UNIVERSIDAD DIEGO PORTALES

Junio, 2025

Tarea 3: Funciones multimodales

Integrantes

José Martín Berríos Piña

Renato Óscar Benjamín Contreras Carvajal

Índice

1. Introducción **2**

2. Evolución Diferencial: Fundamentos y Justificación **2**
2.1. Mecanismo del Algoritmo de Evolución Diferencial 2
2.2. Justificación Específica de la Elección de DE 3

3. Diseño e Implementación de la Metaheurística DE **4**
3.1. Componentes Clave y Pseudocódigo Conceptual de DE 4
3.2. Manejo de Funciones Objetivo, Dominios y Restricciones 6
3.3. Fijación de Semillas para Reproducibilidad 7

4. Configuración Experimental **7**
4.1. Funciones de Prueba 7
4.2. Configuraciones de Parámetros de DE 8
4.3. Metodología de Ejecución 8
4.4. Métricas de Rendimiento Utilizadas 9

5. Resultados y Análisis **9**
5.1. Función $f_1(x)$ 9
5.2. Función $f_2(x)$ 10
5.3. Función $f_3(x)$ 11
5.4. Función $f_4(x)$ 12
5.5. Análisis del Desempeño y Discusión General 13
5.5.1. Análisis Específico por Función y Robustez de DE 13
5.5.2. Impacto Detallado de los Parámetros (NP, F, CR) y Balance Exploración-Explotación 14
5.5.3. Comportamiento General de DE frente a la Dificultad de las Funciones . . . 16

6. Conclusiones **16**

7. Anexos **18**

1. Introducción

El presente trabajo aborda el desafío de la optimización de funciones multimodales. Estas funciones se caracterizan por presentar múltiples óptimos locales, lo que dificulta la convergencia hacia el óptimo global mediante métodos tradicionales. En este contexto, se emplearán metaheurísticas basadas en población, que suelen ofrecer mejores resultados para este tipo de problemas.

En particular, esta tarea se centrará en el diseño e implementación en Python del algoritmo de Evolución Diferencial (DE), una metaheurística seleccionada entre las opciones de Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Genetic Algorithm (GA), y Bat Algorithm (BA). La elección de DE se debe a sus características que se justificarán en detalle más adelante. El algoritmo DE implementado se aplicará con el fin de encontrar el mínimo de cuatro funciones multimodales específicas (f_1, f_2, f_3, f_4), todas definidas en dominios continuos.

Una parte fundamental del estudio consistirá en la experimentación con la metaheurística DE. Para ello, se realizarán 10 ejecuciones para cada una de las cuatro funciones objetivo bajo cada una de las **diez configuraciones de parámetros** seleccionadas. Se analizará detalladamente el impacto que los diferentes valores de dichos parámetros tienen en el rendimiento y la capacidad de convergencia del algoritmo. Con el fin de que estos resultados nos den conclusiones sobre la efectividad de DE y la influencia de su parametrización en la optimización de las funciones multimodales propuestas. Los códigos y gráficos extras se pueden ver en el siguiente Github: https://github.com/renato357/Tarea-3-JB-RC-AyM

2. Evolución Diferencial: Fundamentos y Justificación

Para la optimización de las funciones multimodales presentadas en esta tarea, y en concordancia con el requisito de diseñar e implementar una metaheurística, se ha seleccionado el algoritmo de **Evolución Diferencial (DE)**. Esta elección se fundamenta en los resultados experimentales obtenidos, sus características teóricas y su adecuada adaptación a la naturaleza de los problemas de optimización multimodal, tal como se detalla a continuación. Las funciones multimodales presentan el desafío de múltiples óptimos locales, donde métodos tradicionales pueden tener un desempeño limitado, siendo recomendable el uso de metaheurísticas basadas en población.

2.1. Mecanismo del Algoritmo de Evolución Diferencial

DE es reconocido en la literatura como un algoritmo de optimización global potente y robusto, especialmente para problemas con dominios continuos como los de esta tarea. Una de sus ventajas es su relativa simplicidad conceptual, contando con pocos parámetros de control principales: el tamaño de la población (`NP`), el factor de mutación (F) para la mutación y la tasa de cruce (CR) para el cruce.

La clave de su efectividad, particularmente en casos de solución complejos y multimodales, reside en su particular estrategia de generación de nuevas soluciones. A diferencia de otros algoritmos evolutivos, DE crea nuevas soluciones candidatas utilizando las diferencias vectoriales entre individuos

existentes de la población. Una estrategia común, y la utilizada en este trabajo, es la denominada 'rand/1', donde para generar un vector mutado V_i asociado a un individuo objetivo X_i, se seleccionan tres individuos distintos (X_{r1}, X_{r2}, X_{r3}) de la población actual, tales que $i \neq r1 \neq r2 \neq r3$. El vector mutado se crea mediante la operación $V_i = X_{r1} + F \times (X_{r2} - X_{r3})$. Aquí, F es el factor de mutación que controla la amplitud de la perturbación.

Posteriormente, se aplica un operador de cruce (comúnmente binomial) entre el vector objetivo X_i y el vector mutado V_i para generar un vector de prueba U_i. Cada componente j del vector de prueba $U_{i,j}$ se toma del vector mutado $V_{i,j}$ si un número aleatorio $rand_j[0, 1)$ es menor o igual a CR, o si j es un índice aleatorio j_{rand} (para asegurar que U_i obtenga al menos una componente de V_i). En caso contrario, se toma de $X_{i,j}$. Finalmente, se aplica una selección voraz: si el vector de prueba U_i ofrece un mejor valor de la función objetivo que X_i, entonces U_i reemplaza a X_i en la siguiente generación.

Esta forma de operar le da a DE ventajas para enfrentar funciones con múltiples óptimos locales. La exploración del espacio de búsqueda se ajusta de forma natural: diferencias grandes entre individuos dispersos promueven saltos amplios (exploración), mientras que diferencias pequeñas en poblaciones concentradas permiten un ajuste fino (explotación). Crucialmente, las diferencias vectoriales permiten que las soluciones candidatas 'salten' entre diferentes valles o picos, mejorando la capacidad de escapar de óptimos locales.

2.2. Justificación Específica de la Elección de DE

Los experimentos preliminares realizados, consistentes en 10 ejecuciones con distintas configuraciones de parámetros para cada metaheurística evaluada (PSO, ACO, GA, DE, BA) sobre las diversas funciones, fueron determinantes para la selección. Específicamente, la decisión de utilizar DE se consolidó al observar su rendimiento consistentemente bueno y robusto en la función f_3. La Figure 1 ilustra este punto, mostrando cómo la mayoría de las 10 configuraciones de DE probadas para f_3 lograron alcanzar valores del costo de la función significativamente bajos.

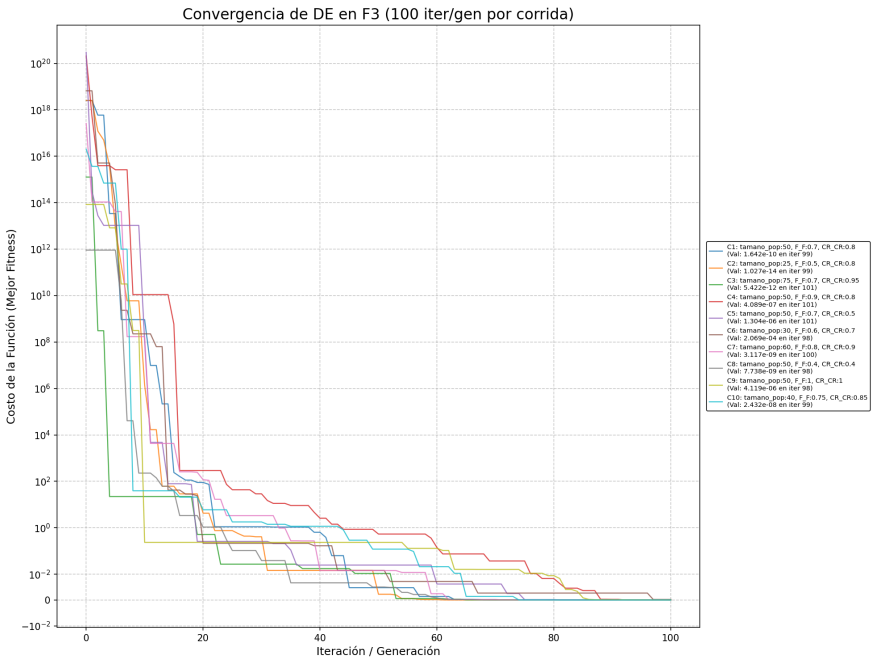


Figura 1: Convergencia de DE en la función f_3 con 10 configuraciones de parámetros distintas (100 iteraciones) durante la fase preliminar de selección de metaheurística.

Al analizar el gráfico de f_3 para DE (Figure 1), se constata que múltiples configuraciones convergen a valores del orden de 10^{-1} o inferiores, lo cual no fue consistentemente observado con las otras metaheurísticas. Por ejemplo, la configuración identificada como 'C2' en los experimentos preliminares (con parámetros `NP=25, F=0.5, CR=0.8`) alcanzó un valor de $1{,}027 \times 10^{-14}$ para f_3. Para una perspectiva comparativa del desempeño con otras metaheurísticas en esta misma función, se pueden consultar los gráficos de convergencia presentados en los Anexos (Figure 8 para ACO, Figure 9 para BA, Figure 10 para GA y Figure 11 para PSO).

3. Diseño e Implementación de la Metaheurística DE

En esta sección se detalla la implementación del algoritmo de Evolución Diferencial utilizado, basándose en el archivo `de_optimizer.py`. Se presenta un pseudocódigo conceptual de DE y luego se especifica cómo la implementación propia aborda los componentes clave.

3.1. Componentes Clave y Pseudocódigo Conceptual de DE

La Evolución Diferencial es un algoritmo basado en población que optimiza un problema intentando mejorar iterativamente una población de soluciones candidatas con respecto a una medida de calidad dada (fitness). Los componentes principales del DE clásico son:

- **Inicialización:** Se crea una población inicial de NP individuos (soluciones candidatas), donde cada individuo es un vector de D variables. Estos individuos se generan aleatoriamente dentro de los límites permitidos para cada variable del problema.

- **Mutación:** Para cada individuo X_i de la población (vector objetivo), se genera un vector mutante V_i. Una estrategia común, y la utilizada en este trabajo (conocida como 'rand/1'), consiste en seleccionar tres individuos distintos (X_{r1}, X_{r2}, X_{r3}) de la población actual (que no sean X_i) y calcular el vector mutante como $V_i = X_{r1} + F \cdot (X_{r2} - X_{r3})$, donde F es el factor de mutación que escala la diferencia vectorial.
- **Cruce (Recombinación):** Se crea un vector de prueba U_i combinando las componentes del vector objetivo X_i con las del vector mutante V_i. En el cruce binomial, cada componente j de U_i se toma de $V_{i,j}$ con una probabilidad CR (tasa de cruce), o de $X_{i,j}$ con probabilidad $1 - CR$. Para asegurar que U_i difiera de X_i, al menos una componente se toma de V_i.
- **Selección:** Se compara el vector de prueba U_i con el vector objetivo X_i. Si U_i tiene un valor de fitness mejor (o igual) que X_i, entonces U_i reemplaza a X_i en la población de la siguiente generación; de lo contrario, X_i se mantiene.

Estos pasos de mutación, cruce y selección se repiten para todos los individuos de la población durante un número determinado de generaciones. El Algorithm 1 presenta un pseudocódigo conceptual de este proceso.

Algorithm 1 Pseudocódigo Conceptual de Evolución Diferencial (DE)

Definir: NP (tamaño de población), F (factor de mutación), CR (tasa de cruce), G_{max} (máximo de generaciones).
Inicializar: Generar aleatoriamente población $P_0 = \{X_{1,0}, X_{2,0}, ..., X_{NP,0}\}$ dentro de los límites.
Evaluar fitness de cada individuo en P_0.
for $g = 0$ hasta $G_{max} - 1$ **do** ▷ Para cada generación
    for $i = 1$ hasta NP **do** ▷ Para cada individuo $X_{i,g}$ en la población P_g
        Paso de Mutación:
        Seleccionar tres individuos aleatorios $X_{r1,g}, X_{r2,g}, X_{r3,g}$ de P_g, distintos entre sí y de $X_{i,g}$.
        Calcular vector mutante: $V_{i,g} = X_{r1,g} + F \cdot (X_{r2,g} - X_{r3,g})$.
        Paso de Cruce:
        Crear vector de prueba $U_{i,g} = (u_{1,i,g}, u_{2,i,g}, ..., u_{D,i,g})$.
        Seleccionar un índice aleatorio $j_{rand} \in \{1, ..., D\}$.
        for $j = 1$ hasta D **do** ▷ Para cada dimensión
            if random$(0, 1) < CR$ **or** $j = j_{rand}$ **then**
                $u_{j,i,g} = v_{j,i,g}$ ▷ Componente del vector mutante
            else
                $u_{j,i,g} = x_{j,i,g}$ ▷ Componente del vector objetivo
        Paso de Selección:
        Evaluar fitness de $U_{i,g}$.
        if $fitness(U_{i,g}) \leq fitness(X_{i,g})$ **then**
            $X_{i,g+1} = U_{i,g}$ ▷ $U_{i,g}$ reemplaza a $X_{i,g}$ en la siguiente generación
        else
            $X_{i,g+1} = X_{i,g}$ ▷ $X_{i,g}$ se mantiene
return El individuo con el mejor fitness de la población final $P_{G_{max}}$.

La implementación específica utilizada en este trabajo, detallada en el archivo `de_optimizer.py`, sigue este flujo general. La clase `DifferentialEvolution` encapsula la lógica, y su método `optimizar` coordina las generaciones. Los detalles como el manejo de los límites de las variables (clipping), la estructura para definir los problemas (diccionario `PROBLEMAS`), y el seguimiento del historial de convergencia son particularidades de dicha implementación.

3.2. Manejo de Funciones Objetivo, Dominios y Restricciones

La gestión de las diferentes funciones, sus dimensionalidades, funciones de restricción y límites de dominio se centralizó en un diccionario llamado `PROBLEMAS` en `de_optimizer.py`. Cada entrada de este diccionario contiene la `funcion_objetivo`, `funcion_restriccion`, `limites_dominio_pdf` (informativos), `limites_generacion` (usados activamente) y la `dimension`.

Para cada una de las cuatro funciones objetivo (f_1, f_2, f_3, f_4), la representación de un individuo es un vector de números reales $x = (x_1, x_2, ..., x_D)$. El manejo de los dominios específicos de cada varia-

ble x_j en cada función se implementó de la siguiente manera (ver método `_inicializar_poblacion` y `_aplicar_limites_vector` en `de_optimizer.py`):

- **Inicialización:** Los individuos de la población inicial se generan aleatoriamente asegurando que cada componente x_j se encuentre dentro de su rango $[lim_{inf,j}, lim_{sup,j}]$ especificado en `limites_generacion` para la función correspondiente. Se verifica que el individuo generado cumpla con `funcion_restriccion` antes de ser aceptado.
- **Manejo de Límites Post-Mutación:** Si alguna componente x_j del vector mutado V_i excede sus límites (definidos en `limites_generacion`), se aplica una estrategia de "clipping" implementada en el método `_aplicar_limites_vector`. Es decir, si $V_{i,j} < lim_{inf,j}$, entonces $V_{i,j} = lim_{inf,j}$; si $V_{i,j} > lim_{sup,j}$, entonces $V_{i,j} = lim_{sup,j}$.
- **Restricciones en Selección:** Antes de evaluar el fitness de un vector de prueba U_i, se verifica mediante `funcion_restriccion`. Si no cumple, se le asigna un fitness de infinito, penalizándolo efectivamente. Para f_4, la propia función objetivo devuelve infinito si x_i está fuera del dominio estricto $2 < x_i < 10$, complementando `limites_generacion` que usa un épsilon para evitar los bordes exactos.

Esta estrategia combinada asegura que todas las soluciones evaluadas y propagadas respeten las restricciones de dominio de las variables. Para la función f_4, se definieron `limites_dominio_f4_generacion` que son ligeramente más estrictos $(2 + \epsilon, 10 - \epsilon)$ que los del enunciado para evitar problemas con la función ln durante la generación de individuos, mientras que la `funcion_restriccion` (`restriccion_f4`) verifica los límites del enunciado ($[-2{,}001, 10]$) y la propia función `f4` implementa el dominio funcional estricto $(2, 10)$.

3.3. Fijación de Semillas para Reproducibilidad

Dado que el algoritmo de Evolución Diferencial (DE), al igual que otras metaheurísticas, posee componentes estocásticos (e.g., en la inicialización de la población y en la selección de individuos para la mutación), los resultados pueden variar entre ejecuciones incluso con los mismos parámetros de configuración. Para asegurar la reproducibilidad de los experimentos y permitir una comparación justa entre las diferentes configuraciones, se fijaron las semillas de los generadores de números aleatorios para cada una de las 10 ejecuciones realizadas por función y configuración. Las semillas utilizadas fueron [101, 202, 303, 404, 505, 606, 707, 808, 909, 1010] para cada conjunto de 10 ejecuciones, tal como se indica en el archivo de resumen de estadísticas `resumen_estadisticas_DE_semillas.txt`.

4. Configuración Experimental

4.1. Funciones de Prueba

Para evaluar el rendimiento de DE, se utilizaron las siguientes cuatro funciones multimodales, definidas en dominios continuos:

- $f_1(x) = 4 - 4x_1^3 - 4x_1 + x_2^2$, con $-5 \leq x_i \leq 5$.
- $f_2(x) = \frac{1}{899}(\sum_{i=1}^{6} x_i^2 2^i - 1745)$, con $0 \leq x_i \leq 1$.
- $f_3(x) = (x_1^6 + x_2^4 - 17)^2 + (2x_1 + x_2 - 4)^2$, con $-500 \leq x_i \leq 500$.
- $f_4(x) = \sum_{i=1}^{10}[(\ln(x_i - 2))^2 + (\ln(10 - x_i))^2] - (\prod_{i=1}^{10} x_i)^{0,2}$, con $2{,}00001 \leq x_i \leq 9{,}99999$. Se ajustó el dominio original para asegurar que los argumentos de los logaritmos sean estrictamente positivos ($x_i - 2 > 0$ y $10 - x_i > 0$).

El objetivo en todos los casos es encontrar el mínimo de la función.

4.2. Configuraciones de Parámetros de DE

Se exploró el impacto de los parámetros clave de DE: tamaño de la población (`NP`), factor de mutación (`F`) y tasa de cruce (`CR`). Se definieron diez configuraciones distintas para observar su efecto en el rendimiento del algoritmo a través de las diferentes funciones. El número de generaciones (`Gens`) se mantuvo constante en 100 para todas las configuraciones y ejecuciones. La Table 1 presenta las diez configuraciones de parámetros utilizadas en la experimentación. Estas configuraciones buscan cubrir un espectro de combinaciones, desde aquellas que podrían favorecer una exploración más amplia (e.g., valores de F más altos, CR más bajos o poblaciones grandes) hasta aquellas orientadas a una explotación más fina (e.g., F más bajos, CR más altos o poblaciones más pequeñas). Esta variedad permite analizar la sensibilidad del algoritmo y la robustez de sus soluciones frente a distintas parametrizaciones.

Configuración	NP (Tamaño Población)	F (Factor Mutación)	CR (Tasa Cruce)
Config 1	50	0.7	0.8
Config 2	25	0.5	0.8
Config 3	75	0.7	0.95
Config 4	50	0.9	0.8
Config 5	50	0.7	0.5
Config 6	30	0.6	0.7
Config 7	60	0.8	0.9
Config 8	50	0.4	0.4
Config 9	50	1.0	1.0
Config 10	40	0.75	0.85

Tabla 1: Diez Configuraciones de Parámetros de DE Utilizadas (Gens=100 para todas)

4.3. Metodología de Ejecución

Para cada una de las cuatro funciones objetivo y para cada una de las diez configuraciones de DE detalladas en la Table 1, se realizaron 10 ejecuciones independientes del algoritmo. Esto permite obtener una medida de la robustez y consistencia de los resultados para cada par función-configuración. Cada ejecución consistió en 100 generaciones.

4.4. Métricas de Rendimiento Utilizadas

El rendimiento de cada configuración de DE para cada función se evaluó utilizando las siguientes métricas estadísticas, calculadas sobre los mejores valores de fitness obtenidos en las 10 ejecuciones:

- **Mejor Fitness (Mínimo):** El mejor valor de la función objetivo encontrado entre las 10 ejecuciones.
- **Peor Fitness (Máximo):** El peor valor (más alto, ya que se minimiza) de la función objetivo encontrado entre las 10 ejecuciones.
- **Promedio Fitness:** El promedio de los mejores valores de fitness de las 10 ejecuciones.
- **Mediana Fitness:** La mediana de los mejores valores de fitness de las 10 ejecuciones.
- **Desviación Estándar:** La desviación estándar de los mejores valores de fitness de las 10 ejecuciones, indicando la variabilidad o consistencia de los resultados.

Adicionalmente, se generaron gráficos de convergencia (mejor fitness vs. generación) para visualizar el progreso del algoritmo.

5. Resultados y Análisis

En esta sección se presentan los resultados obtenidos por las diez configuraciones de DE en las cuatro funciones de prueba. Para cada función, se muestra una tabla resumen con las métricas estadísticas y se analizan los hallazgos principales, incluyendo el ranking de configuraciones. Los datos completos para estas tablas se encuentran en el archivo `resumen_estadisticas_DE_semillas.txt`, disponible en el repositorio del proyecto. Se presentan además gráficos de convergencia selectos para ilustrar el comportamiento del algoritmo.

5.1. Función $f_1(x)$

Los resultados para la función $f_1(x) = 4 - 4x_1^3 - 4x_1 + x_2^2$ se resumen en la Table 2.

Config.	Parámetros (NP, F, CR)	Mejor	Peor	Promedio	Mediana	Desv. Est.
1	(50, 0.7, 0.8)	-5.160000e+02	-5.160000e+02	-5.160000e+02	-5.160000e+02	0.000000e+00
2	(25, 0.5, 0.8)	-5.160000e+02	-5.160000e+02	-5.160000e+02	-5.160000e+02	0.000000e+00
3	(75, 0.7, 0.95)	-5.160000e+02	-5.160000e+02	-5.160000e+02	-5.160000e+02	0.000000e+00
4	(50, 0.9, 0.8)	-5.160000e+02	-5.160000e+02	-5.160000e+02	-5.160000e+02	0.000000e+00
5	(50, 0.7, 0.5)	-5.160000e+02	-5.160000e+02	-5.160000e+02	-5.160000e+02	0.000000e+00
6	(30, 0.6, 0.7)	-5.160000e+02	-5.160000e+02	-5.160000e+02	-5.160000e+02	0.000000e+00
7	(60, 0.8, 0.9)	-5.160000e+02	-5.160000e+02	-5.160000e+02	-5.160000e+02	0.000000e+00
8	(50, 0.4, 0.4)	-5.160000e+02	-5.160000e+02	-5.160000e+02	-5.160000e+02	0.000000e+00
9	(50, 1.0, 1.0)	-5.160000e+02	-5.160000e+02	-5.160000e+02	-5.160000e+02	0.000000e+00
10	(40, 0.75, 0.85)	-5.160000e+02	-5.160000e+02	-5.160000e+02	-5.160000e+02	0.000000e+00

Tabla 2: Resultados Estadísticos para la Función f_1 (10 ejecuciones, 100 generaciones)

Para la función f_1, todas las 10 configuraciones de DE alcanzaron consistentemente el mismo valor de fitness de $-5{,}16 \times 10^2$ en todas las ejecuciones, como lo indica una desviación estándar de

cero. Esto sugiere que f_1 es una función relativamente fácil de optimizar para DE dentro del rango de parámetros probado. La Figure 2 muestra la convergencia para la Configuración 2 como ejemplo representativo.

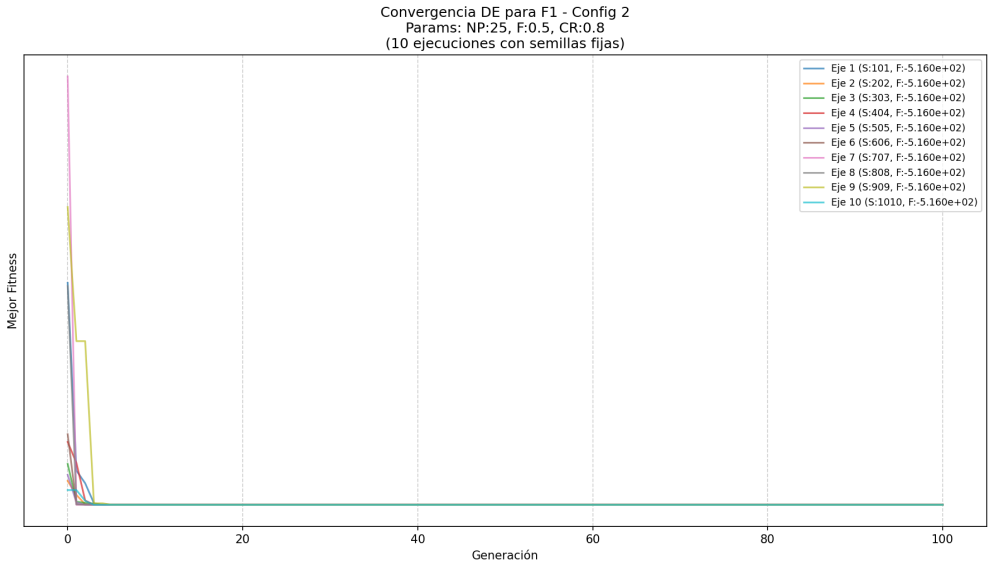


Figura 2: Convergencia de DE para F_1 - Configuración 2 (NP:25, F:0.5, CR:0.8).

5.2. Función $f_2(x)$

Los resultados para la función $f_2(x) = \frac{1}{899}(\sum_{i=1}^{6} x_i^2 2^i - 1745)$ se resumen en la Table 3.

Config.	Parámetros (NP, F, CR)	Mejor	Peor	Promedio	Mediana	Desv. Est.
1	(50, 0.7, 0.8)	-1.941046e+00	-1.941046e+00	-1.941046e+00	-1.941046e+00	0.000000e+00
2	(25, 0.5, 0.8)	-1.941046e+00	-1.941046e+00	-1.941046e+00	-1.941046e+00	9.420555e-16
3	(75, 0.7, 0.95)	-1.941046e+00	-1.941046e+00	-1.941046e+00	-1.941046e+00	0.000000e+00
4	(50, 0.9, 0.8)	-1.941046e+00	-1.941046e+00	-1.941046e+00	-1.941046e+00	0.000000e+00
5	(50, 0.7, 0.5)	-1.941046e+00	-1.941046e+00	-1.941046e+00	-1.941046e+00	0.000000e+00
6	(30, 0.6, 0.7)	-1.941046e+00	-1.941046e+00	-1.941046e+00	-1.941046e+00	0.000000e+00
7	(60, 0.8, 0.9)	-1.941046e+00	-1.941046e+00	-1.941046e+00	-1.941046e+00	0.000000e+00
8	(50, 0.4, 0.4)	-1.941046e+00	-1.941046e+00	-1.941046e+00	-1.941046e+00	0.000000e+00
9	(50, 1.0, 1.0)	-1.941046e+00	-1.941046e+00	-1.941046e+00	-1.941046e+00	0.000000e+00
10	(40, 0.75, 0.85)	-1.941046e+00	-1.941046e+00	-1.941046e+00	-1.941046e+00	0.000000e+00

Tabla 3: Resultados Estadísticos para la Función f_2 (10 ejecuciones, 100 generaciones)

Similar a f_1, la función f_2 también fue optimizada de manera muy consistente por todas las configuraciones de DE, alcanzando prácticamente el mismo valor de fitness en todas las ejecuciones, como se aprecia en la tabla (con una desviación estándar insignificante solo en una configuración). La Figure 3 ilustra la convergencia para la Configuración 2.

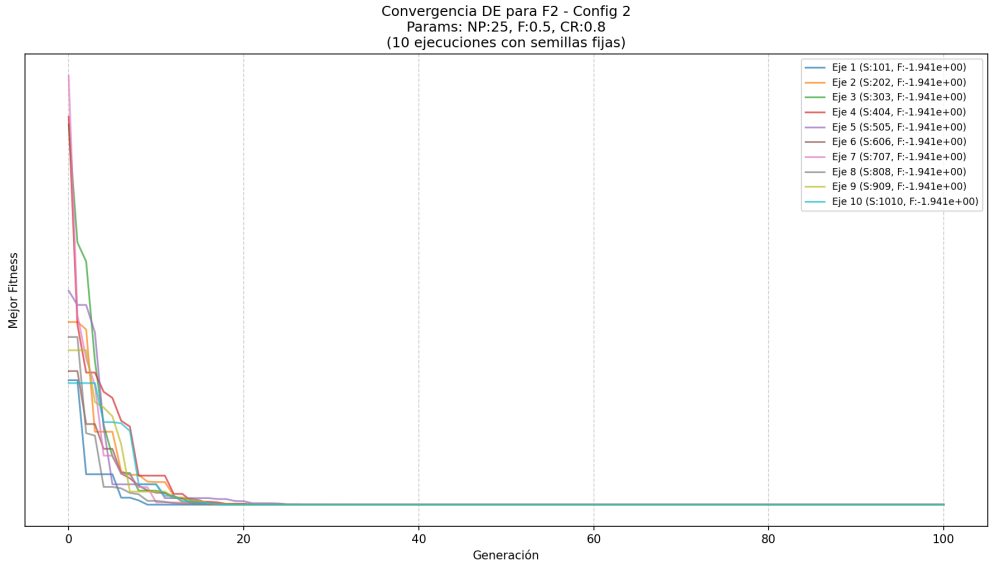


Figura 3: Convergencia de DE para F_2 - Configuración 2 (NP:25, F:0.5, CR:0.8).

5.3. Función $f_3(x)$

Los resultados para la función $f_3(x) = (x_1^6 + x_2^4 - 17)^2 + (2x_1 + x_2 - 4)^2$ se resumen en la Table 4.

Config.	Parámetros (NP, F, CR)	Mejor	Peor	Promedio	Mediana	Desv. Est.
1	(50, 0.7, 0.8)	1.43e-13	5.86e-06	6.18e-07	6.15e-10	1.75e-06
2	(25, 0.5, 0.8)	1.02e-19	2.82e-13	3.11e-14	9.04e-17	8.38e-14
3	(75, 0.7, 0.95)	2.16e-13	1.44e-06	2.93e-07	1.67e-10	5.65e-07
4	(50, 0.9, 0.8)	2.63e-07	4.84e-03	5.29e-04	2.65e-06	1.44e-03
5	(50, 0.7, 0.5)	3.18e-09	3.16e-04	5.25e-05	3.56e-06	1.01e-04
6	(30, 0.6, 0.7)	2.86e-13	2.26e-10	6.65e-11	4.25e-12	8.82e-11
7	(60, 0.8, 0.9)	1.58e-11	1.27e-08	2.94e-09	1.93e-09	3.76e-09
8	(50, 0.4, 0.4)	1.70e-12	4.49e-05	4.50e-06	1.38e-10	1.35e-05
9	(50, 1.0, 1.0)	1.94e-07	1.22e-04	2.02e-05	4.40e-06	3.55e-05
10	(40, 0.75, 0.85)	4.01e-12	3.26e-08	4.95e-09	1.72e-09	9.43e-09

Tabla 4: Resultados Estadísticos para la Función f_3 (10 ejecuciones, 100 generaciones)

A diferencia de f_1 y f_2, la función f_3 muestra una mayor sensibilidad a los parámetros. La Configuración 2 (NP:25, F:0.5, CR:0.8) obtuvo el mejor resultado. Su convergencia se muestra en la Figure 4.

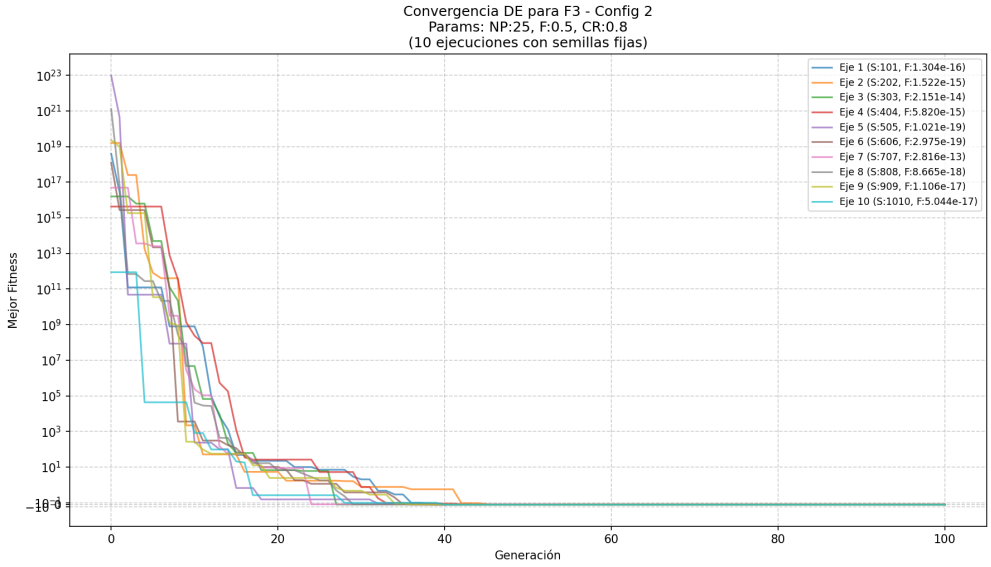


Figura 4: Convergencia de DE para F_3 - Configuración 2 (NP:25, F:0.5, CR:0.8).

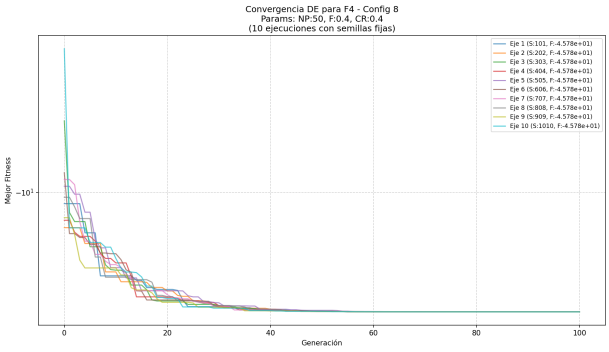
5.4. Función $f_4(x)$

Los resultados para la función $f_4(x) = \sum_{i=1}^{10}[(\ln(x_i - 2))^2 + (\ln(10 - x_i))^2] - (\prod_{i=1}^{10} x_i)^{0,2}$ se resumen en la Table 5.

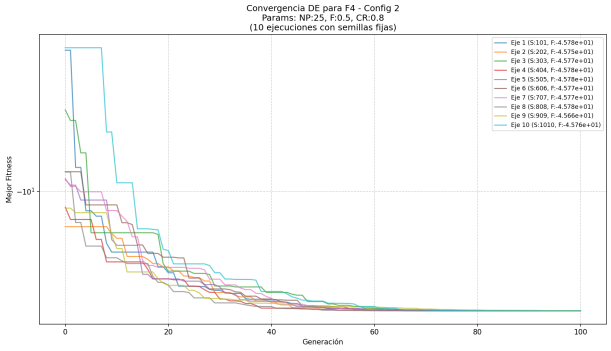
Config.	Parámetros (NP, F, CR)	Mejor	Peor	Promedio	Mediana	Desv. Est.
1	(50, 0.7, 0.8)	-4.485e+01	-4.013e+01	-4.294e+01	-4.301e+01	1.26e+00
2	(25, 0.5, 0.8)	-4.578e+01	-4.566e+01	-4.576e+01	-4.577e+01	3.38e-02
3	(75, 0.7, 0.95)	-4.116e+01	-3.664e+01	-3.900e+01	-3.924e+01	1.60e+00
4	(50, 0.9, 0.8)	-3.742e+01	-2.341e+01	-3.150e+01	-3.278e+01	4.20e+00
5	(50, 0.7, 0.5)	-4.556e+01	-4.543e+01	-4.546e+01	-4.545e+01	4.02e-02
6	(30, 0.6, 0.7)	-4.576e+01	-4.555e+01	-4.568e+01	-4.570e+01	6.81e-02
7	(60, 0.8, 0.9)	-3.755e+01	-2.807e+01	-3.272e+01	-3.280e+01	3.07e+00
8	(50, 0.4, 0.4)	-4.578e+01	-4.578e+01	-4.578e+01	-4.578e+01	1.14e-04
9	(50, 1.0, 1.0)	-3.078e+01	-1.379e+01	-2.117e+01	-2.153e+01	5.22e+00
10	(40, 0.75, 0.85)	-4.176e+01	-3.191e+01	-3.874e+01	-3.946e+01	2.57e+00

Tabla 5: Resultados Estadísticos para la Función f_4 (10 ejecuciones, 100 generaciones)

Para f_4, la Configuración 8 (NP:50, F:0.4, CR:0.4) y la Configuración 2 (NP:25, F:0.5, CR:0.8) fueron las más efectivas. La Figure 5 muestra la convergencia para estas dos configuraciones destacadas.



(a) Configuración 8 (NP:50, F:0.4, CR:0.4) - la mejor.



(b) Configuración 2 (NP:25, F:0.5, CR:0.8) - segunda mejor.

Figura 5: Convergencia de DE para F_4 con las dos configuraciones más destacadas.

5.5. Análisis del Desempeño y Discusión General

El análisis de los resultados a través de las cuatro funciones revela comportamientos interesantes del algoritmo DE y la influencia de sus parámetros.

5.5.1. Análisis Específico por Función y Robustez de DE

Como se observó en las Tables 2 and 3 y se ilustra en las Figures 2 and 3, las funciones f_1 y f_2 resultaron ser relativamente sencillas para el algoritmo DE implementado. Todas las 10 configuraciones probadas convergieron consistentemente al mismo valor óptimo (o prácticamente el mismo en el caso de f_2), evidenciado por desviaciones estándar nulas o insignificantes. Esto sugiere que f_1 y f_2 poseen paisajes de búsqueda donde el óptimo global es fácilmente accesible para DE, independientemente de las variaciones paramétricas dentro del rango estudiado. La convergencia es además muy rápida, alcanzándose el valor óptimo en las primeras 10-20 generaciones.

Para f_3 (Table 4), la situación es más compleja. La Configuración 2 (NP:25, F:0.5, CR:0.8) obtuvo el mejor resultado, alcanzando un valor mínimo de $1{,}02 \times 10^{-19}$. Su gráfico de convergencia (Figure 4) muestra una búsqueda efectiva, aunque con variabilidad entre las 10 ejecuciones, algunas convergiendo a valores ligeramente superiores (hasta $\sim 2{,}82 \times 10^{-13}$). Esto se refleja en una desviación estándar de $8{,}38 \times 10^{-14}$, pequeña en términos absolutos pero indicativa de la sensibilidad de la función. Esta configuración, con una población pequeña y valores moderados de F y CR, parece lograr un buen equilibrio: NP=25 podría permitir una exploración ágil sin excesiva redundancia, F=0.5 genera perturbaciones de magnitud media, y CR=0.8 asegura una alta probabilidad de incorporar estas nuevas características, facilitando el escape de óptimos locales.

Para f_4 (Table 5), la Configuración 8 (NP:50, F:0.4, CR:0.4) fue marginalmente superior en términos del mejor fitness encontrado ($-4{,}577837 \times 10^1$) y notablemente más robusta, con una desviación estándar de solo $1{,}14 \times 10^{-4}$. La Configuración 2 (NP:25, F:0.5, CR:0.8) también fue excelente, con un mejor fitness de $-4{,}577808 \times 10^1$ y una desviación estándar de $3{,}38 \times 10^{-2}$. Los gráficos de convergencia para ambas (Figures 5a and 5b) muestran que las 10 ejecuciones se

agrupan consistentemente hacia el final, especialmente para la Configuración 8, lo que visualiza su baja desviación estándar.

5.5.2. Impacto Detallado de los Parámetros (NP, F, CR) y Balance Exploración-Explotación

La sensibilidad de DE a sus parámetros se hizo evidente en las funciones f_3 y f_4.

- **Tamaño de Población (NP):** No se identificó un NP universalmente óptimo. Para f_3, una población pequeña (NP=25 en Configuración 2) fue la mejor, sugiriendo que para esta función de 2 dimensiones, una población ágil puede ser más eficiente. Para f_4 (10 dimensiones), la Configuración 8 con NP=50 fue superior, aunque la Configuración 2 con NP=25 también fue muy competitiva. Esto podría indicar que a medida que aumenta la dimensionalidad, un NP ligeramente mayor puede ser beneficioso para mantener la diversidad, pero no necesariamente poblaciones muy grandes (e.g., NP=75 en Configuración 3 no superó a las otras consistentemente).

- **Factor de Mutación (F):** Este parámetro controla la amplitud de la perturbación. Para f_3 y f_4, valores de F moderados a bajos (0.4 o 0.5) fueron parte de las configuraciones más exitosas (Config 2 para f_3; Config 2 y 8 para f_4). Un F alto, como F=0.9 en la Configuración 4 (NP:50, CR:0.8), resultó en un rendimiento inferior y una mayor variabilidad para f_4 (Mejor: $-3{,}74194 \times 10^1$, Desv. Est.: 4,20338) en comparación con la Configuración 1 (NP:50, F=0.7, CR=0.8) (Mejor: $-4{,}48543 \times 10^1$, Desv. Est.: 1,25871) o la Configuración 8 (NP:50, F:0.4, CR:0.4). La Figure 7 (Configuración 4 para f_4) muestra una convergencia más errática y dispersa entre sus 10 ejecuciones que la Figure 5a (Configuración 8). Esto sugiere que un F alto (mayor exploración) puede ser perjudicial si no permite al algoritmo explotar suficientemente las regiones prometedoras en funciones complejas.

- **Tasa de Cruce (CR):** CR determina la probabilidad de que las componentes del vector mutante pasen al vector de prueba. Valores de CR altos (e.g., 0.8) fueron comunes en configuraciones de buen rendimiento para f_3 y f_4 cuando se combinaban con F moderados (e.g., Configuración 2: F=0.5, CR=0.8). Un CR alto favorece la herencia de nuevas características del vector mutado, lo que puede ser clave para la exploración y el escape de óptimos locales. Sin embargo, la Configuración 8 para f_4 (F=0.4, CR=0.4) fue la más robusta, sugiriendo que una combinación de F bajo y CR bajo puede ser muy efectiva para una explotación fina y convergencia precisa en algunas funciones de alta dimensión. Esto indica una fuerte interacción entre F y CR: un F pequeño con un CR pequeño podría significar una búsqueda más local y conservadora, mientras que un F grande con un CR grande (como en la Configuración 9) representa una estrategia de cambio muy agresiva.

- **Configuraciones Extremas y Fallo en el Balance:** La Configuración 9 (NP:50, F=1.0, CR=1.0) fue consistentemente la de peor rendimiento en f_3 y f_4. Para f_3, su mejor valor fue

$1{,}94 \times 10^{-07}$. La Figure 6 muestra una convergencia muy lenta y hacia valores de fitness significativamente peores en comparación con la Figure 4. Las 10 ejecuciones para la Configuración 9 en f_3 son muy dispersas y se estancan prematuramente. Esto indica que F=1.0 (máxima amplitud de mutación) y CR=1.0 (el vector de prueba es idéntico al vector mutante) destruyen demasiada información de la solución actual y fuerzan una exploración excesivamente agresiva, impidiendo que el algoritmo converja a óptimos de alta calidad. Este es un claro ejemplo de un desbalance hacia la exploración en vez de la explotación.

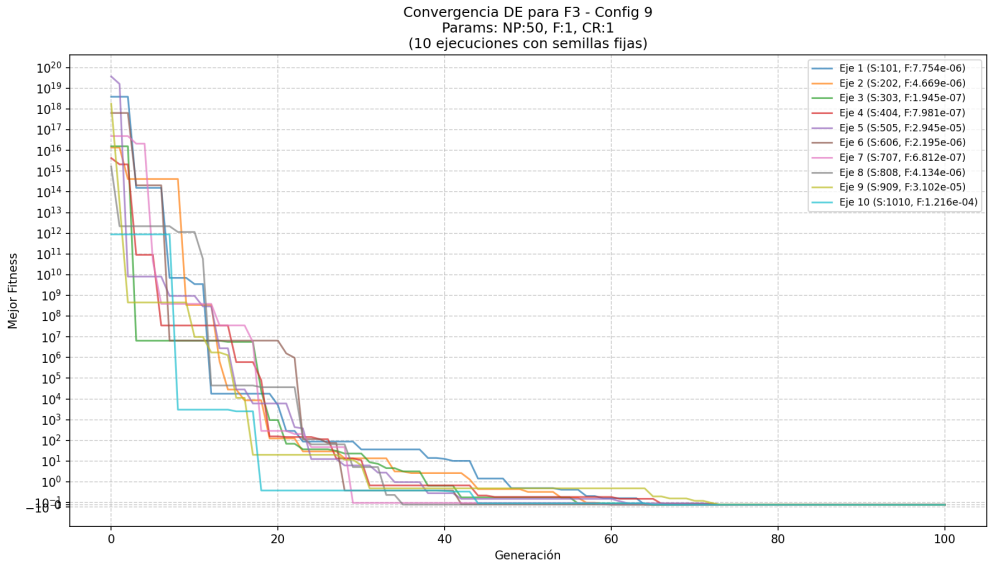


Figura 6: Convergencia de DE para F_3 - Configuración 9 (NP:50, F:1.0, CR:1.0), ilustrando un rendimiento subóptimo.

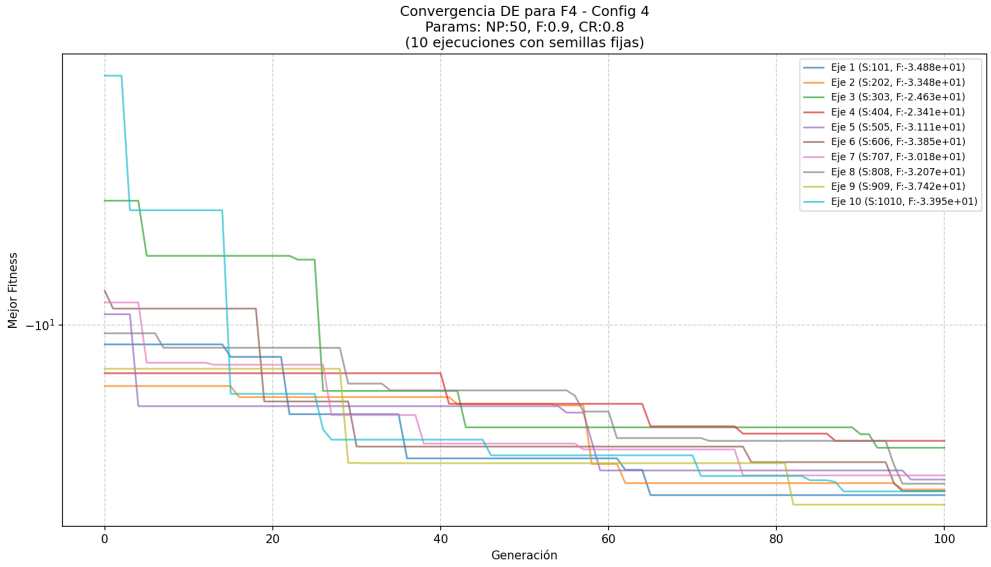


Figura 7: Convergencia de DE para F_4 - Configuración 4 (NP:50, F:0.9, CR:0.8), mostrando mayor variabilidad con F alto.

5.5.3. Comportamiento General de DE frente a la Dificultad de las Funciones

DE demostró ser un optimizador muy robusto y eficiente para las funciones f_1 y f_2, encontrando el óptimo consistentemente con una rápida convergencia, independientemente de la configuración de parámetros dentro de las probadas. Estas funciones, de baja dimensionalidad (2 y 6 respectivamente) y aparentemente con paisajes de búsqueda menos complejos o con óptimos globales muy dominantes, no exigieron un ajuste fino de los parámetros de DE.

Para f_3 (2 dimensiones pero con términos polinomiales de alto grado que pueden generar múltiples óptimos locales) y f_4 (10 dimensiones con términos logarítmicos y producto, indicando un paisaje potencialmente complejo y sensible cerca de los límites del dominio), la elección de parámetros fue crucial. La capacidad de DE para encontrar valores muy cercanos a cero para f_3 (cuyo óptimo conocido es 0) y valores fuertemente negativos para f_4 subraya su potencial para navegar estos paisajes multimodales. La mayor variabilidad (desviación estándar) observada en f_3 y f_4 para ciertas configuraciones (especialmente aquellas con F alto o F y CR extremos) resalta la importancia del equilibrio exploración-explotación. El éxito de configuraciones con F y CR moderados o bajos para estas funciones sugiere que una vez que se identifican regiones prometedoras, una búsqueda más conservadora es beneficiosa para la explotación fina y la convergencia precisa.

6. Conclusiones

El estudio realizado permitió evaluar la efectividad del algoritmo de Evolución Diferencial (DE) para la optimización de cuatro funciones multimodales. Se implementó DE y se experimentó con diez configuraciones de parámetros distintas, realizando diez ejecuciones por cada combinación de

función y configuración.

Los principales hallazgos son:

- DE demostró ser altamente eficaz y robusto para las funciones f_1 y f_2, donde todas las configuraciones probadas convergieron consistentemente al mismo valor óptimo.
- Para las funciones f_3 y f_4, consideradas más complejas, la elección de los parámetros de DE (NP, F, CR) tuvo un impacto significativo en el rendimiento.
  - Para f_3, la Configuración 2 (NP:25, F:0.5, CR:0.8) obtuvo el mejor resultado.
  - Para f_4, la Configuración 8 (NP:50, F:0.4, CR:0.4) fue la más destacada.
- Configuraciones con valores extremos para F y CR (e.g., F=1.0, CR=1.0) resultaron ser perjudiciales para el rendimiento en las funciones más complejas, probablemente debido a una exploración excesiva que dificulta la explotación.
- El algoritmo DE es una herramienta potente para la optimización de funciones multimodales, pero su rendimiento puede depender de una adecuada sintonización de sus parámetros, especialmente frente a paisajes de búsqueda complejos.

Como trabajo futuro, se podría explorar un rango más amplio de parámetros o investigar estrategias adaptativas para F y CR. No obstante, el presente estudio cumple con el objetivo de implementar DE y analizar su comportamiento en funciones multimodales, evidenciando la influencia de su parametrización.

7. Anexos

En esta sección se presentan los gráficos de convergencia de las metaheurísticas Ant Colony Optimization (ACO), Bat Algorithm (BA), Genetic Algorithm (GA) y Particle Swarm Optimization (PSO) para la función f_3, utilizados en la fase preliminar de selección de la metaheurística a implementar.

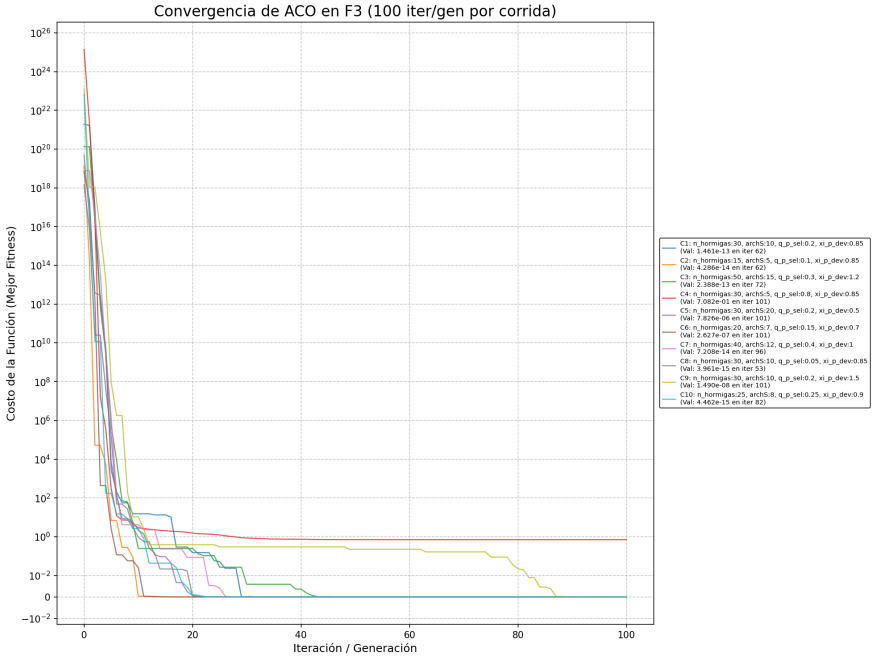


Figura 8: Convergencia de ACO en la función f_3 con 10 configuraciones distintas (Anexo).

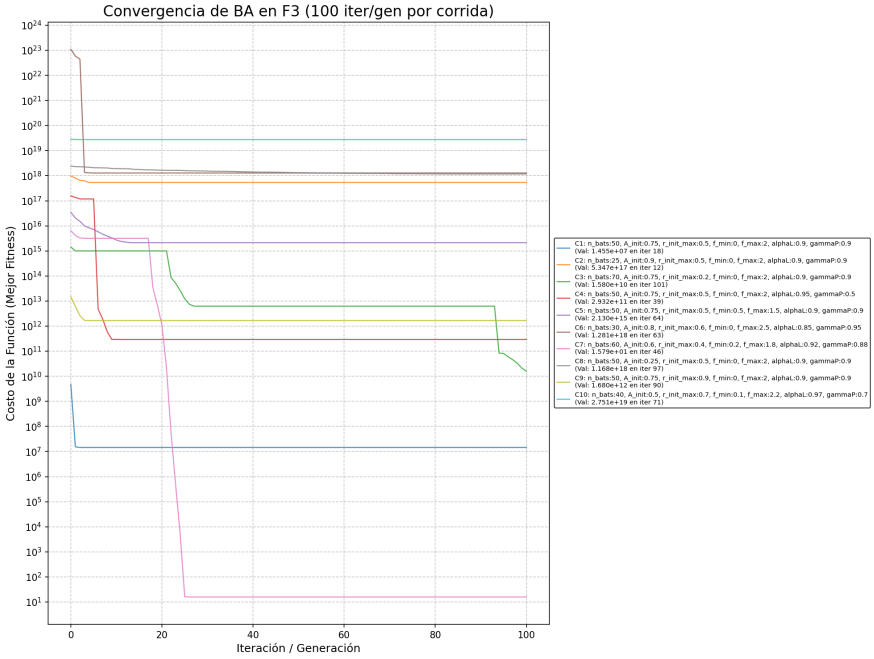


Figura 9: Convergencia de BA en la función f_3 con 10 configuraciones distintas (Anexo).

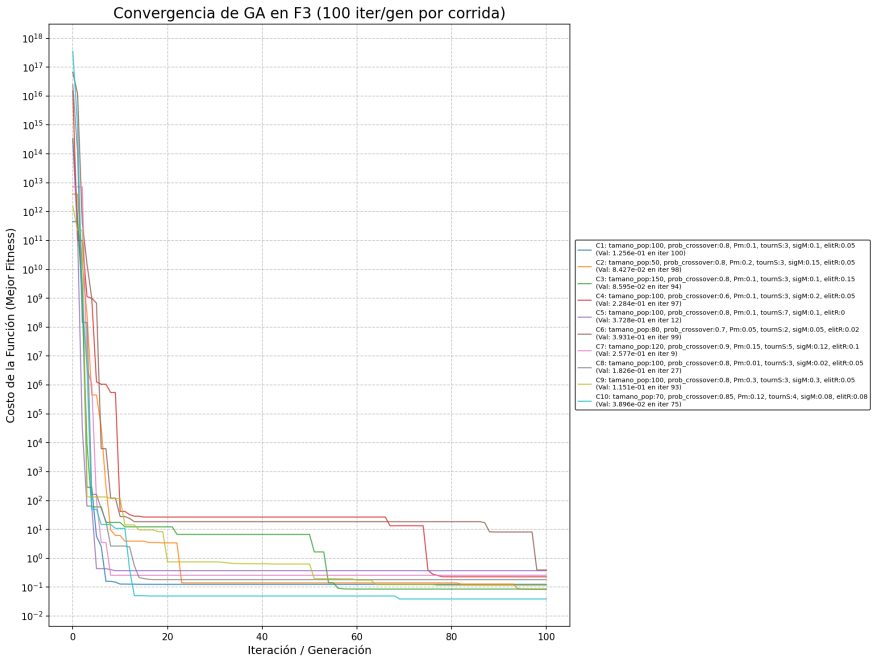


Figura 10: Convergencia de GA en la función f_3 con 10 configuraciones distintas (Anexo).

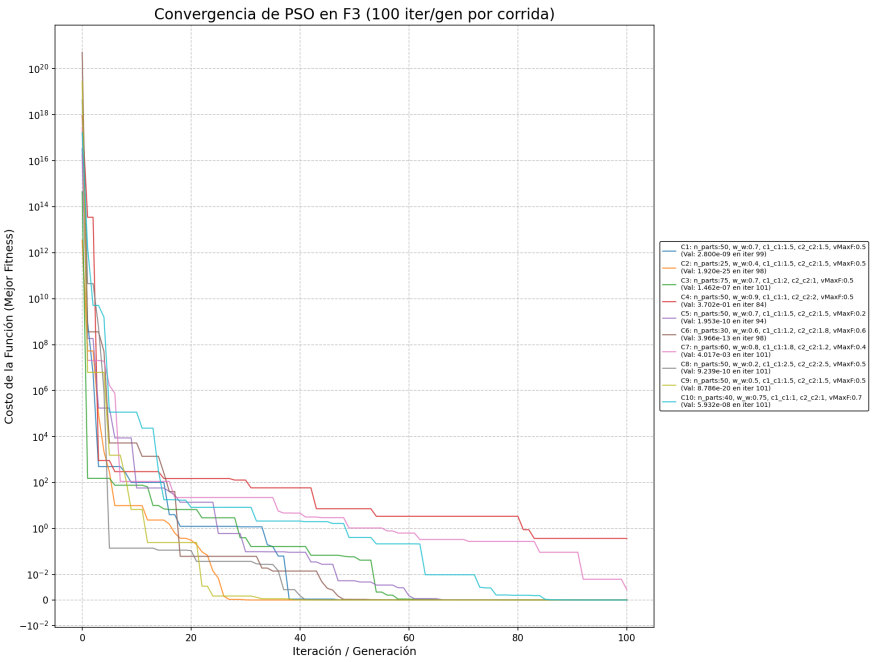


Figura 11: Convergencia de PSO en la función f_3 con 10 configuraciones distintas (Anexo).