

# Algoritmo del Espíritu Musical (MSA)

## Una Metaheurística Bio-Inspirada para Optimización

AAB

Circe

16 de octubre de 2025

# ¿Qué es el Algoritmo del Espíritu Musical?

- Es una técnica de optimización metaheurística inspirada en la improvisación de un músico.
- Se basa en el proceso de buscar la mejor “melodía” o solución para un problema, mejorando iterativamente el rendimiento.
- El algoritmo utiliza una **Memoria Musical** para almacenar soluciones candidatas y sus aptitudes.
- Busca la solución óptima minimizando una **función objetivo**.

# Conceptos Clave del MSA

- **Melodía o Solución:** Un vector de variables de decisión que representa una posible solución.
- **Memoria del Músico (PM):** Una colección de las mejores melodías encontradas por un músico.
- **Memoria Musical (MM):** La memoria colectiva que almacena las memorias de todos los músicos.
- **Pitch Adjusting Rate (PAR):** La tasa de ajuste de tono. Controla la probabilidad de que una nueva melodía sea similar a una existente.
- **Bandwidth (bw):** Ancho de banda. Controla el rango de la búsqueda.

# Fases del Algoritmo MSA

El algoritmo se desarrolla en varias fases iterativas:

- **1. Inicialización:** Se crean melodías aleatorias para poblar la Memoria Musical.
- **2. Improvisación (SIS/GIS):** Se generan nuevas soluciones basadas en la memoria.
- **3. Comparación:** Las nuevas melodías se comparan con las peores de la memoria.
- **4. Actualización:** Si una nueva melodía es mejor, reemplaza a la peor.

Este ciclo se repite hasta alcanzar el criterio de terminación.

# Fórmulas de Improvisación

Las nuevas soluciones se generan de forma aleatoria, pero influenciadas por la memoria. El proceso se basa en dos variables clave:

- **'bw' (Bandwidth):** Determina la amplitud de la búsqueda.

$$bw_{Gen} = bw_{max} \cdot e^{bw_{normalized} \cdot t}$$

- **'PAR' (Pitch Adjusting Rate):** La probabilidad de que una nueva melodía “copie” una de la memoria.

$$PAR_{Gen} = PAR_{min} + (PAR_{normalized} \cdot t)$$

Donde  $t$  es la iteración actual.

# Implementación en el Código

El código utiliza una estructura de clases para gestionar cada componente:

- **'MemoryManager'**: Inicializa y ordena la memoria musical.
- **'Improviser'**: Genera nuevas soluciones basándose en la Memoria Musical, ' $PAR_{Gen}$ ' y ' $bw_{Gen}$ '.
- **'Comparator'**: Compara y actualiza las soluciones.
- **'FeasibleRangeUpdater'**: En la fase GIS, acota el rango de búsqueda para converger más rápido.

# Parámetros Clave del MSA

- **'N'**: Número de variables de decisión.
- **'PMN'**: Número de músicos (memoria de jugadores).
- **'PMS'**: Tamaño de la memoria de cada músico.
- **'PMCR'**: Tasa de consideración de la memoria del músico.
- **'NI' / 'NII'**: Número de iteraciones para la improvisación a corto y largo plazo.
- **' $X_{lower}$ ' / ' $X_{upper}$ '** : *Límites inferiores y superiores de las variables.*

- **Estructura simple:** Fácil de entender e implementar.
- **Exploración y Explotación:** Combina eficazmente la exploración del espacio de búsqueda con la explotación de las mejores soluciones.
- **Adaptabilidad:** Los parámetros 'PAR' y 'bw' se adaptan con cada iteración para guiar la búsqueda.
- Es una alternativa viable a otros algoritmos metaheurísticos como los Algoritmos Genéticos (AG) y PSO.

# Desventajas del MSA

- La calidad de la solución depende de una buena selección de parámetros iniciales.
- No garantiza encontrar la solución óptima global, solo una solución "suficientemente buena".
- El rendimiento puede variar significativamente entre diferentes problemas.

- **Problemas de Optimización Continua:** Como el problema de la función ' $f(x)$ ' en el código.
- **Ingeniería:** Optimización de diseños y parámetros.
- **Robótica:** Planificación de trayectorias.
- **Inteligencia Artificial:** Ajuste de hiperparámetros de modelos de Machine Learning.

# Ejemplo de la Función Objetivo en el Código

La función a minimizar es:

## Función a Optimizar

$$f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$

El código está diseñado para encontrar los valores de  $x_1$  y  $x_2$  que minimizan esta función. La solución óptima es  $x_1 = 10$  y  $x_2 = 20$ .

## Ejemplo de un Resultado (según el código)

El código genera una gráfica que muestra la convergencia del algoritmo.

- La línea azul ('Mean Fitness') muestra el promedio de la aptitud de todas las soluciones en la memoria.
- La línea roja ('Min Fitness') muestra la mejor solución encontrada en cada iteración.

El objetivo es ver cómo la línea roja desciende hasta un valor mínimo.

# Consideraciones Prácticas

- La inicialización aleatoria de la población es crucial para una buena exploración del espacio de búsqueda.
- El balance entre la exploración (fase SIS) y la explotación (fase GIS) es clave para la eficiencia del algoritmo.
- El 'MSAOptimizer' gestiona todo el flujo, desde la inicialización hasta la visualización de los resultados.

# Conclusión

- El MSA es una poderosa herramienta de optimización, inspirada en la creatividad y la habilidad de los músicos.
- Su simplicidad y capacidad de adaptación lo convierten en una excelente opción para resolver problemas complejos.
- La clave de su éxito reside en la interacción y la mejora continua de sus “melodías”.