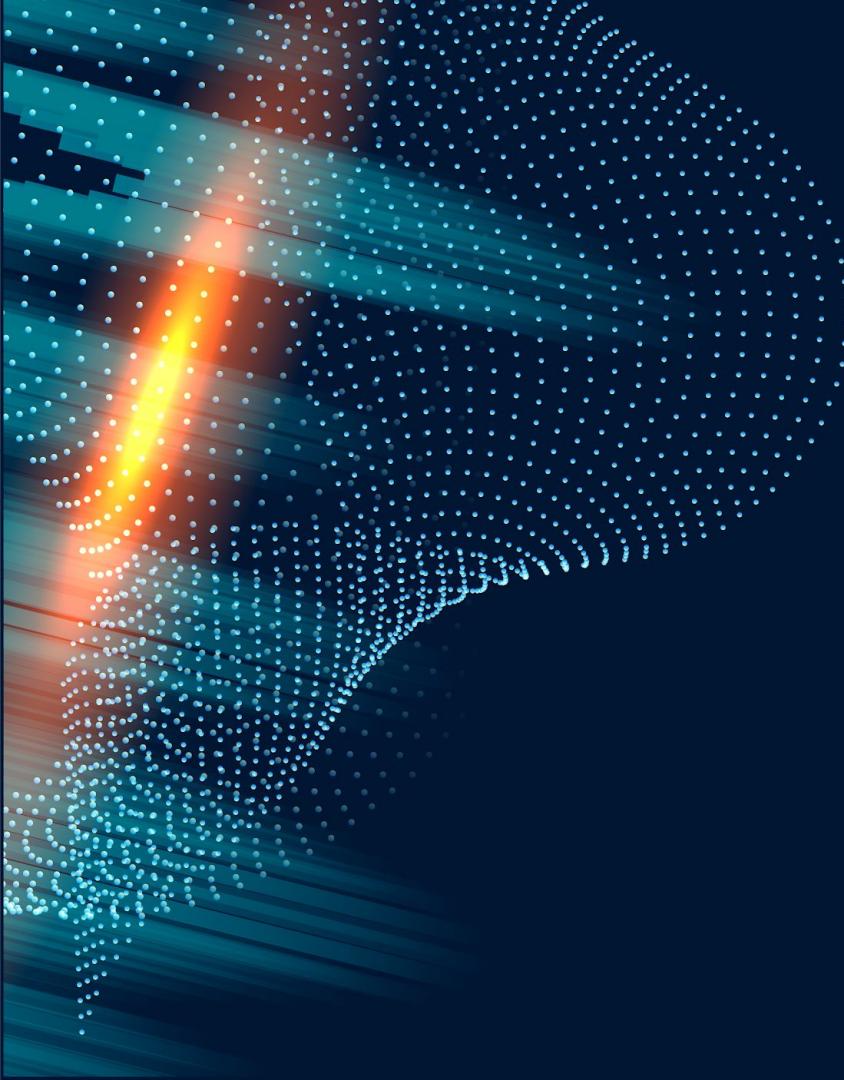


# Battle Royale Optimizer

Facundo Lella  
Alejandro Moreno



# ¿Qué es?

Battle Royale Optimizer (BRO) es una metaheurística game-based (inspirada en juegos tipo battle royale), propuesta originalmente para optimización continua de una sola función objetivo.

**01**

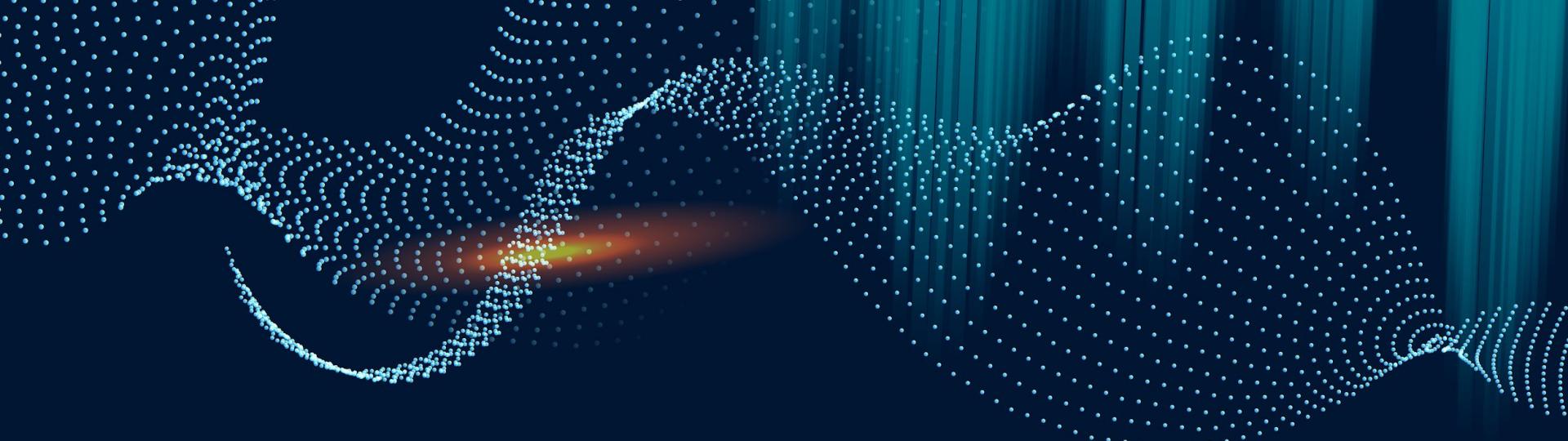
**¿Cómo  
funciona?**

**02**

**¿Cómo evita  
mínimos  
locales?**

**03**

**Aplicación de  
BRO**



# 01 | ¿Cómo funciona?

Battle Royale Optimizer

En cada iteración, agentes (soluciones) “pelean” con su vecino más cercano.

El ganador sigue igual y el perdedor se “daña” y se mueve hacia el mejor conocido, mientras que cada cierto tiempo el “círculo seguro” del espacio de búsqueda se encoge alrededor del mejor para intensificar la explotación.

---

## El paso a paso

- Se generan N soluciones al azar dentro de los límites por dimensión.
- Para cada agente, se busca su vecino más cercano (distancia euclídea) y se comparan fitness: el mejor es el ganador, el otro el perdedor. Se lleva un contador de daño: el perdedor incrementa su daño y el ganador lo resetea.
- El perdedor actualiza su posición acercándose al mejor encontrado (gbest) con un paso aleatorio:  
“posición nueva = posición actual +  $r \cdot (gbest - posición actual)$ ”, con  $r \in [0,1]$ . Esto implementa explotación guiada por el mejor global.



---

## El paso a paso

- Si un agente pierde repetidamente (superá un umbral de daño consecutivo), se elimina y se repone por una solución aleatoria (para reinyectar diversidad).
- Cada  $\Delta$  iteraciones, BRO encoge los límites por dimensión alrededor del mejor, usando la desviación estándar de la población. Este truco imita el círculo que se cierra en un battle royale y enfatiza la búsqueda fina cerca del mejor.
- Se actualiza el mejor global si apareció una solución superior; se repite hasta alcanzar el tope de iteraciones o el criterio de parada.

# Comparación con optimizadores tradicionales

- BRO se distingue por combinar la eliminación competitiva con la colaboración, superando los métodos tradicionales al evitar mínimos locales y mejorar las tasas de convergencia en tareas complejas.



# 02 | ¿Cómo evita los mínimos locales?

Battle Royale Optimizer

# **ESTRATEGIAS**

## **1. Duelos locales en lugar de selección global**

- No todos persiguen al mejor global al mismo tiempo, aumentando la probabilidad de descubrir soluciones mejores fuera del mínimo local

## **3. Movimiento parcial hacia el mejor**

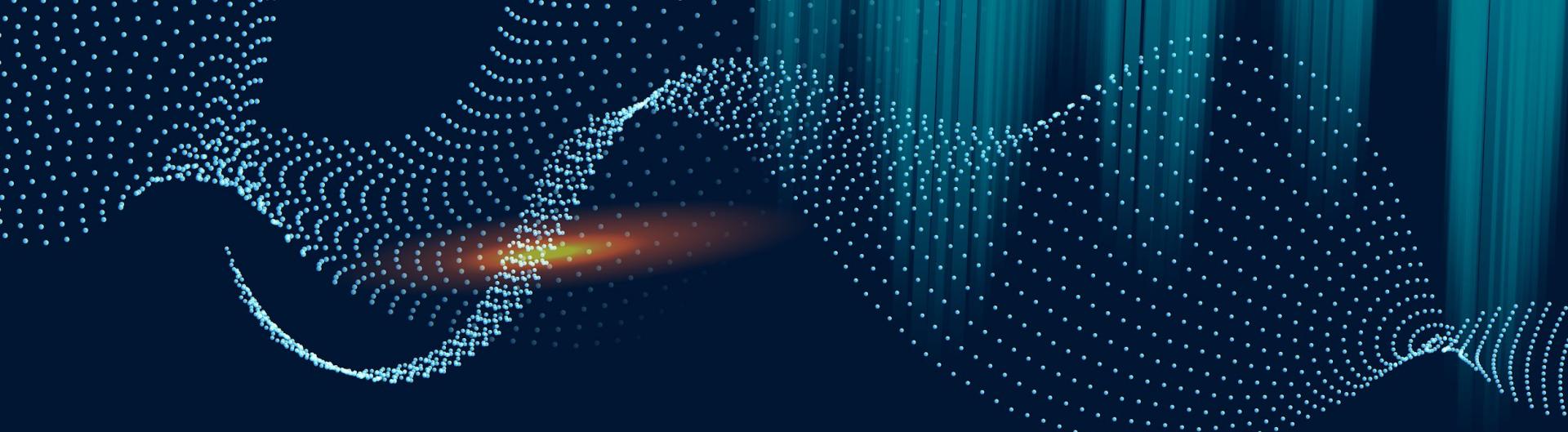
- Al usar un factor aleatorio  $r$ , el nuevo punto no coincide exactamente con el mejor. Esto produce variación estocástica, útil para escapar de pozos poco profundos.

## **2. Reemplazo de perdedores (diversidad)**

- Si pierde demasiados duelos seguidos, se elimina y se reemplaza por una solución aleatoria en el espacio de búsqueda.

## **4. Círculo que se cierra de forma gradual**

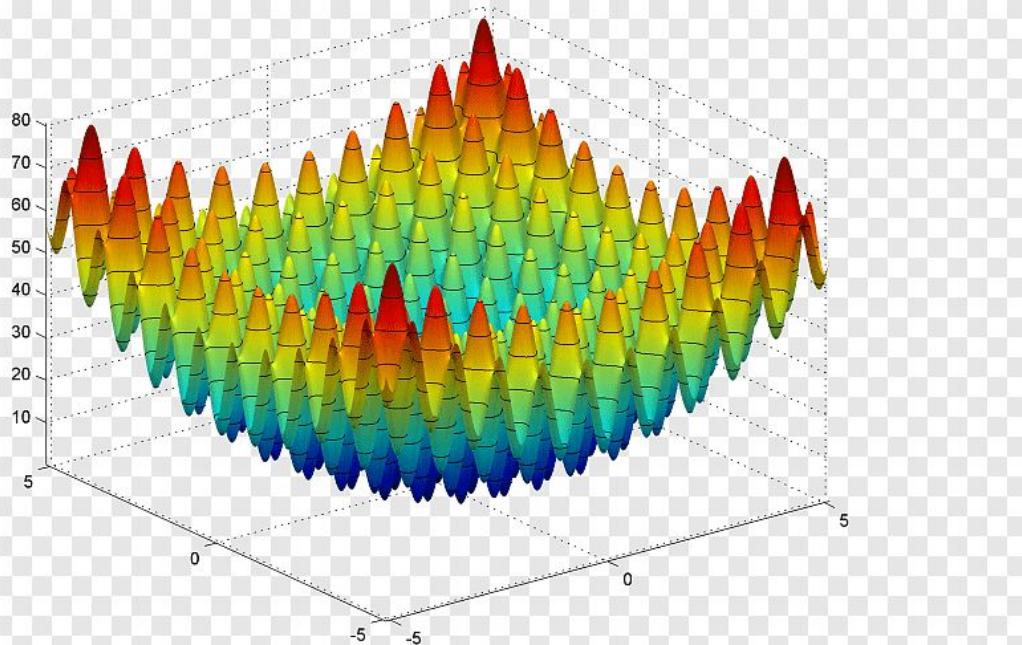
- Este enfoque gradual evita que la búsqueda se vuelva local demasiado pronto y permite seguir probando zonas nuevas antes de la fase de convergencia fina.



03

# APLICACIÓN DE BRO

Battle Royale Optimizer

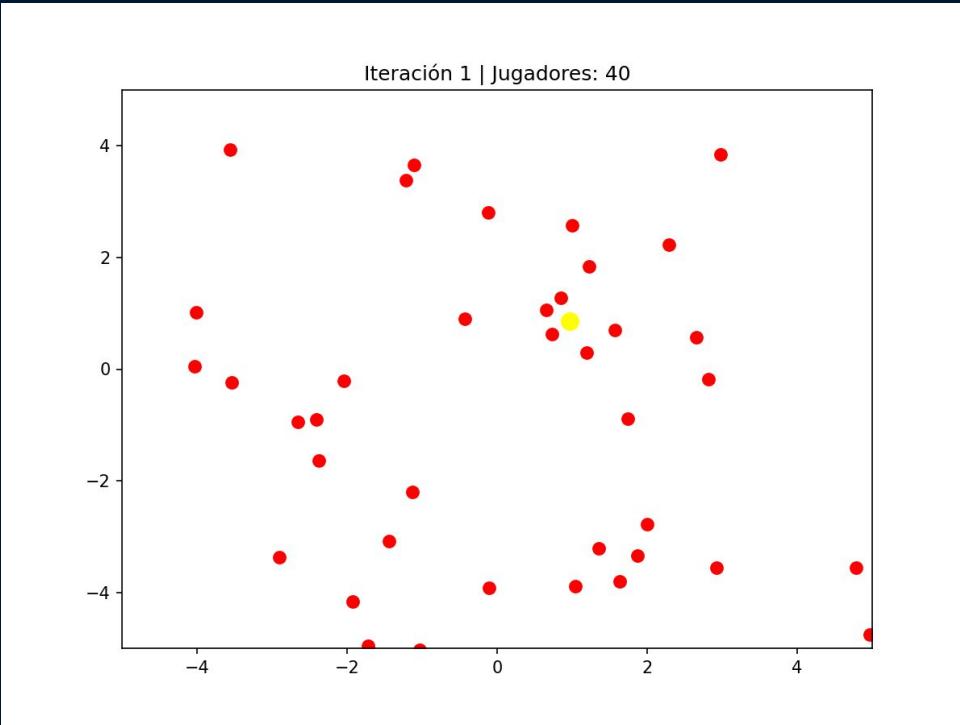


## PROBLEMA

ENCONTRAR EL MÍNIMO  
DE LA FUNCIÓN  
RASTRIGIN

# Visualización

gbest pbest



# GRACIAS

---

¿PREGUNTAS?