

# Recocido Simulado

---

Recocido en la Naturaleza	Recocido Simulado
Estado del Sistema	Solución candidata
Configuración molecular	Variables de decisión
Energía	Función objetivo
Estado Fundamental	Solución óptima global
Estado meta estable	Óptima local
Temperatura <i>muy alta</i>	Tendencia a explorar el espacio de búsqueda
Enfriamiento	Disminución de la tendencia a explorar
Cambio de configuraciones moleculares	Cambio de solución candidata

Temperatura Inicial  
adecuada

alazar

evaluar

- 1.b) Describe e implementa un esquema de enfriamiento.  
- Justifica tu elección del esquema de enfriamiento.

### Enfriamiento Lineal ✓

- Este es el esquema más simple. Se utiliza la siguiente función:

$$T_{k+1} = T_0 - \eta k$$

### Enfriamiento con Decremento Lento

$$T_{k+1} = T_k / (1 + \beta T_k)$$

### Enfriamiento Logarítmico

$$T_{k+1} = c / \ln k$$

- Dependientes de la dimensión Para funciones que tienen diferentes topologías entre diferentes dimensiones, se pueden utilizar diferentes esquemas de enfriamiento para diferentes dimensiones.

- 1.b) Describe e implementa un esquema de enfriamiento.  
- Justifica tu elección del esquema de enfriamiento.

```
# Esquema de enfriamiento exponencial
def esquema_enfriamiento(t, t_max, alpha=0.95):
    return t_max * alpha**t
```

- Geométrico, también referenciado como exponencial.

$$T_{k+1} = \alpha T_k, \alpha \in (0, 1)$$

$$\frac{0.8}{0.7}$$

Temperatura Inicial → Muy alta

5000

1000

200

- 1.a) Describe e implementa un operador de vecindad para soluciones binarias.
- La función solo debe generar un vecino de manera aleatoria.

```
# Operador de vecindad para soluciones binarias
def generar_vecino(solucion):
    vecino = np.copy(solucion)
    idx = np.random.randint(len(vecino))
    vecino[idx] = 1 - vecino[idx] # Cambiar aleatoriamente un bit
    return vecino
```

0 → 1  
1 → 0

## Recorrido Simulado

- Sol aleatoria inicial  $\leftarrow$  generarla

- evaluar Sol  $\leftarrow$  funci obj

probabilidad  $\downarrow$

$\rightarrow$  elegir solos peores

esq. enfriar + tiempo

x iteración

exploración

$\leftarrow$  sol actual  $\rightarrow$  búsqueda local  $\rightarrow$  explotación

- criterio parada  $\rightarrow$  devolver Mejor Sol

= n iteraciones , 100 000 , 1M

- n evaluaciones

- tiempo