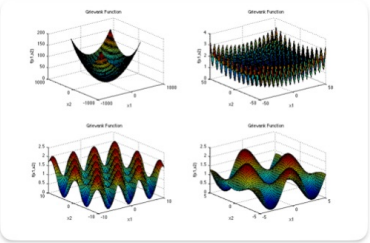
Si P. Decisión se resuelve en tiempo poli
⇒ P. Opt " " " "

Ejercicio 1. Funciones de prueba para optimización continua

1.1. Investiga algunas (al menos 3) funciones que se utilizan para probar métodos de optimización continua. Las funciones deben ser diferentes a las vistas en clase.
   a. Para cada función, da una breve descripción / discusión de sus características
   b. ¿Cuál podría ser más fácil y difícil de optimizar?, ¿Por qué?
   c. ¿La dimensión podría alterar la dificultad?

1.2. Implementar las funciones del inciso anterior
Cada función deberá estar implementada en un método (o función) diferente. En todos los casos, se debe considerar que el parámetro de entrada será el valor del punto a evaluar x; dependiendo del lenguaje y la implementación, se puede recibir también la dimensión del problema.





$$f(x) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

$$f(\mathbf{x}) = \sum_{i=1}^{d}\frac{x_i^2}{4000} - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

```python
def ackley(x):
    term1 = 20
    term2 = np.exp(1)
    term3 = -20*np.exp(-0.2*math.sqrt(np.sum(x**2)))
    term4 = np.exp(np.sum(np.cos(2*math.pi*x))/len(x))
    return term1+term2-term3-term4
```

```python
def griewank(x):
    term1 = 1
    term2 = np.sum((x**2)/4000)
    term3 = 1
    for i in range(len(x)):
        term3 *= np.cos(x[i]/math.sqrt(i+1))
    return term1+term2-term3
```

```python
def Ackley(x):
    if(len(x) == 0):
        print(f"Arreglo: {x}")
    res = 0
    res1 = 20*np.exp(-0.2*mt.sqrt(sphere(x)/len(x)))
    res2 = 0

    for i in range(len(x)):
        res2 += mt.cos(2*mt.pi*x[i])

    res = 20+mt.e-res1-mt.exp(res2/len(x))

    return res
```

```python
def Griewank(x):
    res = 0
    res1 = sphere(x)
    res2 = 0

    for i in range(len(x)):
        res2 *= mt.cos(x[i]/mt.sqrt(i+1))

    res = 1 + res1/4000 - res2

    return res
```

Ejercicio 3. Preguntas de repaso

1. Mencionar algún ejemplo de problema de optimización combinatoria, diferente a los mencionados en clase.
   Indicar claramente:
   - Espacio de búsqueda
   - Función objetivo
   - Tamaño del espacio de búsqueda
   - Ejemplar concreto del problema
   - Ejemplo de una solución

AGENTE VIAJERO, [TSP]

Ejemplar: Un conjunto finito de ciudades $\{c_1, c_2, \ldots, c_n\}$, una distancia positiva entera $d(i,j)$ entre cada par $\{c_i, c_j\}$, y un entero B.
Pregunta: ¿Existe una permutación π de $\{1, \ldots, n\}$ tal que

$$(\Sigma_{i=1}^{n-1} d(\pi(i), \pi(i+1))) + d(\pi(n), \pi(1)) \leq B$$

?

COLORACIÓN

Ejemplar: Una gráfica finita G y un entero K.
Pregunta: ¿G tiene una coloración con K colores?
- i.e., una asignación de colores a los vértices tales que dos vértices adyacentes tienen colores diferentes

SAT

Ejemplar: Una fórmula booleana ϕ
Pregunta: ¿ϕ es satisfacible?

scheduling (calendarización)
machine ← algoritmos