

## EVALUACIÓN BLOQUE III: ALGORITMOS EVOLUTIVOS

### Parte I: Problemas Multiobjetivo sin restricciones

RAMOS GONZÁLEZ, VÍCTOR ([vicramgon@alum.us.es](mailto:vicramgon@alum.us.es))

Fecha de elaboración: *30 de marzo de 2021*

Profesor: *Dr. Francisco V. Fernández Fernández*

Asignatura: *Aplicaciones de Soft-Computing.*

Departamento: *Electrónica y Electromagnetismo.*

*GII. Tecnologías Informáticas - Universidad de Sevilla.*

---

### Contenidos

1. MOEA/D. Introducción	1
2. MOEA/D. Metodología	4
3. MOEA/D. Implementación	8
4. MOEA/D. Experimentación y resultados	16
5. Conclusion final	42
6. CMOEA/D. Introducción	48
7. CMOEA/D. Metodología	48
8. CMOEA/. Implementación	48
9. CMOEA/D. Experimentación y resultados	48
10. Conclusiones	48

---

### 1. MOEA/D. Introducción

Se considera el problema multi-objetivo (sin pérdida de generalidad) consistente en la minimización de  $m$  funciones objetivo:

$$\text{minimize } F(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_m(\mathbf{x}))$$



de manera que el espacio de búsqueda  $\mathbf{x} \in \Omega$  es acotado y corresponde a  $\Omega = X_1 \times \cdots \times X_p$ , con  $X_i$  el conjunto continuo de valores posibles para la componente  $x_i$  acotado superior e inferiormente por  $x_{Li}$  y  $x_{Ui}$  respectivamente.

En los problemas de optimización multiobjetivo, es frecuente que los objetivos sean contrapuestos, esto es, la mejora en uno de los objetivos implica la desmejora de otro, de forma que no existe un único punto que optimice todos los objetivos, si no que se tiene lo que se conoce como frente, en concreto se habla del frente de Pareto o frente Pareto-óptimo.

El frente de Pareto corresponde al conjunto de puntos en el espacio de objetivos, tal que ninguno de ellos es dominado por ningún otro de punto de dicho espacio. Dado que un punto  $\mathbf{u}$  es dominado por otro  $\mathbf{v}$  si y sólo si  $\forall i \in \{1, \dots, m\} : u_i \geq v_i$  y además  $\exists j \in \{1, \dots, m\} : u_j > v_j$ , esto es  $\mathbf{u}$  es mejor que  $\mathbf{v}$  en al menos un objetivo y no es peor en ninguno de los restantes. De forma que el objetivo corresponde, precisamente, a la obtención de este conjunto de puntos. Notaremos la dominancia Pareto como  $v \preceq u$ .

En el estado del arte, existen multitud de aproximaciones a esta categoría de problemas (*MOP*), en este trabajo, se propone abordarla desde la filosofía de los algoritmos evolutivos basados en agregación (*MOEA/D*), que basan su funcionamiento en la descomposición del problema de optimización multi-objetivo en varios subproblemas con un único objetivo (mono-objetivo), de manera que la función objetivo de cada uno de los subproblemas es definida por agregación ponderada de las funciones del problema multi-objetivo. Dentro de este campo existe una amplia variedad de subfilosofías y mejoras al algoritmo original (entre ellas *MOEA/D-DE* [1] o *MOEA/D-DRA* [2]) pero nosotros nos referiremos generalmente a las mismas como *MOEA/D*.

Para generar los distintos problemas mono-objetivo existen distintas formulaciones (como *Weighted Sum* o *Boundary Intersections* [3]), en este trabajo se seguirá la formulación más corriente en la literatura dada por Tchebychef como:

$$g^{te}(\mathbf{x}|\boldsymbol{\lambda}, \mathbf{z}^*) = \max_{1 \leq i \leq m} \{ \lambda_i |f_i(\mathbf{x}) - z_i^*| \}$$

donde  $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_m)$  corresponde a un vector de pesos y  $\mathbf{z}^* = (z_1^*, \dots, z_m^*)$  corresponde al vector cuyas componentes corresponden a los valores óptimos (mínimos) de las funciones objetivo consideradas de forma aislada, esto es:

$$z_i^* = \min \{ f_i(\mathbf{x}) \mid \mathbf{x} \in \Omega \}$$

De forma que, para cada punto  $\mathbf{x}^*$  del frente de Pareto existe un vector de pesos ( $\boldsymbol{\lambda}$ ) tal que  $\mathbf{x}^*$  es solución óptima del subproblema asociado a ( $\boldsymbol{\lambda}$ ), de manera que recorriendo completamente los posibles  $\boldsymbol{\lambda}$  se obtendrían todos los puntos del frente Pareto-óptimo. Computacionalmente, nos quedaremos con una muestra de todos los posibles vectores de pesos  $\boldsymbol{\lambda}$ , de forma que trataremos de obtener una aproximación al frente de Pareto completo.

La filosofía *MOEA/D* se enmarca dentro de los conocidos como *Algoritmos Evolutivos* y como tal uno de los pasos fundamentales del funcionamiento del mismo se basa en la mutación y recombinación de los individuos de la población. En este aspecto la literatura está plagada de distintos operadores, dentro de los cuales los basados en *Evolución Diferencial* son claramente los más numerosos. En este trabajo compararemos tres operadores (*EOP1*, *EOP2*, *EOP3*), sobre un mismo marco de desarrollo, mostrando



tanto la metodología seguida como la implementación realizada y los resultados obtenidos, evaluando, mediante el uso de distintas métricas características en la literatura (*Hypervolume*, *Spacing*, *Cober Set*), la eficacia del algoritmo presentado (y los distintos operadores) en relación a otros presentados en este mismo ámbito (*NSGA-II*).

## 2. MOEA/D. Metodología

En esta sección explicaremos la metodología propuesta, en la que detallaremos los pasos del marco de desarrollo del algoritmo, así como los distintos operadores evolutivos.

### 2.1. Funcionamiento general del algoritmo

La figura 1 muestra el diagrama de flujo general del algoritmo, ajustándose a un esquema típico de algoritmo evolutivo. De manera superficial, el algoritmo recibe las funciones objetivo, el espacio de búsqueda y una serie de parámetros que intervienen en el algoritmo (detallados más adelante), lleva a cabo una primera etapa de inicialización de la población, los vectores de pesos y vecindad de los subproblemas, así como el punto de referencia . A partir de ahí se lleva a cabo un proceso iterativo, en el que se va actualizando la población así como el punto de referencia. De forma que al fin del proceso la población constituye la aproximación del algoritmo al frente de Pareto para el problema tratado.

### 2.2. Datos de entrada

El algoritmo recibe como datos de entrada:

- **Funciones objetivo:** Un conjunto ordenado de funciones  $f_1, \dots, f_m$  consideradas como las funciones objetivo a optimizar por el algoritmo, tratando todas ellas como caso de minimización.
- **El espacio de búsqueda  $\Omega$ :** Dado por el producto cartesiano de los espacios de búsqueda de cada una de las variables, acotados superior e inferiormente por  $x_{Lj}$  y  $x_{Uj}$  respectivamente.
- $N$ : Corresponde al número de subproblemas considerados para la división del objetivo múltiple en subobjetivos individuales (por agregación).
- $G$ : Corresponde al número de generaciones máximo que realizará el algoritmo, estableciendo un criterio de parada para el mismo.

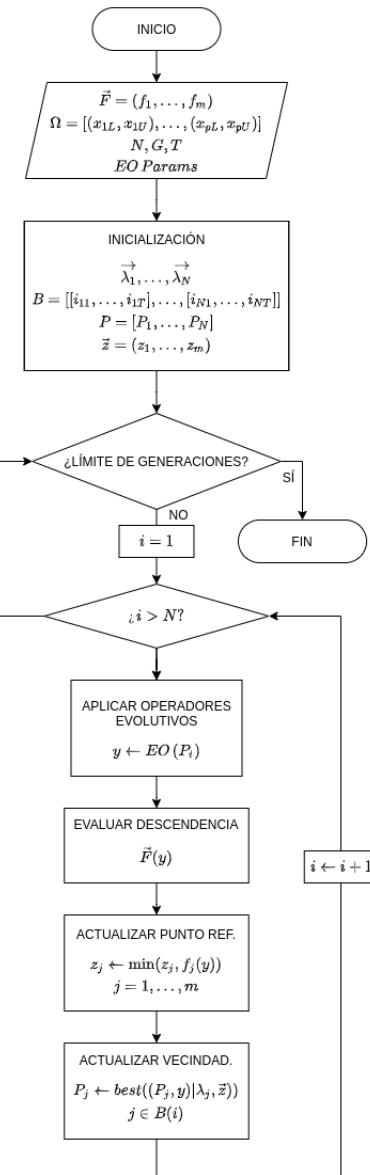


Figura 1: Flujo general del algoritmo

- $T$ : Corresponde al número de vecinos de cada subproblema esto es, el número de subproblemas, incluyéndose a sí mismo, que serán considerados parte de la vecindad de cada uno de los subproblemas y que influirán en el proceso evolutivo de la población.
- **Operador evolutivo:** Durante el desarrollo del algoritmo son utilizados operadores de cruce y mutación que han de ser dados por el usuario y que obtendrán un solo individuo como descendiente de cada uno de los individuos de la población.
- **Otros parámetros**, que detallamos y justificamos en el apartado correspondiente.

### 2.3. Inicialización del algoritmo

Para poder llevar a cabo la ejecución del algoritmo (proceso iterativo) hemos de disponer de algunos elementos que debemos inicializar al comienzo del algoritmo como son:

- **El conjunto de vectores de pesos**  $\lambda_1, \dots, \lambda_N$ , vectores  $m$ -dimensionales que identifican a cada subproblema, de forma que para todos ellos debe cumplirse  $\|\lambda_i\|_1 = 1$  y además deben estar distribuidos uniformemente (equiespaciados, tomando como distancia la forma euclídea). Para el caso bi-objetivo, dado que los vectores han de estar normalizados, los  $N$  vectores pueden inicializarse siguiendo la expresión  $\lambda_i = \left( \frac{i-1}{N-1}, \frac{N-i}{N-1} \right)$  con  $i \in \{1, \dots, N\}$ .
- **Vecindad  $B(i)$  de cada vector peso  $\lambda_i$** , que corresponde al conjunto de los los  $T$  vectores más cercanos a  $\lambda_i$  de entre todos los vectores de pesos (incluyendo al propio  $\lambda_i$ ). Nótese que  $B(i)$  será considerado el entorno del subproblema  $i$ , de manera que debe considerarse un entorno cercano ( $T$  pequeño) para la evolución del individuo asociado al subproblema considerado.
- **Inicialización de la población  $P$** , con  $N$  individuos cada uno de ellos asociados a uno de los subproblemas. Para la inicialización se generará una población aleatoria (respetando los espacios de búsqueda para cada variable).
- **Inicialización del punto de referencia  $z$** . Recuérdese que según la formulación de Tchebycheff  $z^*$  debe contener los óptimos globales de cada una de las funciones objetivo en el espacio de búsqueda, pero esto implica resolver  $m$  problemas de optimización mono-objetivo. En vez de eso relajaremos  $z^*$  en  $z$  de forma que contendrá en cada momento el mejor (menor) valor para cada  $f_j$  ( $j = 1, \dots, m$ ) alcanzado por cualquiera de los individuos evaluados hasta ese instante. De forma que dicho punto de referencia  $z$  se irá actualizando de forma síncrona al avance del algoritmo.
- **Inicialización del conjunto de soluciones no dominadas**. Opcionalmente, el algoritmo puede mantener un conjunto con las soluciones no dominadas por ninguna otra encontrada hasta el momento de forma que el conjunto será actualizado a lo largo del proceso evolutivo.

## 2.4. Desarrollo iterativo

Tras la inicialización, comienza un proceso iterativo (evolutivo) en el que la población, se actualiza por medio del uso de operadores evolutivos que presentaremos a continuación. Como criterio de parada para este proceso iterativo se tomará el límite de generaciones fijado por el usuario (al igual que en otros algoritmos de carácter evolutivo o inteligencia colectiva se pueden fijar otros criterios de parada anticipada). De manera que para cada iteración (generación) se llevan a cabo las siguientes operaciones:

1. *Reproducción*: Para cada uno de los individuos se aplican operadores evolutivos (cruce y mutación) de forma que cada individuo es perturbado (en mayor o menor medida) por el entorno. Como entorno se considera el vecindario  $B(i)$ , del que son tomados, con cierta aleatoriedad, (según el operador) una serie de individuos con los que se aplican los operadores evolutivos. Como el vecindario son los considerados problemas cercanos, es esperable que los cambios entre ellos van a ser pequeños por lo que los individuos tenderán a ser parecidos.

En concreto se van a implementar y realizar pruebas utilizando los siguientes operadores evolutivos:

- **EOP1** Basado en [4] y en el propuesto en el enunciado. Se propone el uso de DE (*Differential Evolution*) + GP (*Gaussian Perturbation*), de manera que, en primer lugar para el individuo  $P_i$  se genera el vector mutante  $\hat{y}$ , con el uso de cinco vectores ‘noisy’, mediante la expresión:

$$\hat{y} = \mathbf{x}^{(r1)} + F \cdot (\mathbf{x}^{(r2)} - \mathbf{x}^{(r3)}) + rand \cdot F \cdot (\mathbf{x}^{(r4)} - \mathbf{x}^{(r5)})$$

donde  $F$  es elegido aleatoriamente de un pool de valores (entre 0 Y 1) y  $rand$  corresponde a un número aleatorio (entre 0 y 1). Tras esto, se realiza una recombinación utilizando el cruce típico de evolución diferencial, tal que cada componente del vector  $y$  se elige aleatoriamente entre  $\hat{y}$  y  $P_i$ , de forma que  $y$  contenga al menos una componente del vector mutante. Para ello se sigue la expresión:

$$y_j = \begin{cases} \hat{y}_{ij} & \text{si } rand \leq CR \text{ o } j = \delta \\ P_{ij} & \text{e.o.c.} \end{cases}$$

donde  $CR$  corresponde a un parámetro ajustable (entre 0 y 1) que representa la probabilidad de cruce,  $\delta$  corresponde a una de las componentes del descendiente elegida aleatoriamente y  $rand$  corresponde a un número aleatorio (entre 0 y 1).

Finalmente cada componente de  $y$  es perturbada con probabilidad  $p_m$  (parámetro ajustable entre 0 y 1, típicamente establecida en la literatura como  $1/p$ , con  $p$  la dimensionalidad del espacio de búsqueda) mediante una distribución gaussiana ( $N(0, \sigma_j = \frac{x_{Uj} - x_{Lj}}{SIG})$  con  $SIG$  un parámetro ajustable). Si alguna de las componentes quedara fuera del espacio de búsqueda correspondiente su valor es establecido al valor de la cota correspondiente.

- **EOP2**, basado en [1] y [4], se propone una aproximación con el uso de DE + SBX, de forma que se genera el vector mutante con 5 vectores ‘noisy’ siguiendo la expresión:

$$\hat{y} = \mathbf{x}^{(r1)} + F \cdot (\mathbf{x}^{(r2)} - \mathbf{x}^{(r3)}) + rand \cdot (\mathbf{x}^{(r4)} - \mathbf{x}^{(r5)})$$

donde  $F$  corresponde a un parámetro de ajuste (fijo) y  $rand$  corresponde a un número aleatorio (en  $[0, 1]$ ). Una vez ejecutada dicha mutación, el operador de cruce corresponde al mismo



que EOP1. El vector resultante  $\mathbf{y}$  es finalmente perturbado en cada componente utilizando la expresión (proveniente de SBX) siguiendo la expresión :

$$y_j = \begin{cases} y_j + \sigma_k \cdot (x_{Uj} - y_{Lj}) & \text{si } rand < p_m \\ y_j & \text{e.o.c} \end{cases}, \quad \sigma_k = \begin{cases} (2 \cdot rand)^{\frac{1}{\eta+1}} - 1 & \text{si } rand < 0,5 \\ 1 - (2 - 2 \cdot rand)^{\frac{1}{\eta+1}} & \text{e.o.c} \end{cases}$$

donde  $rand$  corresponde a un número aleatorio (entre 0 y 1), distinto en cada expresión (izqda. y dcha.),  $\eta$  corresponde al índice de distribución de cruce<sup>1</sup> y  $p_m$  la probabilidad de mutación. Finalmente, si alguna componente de  $\mathbf{y}$  se encuentra fuera de las cotas establecidas para la variable, es ajustada a la cota correspondiente.

- **EOP3**, basado en el esquema de Recombinación Uniforme (*Uniform Crossing (UX)*) se propone una aproximación con el uso de UX + GP, de forma que para el individuo  $P_i$  se genera descendiente  $\mathbf{y}$ , con  $\varepsilon \leq T$  vectores tomados aleatoriamente del vecindario, de manera que cada componente del vector mutante es elegida aleatoriamente entre los vectores del entorno:

$$y_j = x_j^{(ri)}$$

donde  $ri \in B(i)$ . Sobre el descendiente se lleva a cabo una mutación (perturbación) gaussiana análoga a la planteada en EOP1. Finalmente, si alguna componente de  $\mathbf{y}$  se encuentra fuera de las cotas establecidas para la variable, es ajustada a la cota correspondiente.

2. *Evaluación*: El descendiente obtenido es evaluado respecto a todos los objetivos  $\mathbf{F}(\mathbf{y})$ . Nótese que hemos exigido que el operador evolutivo considerado genere un único descendiente. Aunque en la literatura existen versiones de *MOEA/D* que trabajan con varios descendientes, es muy común encontrar la versión adoptada, con la ventaja de que para cada individuo y cada generación se realiza una única evaluación.
3. *Actualización del punto de referencia  $z$* : Recuérdese que  $z$  correspondía al vector  $m$ -dimensional ( $m$  el número de objetivos) cuyas componentes correspondientes correspondían a los pseudo-óptimos (mejores valores obtenidos hasta el momento) luego hemos de actualizar  $z$  en aquellas componentes tal que  $F_j(\mathbf{y}) < z_j$  (con  $j = 1, \dots, m$ ).
4. *Actualización del entorno  $B(i)$* : Recuérdese que  $z$  correspondía al vector  $m$ -dimensional ( $m$  el número de objetivos) cuyas componentes correspondientes correspondían a los pseudo-óptimos (mejores valores obtenidos hasta el momento) luego hemos de actualizar  $z$  en aquellas componentes tal que  $F_j(\mathbf{y}) < z_j$  (con  $j = 1, \dots, m$ ).
5. **Actualización del conjunto de soluciones no dominadas**. En caso de que se utilice, se ha de actualizar el conjunto de soluciones no dominadas. De forma que si y sólo si ninguna de las soluciones del conjunto dominan a la valoración del descendiente  $\mathbf{F}(\mathbf{y})$  entonces este pasa a formar parte de las soluciones no dominadas y son eliminadas del conjunto aquellas que son dominadas por  $\mathbf{F}(\mathbf{y})$ . En otro caso no se realiza actualización alguna.

<sup>1</sup>El valor numérico de este parámetro de control es inversamente proporcional a la cantidad de perturbación en las variables de diseño. Cuanto menor sea el valor de este parámetro de control, mayor será la perturbación y viceversa. Por tanto, un valor más pequeño mejora la resistencia a la convergencia prematura a costa de una búsqueda muy centrada.

#### 2.4.1. Algunas consideraciones

Dado el proceso iterativo propuesto, aunque no ha sido especificado, el proceso de actualización de la población (y sus valoraciones) se realiza directamente sobre la población, y no se espera para ser efectiva al fin de la generación. Por ello es conveniente que a la hora de llevar a cabo las actualizaciones los elementos se recorran en un orden aleatorio, para tratar de evitar descompensaciones entre los individuos.

Además hemos de tener en cuenta que el hecho de realizar la sustitución en los vecinos puede conllevar una alta disminución de la diversidad y llevar al proceso a una convergencia prematura. Por ello, opcionalmente, se puede establecer un parámetro *UN* (*updatings number*) que limite el número de vecinos sustituidos [5].

#### 2.5. Final del algoritmo

El proceso iterativo previo se realiza hasta alcanzar el criterio de parada, dado por el número máximo de generaciones  $G$ . De forma que al final del proceso en la última población se tienen justamente los mejores individuos encontrados durante el proceso para cada uno de los subproblemas y cuyas valoraciones constituyen, precisamente, la aproximación al frente de Pareto.

En el caso de hacer uso del conjunto de soluciones no dominadas  $NSD$  éste sería una aproximación más cercana y más amplia al frente real Pareto-óptimo, dado que en él se encontrarán todas aquellas soluciones no dominadas por ninguna otra de las encontradas durante todo el proceso de búsqueda.

### 3. MOEA/D. Implementación

En este apartado describiremos la implementación concreta realizada de la metodología previamente propuesta, concretando cada una de las etapas y procesos. En el *algoritmo 1* se presenta el pseudocódigo del marco general de desarrollo, del algoritmo. En los puntos esta sección concretaremos cada una de las partes del mismo, dando los detalles de implementación correspondientes.

#### 3.1. Datos de entrada

Como se propone en la metodología los datos de entrada corresponden a:

- **Funciones objetivo:** Las funciones objetivo  $f_1, \dots, f_m$  consideradas son dadas como una lista de longitud  $m$  de funciones, que reciban como entrada un individuo (vector  $p - dimensional$ ) y devuelvan una lista de valores reales correspondientes a la valoración del individuo respecto de la correspondiente función.

- **El espacio de búsqueda  $\Omega$ :** Que establece para cada una de las variables los límites de variación (continua) de la variable. De forma que dichos límites se darán como una lista de longitud  $p$  de pares  $(x_{Lj}, x_{Uj})$  de valores reales. Dichos pares son ordenados para asegurar que  $x_{Lj} \leq x_{Uj}$ .
- **Número de subproblemas  $N$ :** Un entero que corresponde al número de subproblemas considerados, esto es el número de vectores de pesos  $\lambda_i$  y el tamaño considerado para la población.
- **Número de generaciones  $G$ :** Un entero que corresponde al número de máximo de iteraciones llevadas a cabo por el algoritmo.
- **Número de vecinos  $T$ :** Un entero correspondiente al número de subproblemas, incluyéndose a sí mismo, que serán considerados parte de la vecindad.

### 3.2. Inicialización

Durante la fase de inicialización se establecen los valores iniciales para:

- **Vectores de pesos  $(\lambda_1, \dots, \lambda_N)$ ,** de forma que si el problema es bi-objetivo dichos vectores son generados utilizando la expresión  $\lambda_i = \left( \frac{i-1}{N-1}, \frac{N-i}{N-1} \right)$  con  $i \in \{1, \dots, N\}$  para  $i = 1, \dots, N$ . Si se desea es posible especificar un fichero en formato *ASCII* del que se leerán las  $N$  primeras líneas que deberán contener los vectores de pesos considerados.
- **Vecindad de los subproblemas  $B$ .** Corresponde a una lista de listas que contienen los índices considerados dentro de la vecindad de un subproblema, de forma que para cada subproblema  $i$  son considerados como vecinos los  $T$  cuyos vectores asociados son más cercanos a  $\lambda_i$ . Para ello se crea (temporalmente) un matriz de distancias  $D_{N \times N}$  tal que cuya posición  $i, j$  corresponde a la distancia euclídea entre  $\lambda_i$  y  $\lambda_j$ . Una vez calculada dicha matriz, el vecindario de  $i$  corresponde a los  $T$   $j$  con menor  $d_{ij}$ .

---

**Algoritmo 1:** Marco gral. del MOEA/D

---

**Entrada:**

- MOP:

$$\mathbf{F} = [f_1, \dots, f_m]$$

$$\Omega = [(x_{L1}, x_{U1}), \dots, (x_{Lp}, x_{Up})]$$

- $N$ : número de subproblemas
- $G$ : número de generaciones
- $T$ : número de vecinos
- $EOP$ : operador evolutivo
- $UN$ : Updations number
- $NDS$ : Usar/no usar NDS.

**Inicialización:**

- $A$ . Vectores de pesos  $(\lambda_1, \dots, \lambda_N)$
- $B$ . Vecindad de los subproblemas  $[[B_{1,1}, \dots, B_{1,T}], \dots, [B_{N,1}, \dots, B_{N,T}]]$
- $P$ . Población  $(I_1, \dots, I_N)$
- VP. Valoración de la población  $(F(P))$
- $z$ . Punto de referencia  $(z_1, \dots, z_m)$
- *Conjunto de soluciones no dominadas (NDS Set).*

**Actualización** (hasta el límite  $G$ ):

Para  $i = 1, \dots, N$

- $y_i \leftarrow EOP(P_i)$
- Evaluar  $y$  ( $F(y)$ )
- Actualizar  $z$
- Actualizar soluciones vecinas de  $i$ .
- Actualizar NDS Set

**Salida:**

- Aproximación al frente de Pareto. (NDS Set)
-

- **Población**  $P = [I_1, \dots, I_N]$ . Se inicializa con valores aleatorios (dentro del espacio de búsqueda) para cada una de las variables de cada uno de los individuos. Para ello se genera una matriz  $\hat{P}_{N \times p}$  de números aleatorios (entre 0 y 1). De forma que ahora  $P$  es obtenida aplicando a cada elemento de cada columna la expresión  $P_{ij} = \hat{P}_{ij} (x_{Uj} - x_{Lj}) + x_{Lj}$ , obteniendo un número aleatorio entre  $x_{Lj}$  y  $x_{Uj}$ . De forma que la población  $P$  es almacenada como una matriz  $P_{N \times p}$  en la que cada fila corresponde al individuo  $P_i$  asociado al problema  $i$ ; y cada columna corresponde al valor de las características de dicho individuo (genotipo).
- **Valoración de la población**  $VP_{N \times m} = F(P)$ . Es inicializada con la valoración de los individuos de  $P$ . Para ello a cada una de los individuos (filas) de  $P$  son valorados según las funciones objetivo  $f_1, \dots, f_m$ , dando lugar a la matriz  $VP$ , en la que cada fila  $i$  corresponde a las valoraciones del individuo  $P_i$  (fenotipo) y cada columna  $j$  a la valoración de dicho individuo según  $f_j$ .
- **Punto de referencia**  $z = (z_1, \dots, z_m)$ . Corresponde al vector de los mejores valores encontrados para cada una de las funciones objetivo  $f_1, \dots, f_m$ . Teniendo en cuenta que inicialmente solo se tienen los individuos de  $P$ , cuyas valoraciones corresponden a  $VP$ , para la inicialización de  $z$  basta coger el mínimo de cada columna de  $VP$  que será el mejor (menor) valor encontrado hasta el momento).
- **Conjunto de soluciones no dominadas NDS Set**. Corresponde al conjunto de vectores en el espacio de objetivos (valoraciones) que no son dominada por ninguna encontrada hasta el momento. Teniendo en cuenta que inicialmente solo se han encontrado las soluciones de  $VP$  basta recorrer  $VP$  y añadir  $NDS$  Set aquellas que no son dominadas por ninguna otra. Esto es equivalente a tomar el inicialmente  $NDS$  Set y realizar una actualización de  $NDS$  Set con cada solución (vector fila) de  $VP$ .

### 3.3. Desarrollo evolutivo (iterativo)

Una vez inicializado el algoritmo se lleva a cabo un proceso iterativo de evolución durante  $G - 1$  iteraciones (de forma que el algoritmo genera en total  $G$  generaciones y realiza  $G \cdot N$  evaluaciones. Durante cada una de las iteraciones del proceso evolutivo se lleva a cabo:

#### 3.3.1. Reproducción

Consistente en la generación de los nuevos descendientes. Para ello, se pueden utilizar distintos operadores evolutivos con la condición de que devuelvan un único descendiente. Para una mayor generalidad, se propone que la función generadora de los descendientes  $EOP$  sea dada al algoritmo también como entrada a fin de poder dar un marco general para la experimentación. En concreto la experimentación se llevará a cabo, tal y como se propone en la metodología con 3 operadores evolutivos (descritos en la metodología). Exponemos a continuación los detalles de implementación para cada uno de ellos.

---

**Algoritmo 2:** Pseudocódigo EOP1

---

**Entrada:**

- $i$ : índice del individuo actual
- $P$ : Población (genotipos)
- $VP$ : Valoración de la población (fenotipos)
- $B_i$ : Índices de vecinos de  $i$
- $\Omega$ : Espacio de búsqueda.

- Parámetros :
  - $F_s$  : Pool de valores para  $F$
  - $CR$  : Probabilidad de cruce
  - $PM$  : Probabilidad de mutación
  - $SIG$  : Apertura de la mutación

**(1) Mutación de  $P_i$ :** Vector mutante  $\hat{y}$ .

1.  $r_1, r_2, r_3, r_4, r_5 \in B_i$ .
2.  $\hat{y} \leftarrow P_{r1} + F \cdot (P_{r2} - P_{r3}) + rand \cdot (P_{r2} - P_{r3})$

**(2) Cruce de  $P_i$  con  $\hat{y}$ :**

1.  $Cp \leftarrow (\dots, Cp_\delta = 1, \dots) \in \{0, 1\}^p$ .
2.  $y_j \leftarrow Cp \odot \hat{y} + (\mathbf{1}^{(p)} - Cp) \odot P_i$

**(3) Mutación de  $y$ :**

Para  $j = 1, \dots, \dim(P_i)$

1. Si  $rand \in [0, 1] < PM$ :
  - a)  $\sigma_j \leftarrow |x_{Uj} - x_{Lj}| / SIG$
  - b)  $y_j \leftarrow y_j + rand \in \mathcal{N}(0, \sigma_j)$

**(4) Reparación de  $y$ :**

Para  $j = 1, \dots, \dim(P_i)$

1.  $y_j \leftarrow \min(\max(y_j, x_{Lj}), x_{Uj})$

**Salida:** Descendiente  $y \in \Omega$

---

**EOP1.** Basado en DE + GP. En el *algoritmo 2* se presenta el pseudocódigo de la función que describimos a continuación. El operador evolutivo EOP1 toma un individuo  $P_i$  y genera un descendiente  $y$  a partir de él, de forma que:

1. Realiza una primera mutación de  $P_i$  (procedente de DE) de forma que se genera un vector mutante a partir de cinco vectores escogidos del entorno de  $P_i$ . En concreto se escogen al azar cinco elementos del vecindario de  $i$  (correspondientes a los subproblemas  $r1, r2, r3, r4$  y  $r5$ , con reemplazamiento si es necesario) y se genera el vector mutante aplicando la expresión correspondiente a partir de los vectores asociados a los subproblemas que son justamente las filas correspondientes a los índices  $r1, r2$  y  $r3$  de la población (matriz)  $P$ , de forma que el vector mutante corresponde a  $\hat{y} = P_{r1} + F(P_{r2} - P_{r3})$ . Con  $F$  un parámetro de la función establecido por el usuario (típicamente 0,5).
2. Realiza el cruce del individuo  $P_i$  con el vector mutante  $\hat{y}$ . Para ello se genera un vector de números aleatorios booleanos  $CP$  (crossing points) (con probabilidad  $CR$ , dada por el usuario, típicamente 0,5), esto es un vector de valores 0 y 1, de forma que para garantizar que el vector resultante contiene al menos un elemento del vector mutante, una posición aleatoria de  $CP$  es fijada a 1. Ahora para el vector resultante (descendiente) se toman de  $\hat{y}$  aquellas componentes en las que  $CP_j = 1$  y de  $P_i$  aquellas que son 0, esto es justamente (en operaciones vectoriales)  $y = CP \odot \hat{y} + (\mathbf{1}^{(p)} - CP) \odot P_i$  (con  $\mathbf{1}^{(p)}$  el vector  $p$ -dimensional con todas sus componentes 1).
3. Realiza una perturbación gaussiana en algunas de las componentes del descendiente. De forma que para cada una de las componentes se sortea aleatoriamente con probabilidad  $PM$  (dada por el

usuario o en su ausencia  $1/p$ , con  $p$  el número de componentes de los individuos) y en caso favorable dicha componente  $y_j$  es perturbada añadiéndole una cantidad aleatoria obtenida de una distribución normal (gaussiana) de media 0 y desviación estándar  $\frac{|x_{Uj} - x_{Lj}|}{SIG}$  (donde  $SIG$  corresponde a un parámetro ajustable por el usuario, típicamente 20).

4. Finalmente cada una de las componentes de  $y$  que se encuentren fuera del espacio de búsqueda son ajustadas a la cota correspondiente, equivalente a aplicar la función  $y_j = \min(\max(y_j, x_{Lj}), x_{Uj})$  sobre cada componente.

---

**Algoritmo 3:** Pseudocódigo EOP2

---

**Entrada:**

- $i$ : índice del individuo actual
- $P$ : Población (genotipos)
- $VP$ : Valoración de la población (fenotipos)
- $B_i$ : Índices de vecinos de  $i$
- $\Omega$ : Espacio de búsqueda.
- Parámetros :  $F, CR, PM, \eta$

**(1) Mutación de  $P_i$ :** Vector mutante  $\hat{y}$ .

1.  $r_1, r_2, r_3, r_4, r_5 \in B_i$ .
2.  $\hat{y} \leftarrow P_{r1} + F \cdot (P_{r2} - P_{r3}) + rand \cdot (P_{r4} - P_{r5})$

**(2) Cruce de  $P_i$  con  $\hat{y}$ :**

1.  $Cp \leftarrow (\dots, Cp_\delta = 1, \dots) \in \{0, 1\}^p$ .
2.  $y_j \leftarrow Cp \odot \hat{y} + (\mathbf{1}^{(p)} - Cp) \odot P_i$

**(3) Mutación de  $y$ :**

Para  $j = 1, \dots, \dim(P_i)$ :

1. Si  $rand \in [0, 1] < PM$ :
  - a)  $\mu \leftarrow rand \in [0, 1]$
  - b) Si  $\mu < 0,5$  :  $\sigma_j \leftarrow (2\mu)^{\frac{1}{n+1}}$
  - c) Si no :  $\sigma_j \leftarrow 1 - (2 - 2\mu)^{\frac{1}{n+1}}$
  - d)  $y_j \leftarrow y_j + \sigma_j$

**(4) Reparación  $y$ :**

Para  $j = 1, \dots, \dim(P_i)$ :  $y_j \leftarrow \min(\max(y_j, x_{Lj}), x_{Uj})$

**Salida:** Descendiente  $y \in \Omega$ 


---

**EOP2.** Basado en DE + SBX. En el *algoritmo 3* se presenta el pseudocódigo de la función que describimos a continuación. El operador evolutivo EOP2 toma un individuo  $P_i$  y genera un descendiente  $y$  a partir de él, de forma que:

1. Realiza una primera mutación de  $P_i$  (procedente de DE) de forma que se genera un vector mutante a partir de cinco vectores escogidos del entorno de  $P_i$ . En concreto se escogen al azar 5 elementos del vecindario de  $i$  (correspondientes a los subproblemas  $r_1, r_2, r_3, r_4$ , y  $r_5$ ) y se genera el vector mutante aplicando la expresión correspondiente (mostrada en la metodología a partir de los vectores asociados a los subproblemas, correspondiente a  $\hat{y} = P_{r1} + F(P_{r2} - P_{r3}) + rand(P_{r4} - P_{r5})$ ). Con  $F$  un parámetro de la función establecido por el usuario (típicamente 0,5) y  $rand$  un número aleatorio entre 0 y 1.
2. Realiza el cruce del individuo  $P_i$  con el vector mutante  $\hat{y}$  utilizando el cruce típico de DE, detallado en EOP1

3. Realiza una mutación inspirada en el operador SBX (Simulated Binary Crossover). De forma que para cada una de las componentes se sortea aleatoriamente con probabilidad  $PM$  (dada por el usuario o en su ausencia  $1/p$ , con  $p$  el número de componentes de los individuos) y en caso favorable dicha componente  $y_j$  es perturbada añadiéndole una cantidad obtenida mediante la expresión inferior, de forma que  $PM, \eta$  son parámetros (ajustables) de entrada.

$$y_j = \begin{cases} y_j + \sigma_k \cdot (x_{Uj} - y_{Lj}) & \text{si } rand < PM \\ y_j & \text{e.o.c} \end{cases}, \sigma_k = \begin{cases} (2 \cdot rand)^{\frac{1}{\eta+1}} - 1 & \text{si } rand < 0,5 \\ 1 - (2 - 2 \cdot rand)^{\frac{1}{\eta+1}} & \text{e.o.c} \end{cases}$$

4. Finalmente cada una de las componentes de  $y$  que se encuentren fuera del espacio de búsqueda son ajustadas a la cota correspondiente, equivalente a EOP1.

---

**Algoritmo 4:** Pseudocódigo EOP3

---

**Entrada:**

- $i$ : índice del individuo actual
- $P$ : Población (genotipos)
- $VP$ : Valoración de la población (fenotipos)
- $B_i$ : Índices de vecinos de  $i$
- $\Omega$ : Espacio de búsqueda.
- Parámetros :  $PM, SIG$

**(1) Cruce de  $P_i$ :**

1.  $S \leftarrow \{r_1, \dots, r_\epsilon\} \subseteq B_i$
2. Para  $j = 1, \dots, \dim(P_i)$ :  $y_j \leftarrow (P_{rand \in S})_j$

**(3) Reparación  $y$ :**

Para  $j = 1, \dots, N$ :  $y_j \leftarrow \min(\max(y_j, x_{Lj}), x_{Uj})$

**(2) Mutación de  $y$ :**

Para  $j = 1, \dots, \dim(P_i)$

1. Si  $rand \in [0, 1] < PM$ :
  - a)  $\sigma_j \leftarrow |x_{Uj} - x_{Lj}| / SIG$
  - b)  $y_j \leftarrow y_j + rand \in \mathcal{N}(0, \sigma_j)$

**Salida:** Descendiente  $y \in \Omega$

---

**EOP3.** Basado en UX + GP. En el *algoritmo 4* se presenta el pseudocódigo de la función que describimos a continuación. El operador evolutivo EOP3 toma un individuo  $P_i$  y genera un descendiente  $y$  a partir de él, de forma que:

1. Se realiza un cruce uniforme de forma que se eligen de entre el vecindario de  $i$  *epsilon* (parámetro de entrada) vecinos aleatoriamente (con igual probabilidad)  $r_1, r_2, \dots, r_\epsilon$ . Ahora el descendiente  $y$  es generado de forma que el valor de cada componente (cromosoma) es elegido aleatoriamente entre los valores de dicha componente de los vecinos seleccionados.
2. Realiza una mutación gaussiana de forma análoga a EOP1.
3. Finalmente cada una de las componentes de  $y$  que se encuentren fuera del espacio de búsqueda son ajustadas a la cota correspondiente, equivalente a EOP1.

### 3.3.2. Evaluación del descendiente

Una vez generado el sucesor éste es evaluado respecto a cada uno de los objetivos almacenándose en un vector  $v\mathbf{y}$  las valoraciones tal que la posición  $k$  de dicho vector corresponde a  $f_k(\mathbf{y})$ .

### 3.3.3. Actualización del punto de referencia

Como  $\mathbf{z}$  debe mantener en todo momento los mejores valores obtenidos por alguno de los individuos, es necesario actualizar aquellas componentes en las que  $v\mathbf{y}$  sea mejor (menor) que  $\mathbf{z}$ , de forma que para cada componente  $j = 1, \dots, m$ , se actualiza  $\mathbf{z}$  con  $z_j = \min(z_j, v\mathbf{y}_j)$ .

### 3.3.4. Actualización de los vecinos

Equivalente, en cierto modo, al proceso de selección de los nuevos individuos en otros algoritmos evolutivos, de forma que el descendiente sustituirá al individuo considerado si y sólo si es mejor que él, teniendo en cuenta la formulación de Tchebycheff si  $g^{te}(\mathbf{y}|\boldsymbol{\lambda}_i, \mathbf{z}) \leq g^{te}(\mathbf{P}_i|\boldsymbol{\lambda}_i, \mathbf{z})$ . Téngase en cuenta que esta solución también puede ser mejor para los problemas vecinos, por ello, estos también son actualizados siguiendo el mismo criterio. Para ello, para cada uno de los vecinos  $j$  de  $i$  ( $B_i$ ) el elemento (fila) de la matriz  $P$  es sustituido  $\mathbf{y}$  si se cumple el criterio de Tchebycheff (actualizando consecuentemente  $VP$ ). Recuérdese que  $i \in B_i$  por tanto basta ejecutar la acción sobre el vecindario (sin hacerlo de forma separada para el propio individuo).

La actualización de muchos de los vecinos puede llevar al algoritmo a una convergencia prematura (estancamiento) por ello se controla mediante el parámetro  $UN$  el número máximo de actualizaciones (mediante un contador que es incrementado cada vez que se produce una actualización).

### 3.3.5. Actualización de *NDS Set*

Para mantener en todo el momento la consistencia del conjunto cada vez que es generado un descendiente se actualiza el conjunto añadiendo la valoración asociada y eliminando las soluciones dominadas por ella, si procede. Para ello se recorre el conjunto de soluciones dominadas, comparando cada solución con  $\mathbf{F}(\mathbf{y})$  en el momento en que se encuentre una que la domine, el proceso es abortado. En caso de que no se encuentre ninguna, se habrán ido eliminando, al mismo tiempo que se recorren, aquellas soluciones dominadas por  $\mathbf{F}(\mathbf{y})$ , por lo que basta añadir  $\mathbf{F}(\mathbf{y})$  al conjunto para completar la actualización.

### 3.3.6. Otras cuestiones

Nótese que el proceso de evolución (actualización de  $P$  y  $VP$ ) se realiza de forma síncrona, esto es la actualización de  $P_i$  se realiza directamente sobre  $P$  y no en una matriz auxiliar que luego sea volcada. Por ello, los individuos son recorridos en orden aleatorio en cada iteración (mediante una lista con los

índices que es reordenada aleatoriamente en cada iteración) de manera que las actualizaciones influyan (probabilísticamente) igual en todos los individuos.

A fin de que se pueda testear el funcionamiento del algoritmo al final de cada iteración, la población (su valoración) es volcada a un fichero *ASCII* en el que cada fila corresponde a un individuo y sus columnas reflejan las valoraciones del individuo y una última columna con el número de restricciones violadas. También es generado un fichero con el contenido de todas las generaciones.

## 4. MOEA/D Experimentación y resultados

La implementación del algoritmo propuesto se ha llevado a cabo utilizando el lenguaje *Python* haciendo uso de las librerías *Numpy*, para el manejo de vectores y matrices, *os* y *shutil* para el manejo de los ficheros y directorios, y *matplotlib* y *colour* para algunas representaciones gráficas. El código Python (dibidamente comentado) se encuentra disponible en: [Código MOEA/D](#). Para la evaluación de las métricas se ha utilizado el software proporcionado en la asignatura.

A continuación se muestran los resultados obtenidos para la evaluación del algoritmo propuesto con los distintos operadores. Para ello testearemos el mismo con la función matemática ZDT3 (con 30 dimensiones)[6], comprobando tanto el comportamiento del mismo como evaluando una serie de métricas para los distintos operadores y comparándolo también con el algoritmo de referencia *NSGA-II* [7].

### 4.1. Evaluación MOEA/D + EOP1 con ZDT3

Vamos a realizar algunas pruebas para comprobar la efectividad del algoritmo. Tanto utilizando una capacidad de cómputo de 10000 evaluaciones (en distintos casos) como una de 4000 evaluaciones (con distintos repartos). Para ello mostraremos algunas gráficas del comportamiento típico del algoritmo. Aunque en esta sección se muestra una o dos ejecuciones se han realizado 10 ejecuciones, cuyos resultados se presentan en los apéndices.

#### 4.1.1. Experimentación con 10000 ev.

En este apartado realizamos un estudio (gráfico) del comportamiento del algoritmo, analizando la convergencia y diversidad de la población a lo largo de la ejecución del algoritmo. Propósito de tal estudio es además tratar de esclarecer la influencia de los parámetros  $G$  y  $N$  en el proceso evolutivo presentando una comparativa de las soluciones finales variando dichos parámetros, manteniendo constante el número

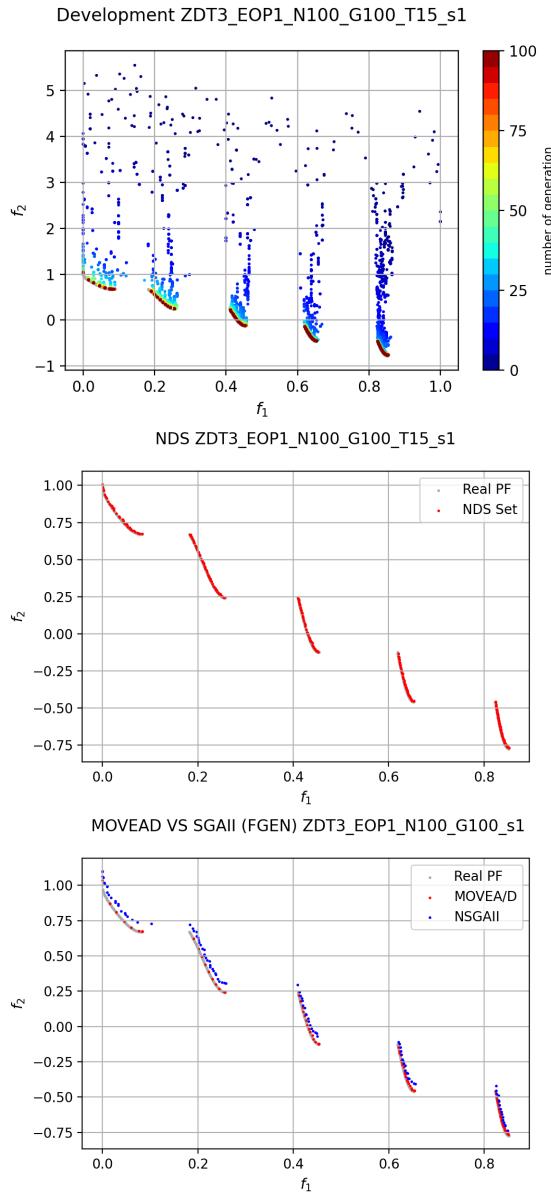


Figura 2: MOEA/D + EOP1 + N100G100

de evaluaciones, para los casos ( $G = 100, N = 100$ ), ( $G = 250, N = 40$ ) y ( $G = 50, N = 200$ ).

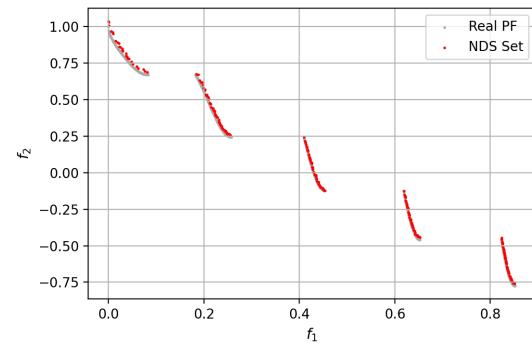
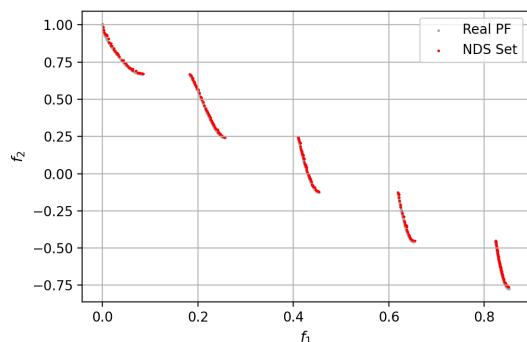
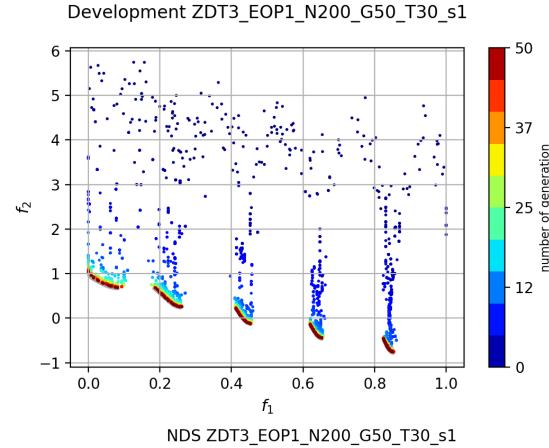
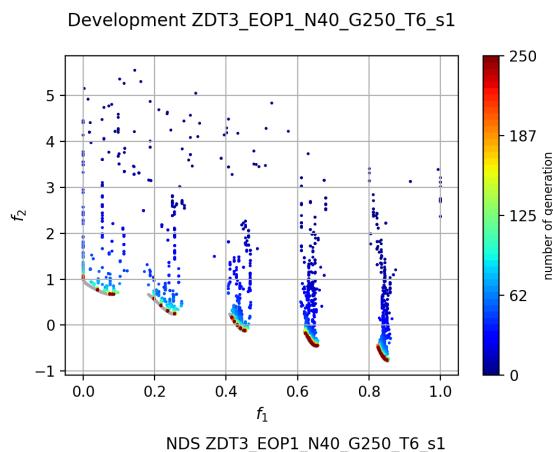
En la *figura 2* presentamos tres gráficos pertenecientes a una ejecución característica del algoritmo con el operador evolutivo EOP1,  $N = 100, G = 100$  y  $T = 15$  (para todos los casos se ha tomado un 15% (recuérdese que en los anexos se muestran varias ejecuciones que validan las discusiones) el primer diagrama muestra el desarrollo de las soluciones en el espacio de objetivos a lo largo de la ejecución del algoritmo (en las distintas generaciones). En él podemos apreciar que, tal como esperábamos a medida que el algoritmo avanza (pasan las generaciones) las soluciones tienden a ir convergiendo hacia el frente real de Pareto (señalado en gris), además de forma bastante rápida (los azules corresponden a las primeras generaciones y los anaranjados y rojos a las últimas) de forma que en menos de 50 generaciones se sitúan ya próximas al frente real. Otro aspecto a destacar es la diversidad, en los algoritmos evolutivos es corriente que las soluciones tiendan a concentrarse (se pierde diversidad); en este caso podemos ver dicha tendencia pero al mismo tiempo se puede notar que se tiende a conservar cierta dispersión en los puntos lo que denota una intención de mantener la diversidad (objetivo de todo algoritmo evolutivo).

En el segundo diagrama se presenta el conjunto de las soluciones no dominadas calculadas por el algoritmo, que parece corresponder en gran medida con el frente real de pareto, tanto en proximidad (convergencia) como en cobertura (dispersión) en los puntos del frente. Lo que denota, a priori, y a falta de métricas un buen comportamiento del algoritmo. De hecho, esos puntos podrían ser considerados como la salida del algoritmo, esto es la aproximación al frente de pareto que parece ser fiel al frente real (mostrado en gris).

Finalmente, presentamos una gráfica comparativa para nuestro algoritmo y para el algoritmo *NSGA-II*. Para ello hemos representado las soluciones en el espacio de objetivos, las cuales corresponden a la última generación de ejecuciones en iguales condiciones ( $G = 100, N = 100$ ) para ambos algoritmos. Podemos ver que en cuanto a convergencia (proximidad al frente real) el frente del nuestro algoritmo es sensiblemente mejor que el frente proporcionado por *NSGA-II* (una gran mayoría de los puntos rojos están por debajo de los azules), mientras que en dispersión parece que los puntos del frente de *NSGA-II* se distribuyen más uniformemente que los de nuestro algoritmo. Pero nótese que si consideramos como salida la proporcionada por el frente de las soluciones no dominadas la tanto la cobertura como la convergencia parecen más que aceptables.

Veámos ahora unas gráficas análogas a las presentadas pero para el caso ( $G = 250, N = 40$ ) en la *figura 3* podemos notar que el comportamiento es similar al caso anterior, esto es, como se espera a través de las generaciones las soluciones van aproximándose la convergencia al igual que en el caso anterior sigue siendo es buena, sin embargo se nota que la reducción en el número de subproblemas afecta de manera notable a la dispersión, sobre todo en las últimas generaciones. Sin embargo, si miramos el frente de soluciones no dominadas, vemos que tanto en convergencia (a priori, sin otro algoritmo de referencia) que tanto el grado de convergencia de las soluciones como el grado de dispersión hacen que se aproximen en buena medida al frente real de pareto, por lo que nos incita a pensar que la pérdida de diversidad debe darse relativamente al final porque existen soluciones buenas a lo largo prácticamente de todo el frente. Si comparamos ahora la última generación con la del algoritmo *NSGA-II* (ejecutado en las mismas condiciones) podemos ver que ahora la superioridad no es tan patente como en el casi anterior y ambos algoritmos alcanzan un grado análogo de convergencia y dispersión (en este último aspecto quizá

NSGA-II parezca un poco mejor).



**Figura 3:** MOEA/D + EOP1 + N40G250

Y en último lugar comprobemos cuál es el resultado de elevar el número de subproblemas frente al número de generaciones. Dado que según hemos visto hasta ahora la convergencia suele ser rápida, y que el número de subproblemas favorecerá la diversidad, a priori el comportamiento debería ser satisfactorio. Veámos qué ocurre para el caso ( $G = 250, N = 40$ ) en las gráficas presentadas en la figura 4. Podemos notar que la convergencia es bastante buena y la diversidad también permite que en las últimas generaciones se cubran los tramos del frente de manera más o menos uniforme. De hecho, si observamos

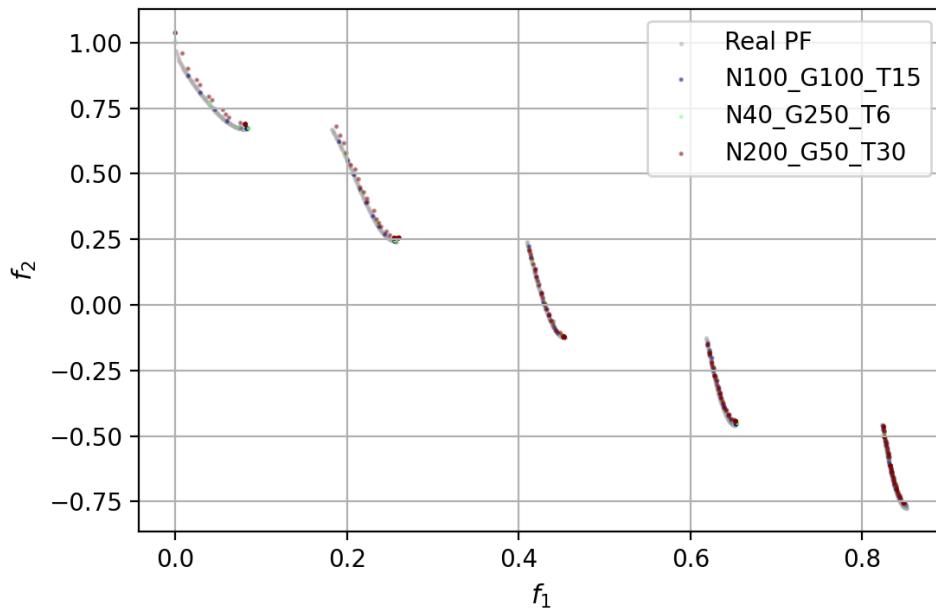
**Figura 4:** MOEA/D + EOP1 + N200G50

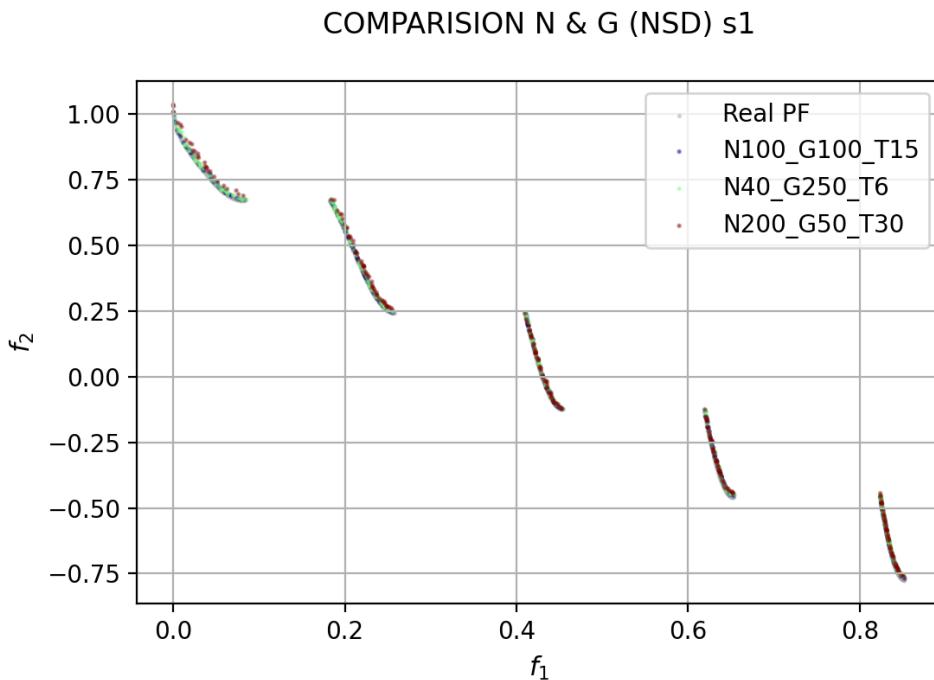
el diagrama de las soluciones no dominadas podemos notar que, primero el ajuste es ligeramente peor que en los otros casos (normal dado que tiene menos generaciones para tratar de ajustarse) y la cobertura es buena y se reparte de forma más o menos uniforme. Si comparamos con la ejecución del *NSGA-II* en este caso nuestro algoritmo es claramente mejor en convergencia y también algo mejor en cobertura (tramos más largos).

Finalmente vamos a realizar una comparativa entre los tres casos tanto para los frentes de la última generación como para los frentes *NSD*. En la figura 5 se muestran las dos gráficas de comparativa de los frentes anteriormente indicados. En cuanto a las soluciones de la generación final todas se encuentran bastante superpuestas, quizás la roja con una ligera menor convergencia (proximidad al frente) y la verde con una menor convergencia pudiendo ser la azul la más adecuada, esto es un balance en el número de generaciones y número de subproblemas lo que proporciona la mejor solución, aunque los resultados no son concluyentes y trataremos de realizar un estudio más profundo con el uso de las métricas.

En cuanto al *NSD*, las sensaciones son similares se nota bastante superposición entre los puntos, quizás con la roja un poco menor de convergencia en algunas partes y la azul y la verde muy similares, los resultados no son nada concluyentes, así que intentaremos clarificarlos con el uso de métricas y el correspondiente estudio estadístico de las mismas.

COMPARISION N & G (Final Gen.) s1



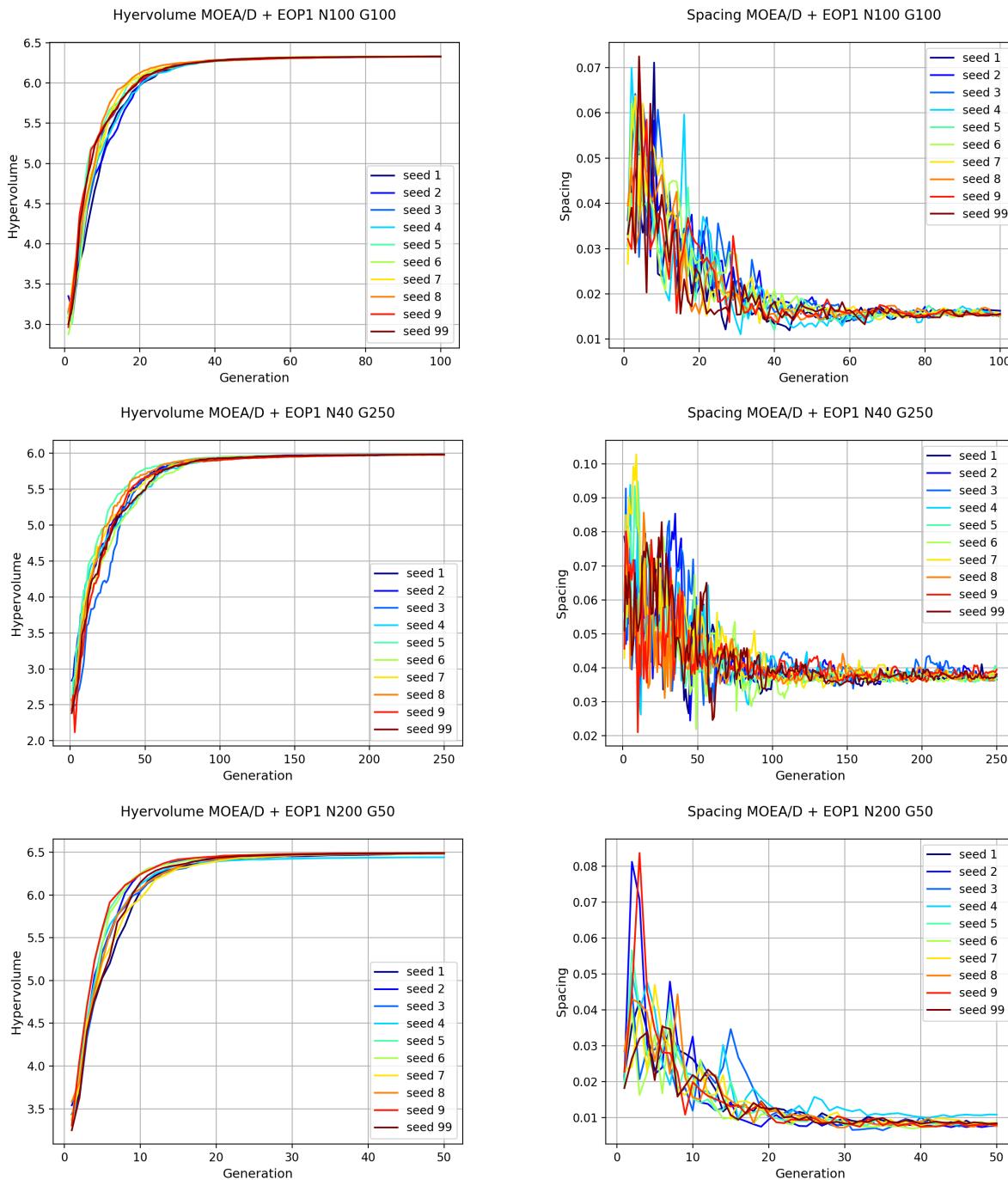


**Figura 5:** MOEA/D + EOP1. Comparación de casos

#### 4.1.2. Análisis de métricas para 10000 ev.

Presentando el estudio preliminar anterior vamos a tratar de profundizar para exclarecer y llevar a cabo una discusión más profunda de los casos anteriores. Para ello realizaremos un estudio de algunas métricas para los casos ya presentados. Tales métricas corresponden al hipervolumen, espaciado y cover set.

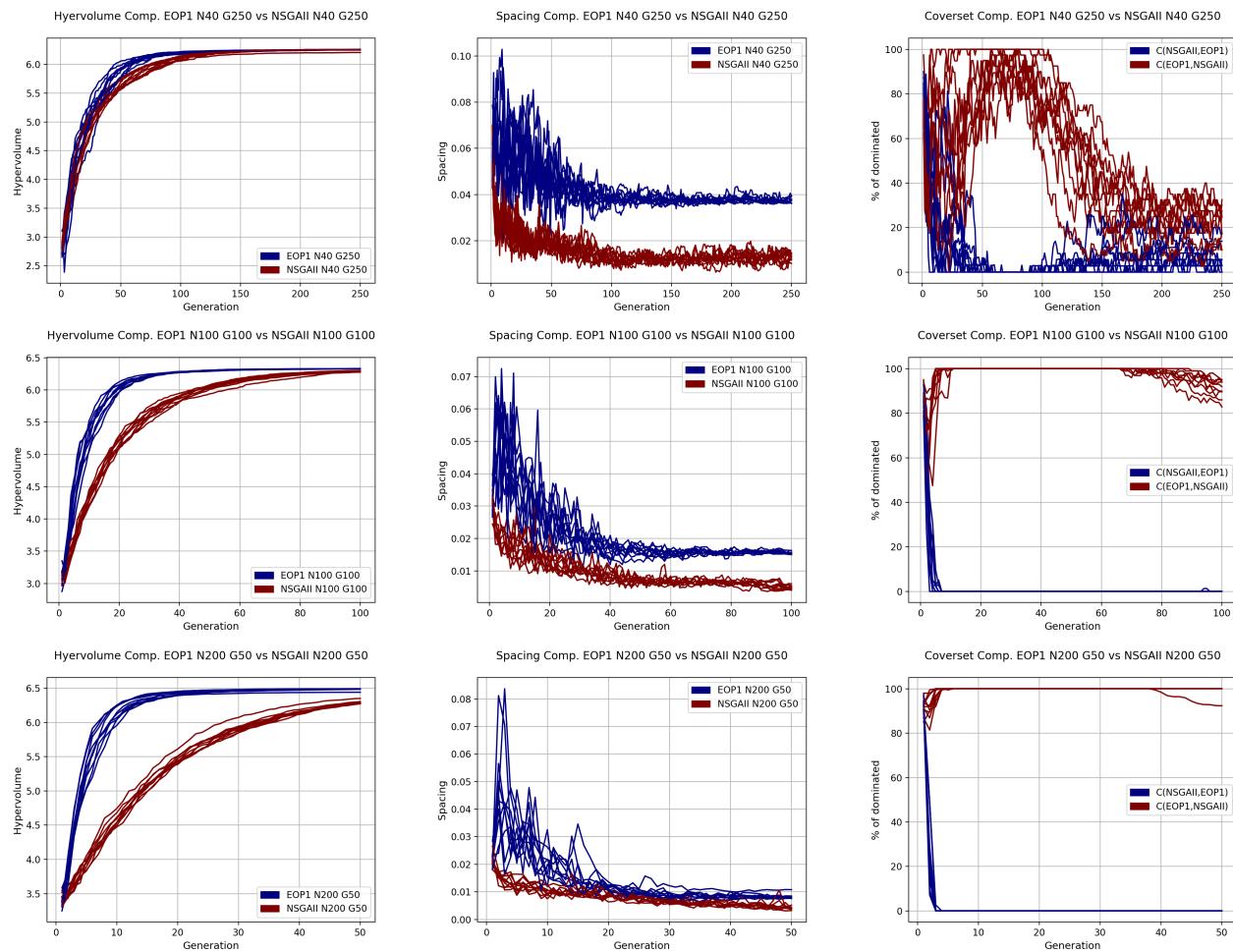
Comencemos viendo algunas gráficas asociadas a las métricas para los casos planteados en el apartado anterior. En la figura 6 se muestra la evolución de hipervolumen y el espaciado en durante las distintas generaciones para 10 ejecuciones del algoritmo. Podemos observar que en todos los casos el desarrollo del hipervolumen como el del espaciado se comportan de manera bastante uniforme para todas las generaciones, lo que denota que nuestro algoritmo es robusto (aunque, lógicamente, dos ejecuciones distintas tienen comportamientos distintos, dado el carácter estocástico del algoritmo). Aunque no es posible hacer un análisis conjunto del hipervolumen (ya que el punto de referencia para su cálculo es distinto en cada caso) sí podemos destacar la rápida convergencia del algoritmo (que destacamos también en el estudio preliminar) y el aparente estancamiento final, por lo que es preferible primar el número de subproblemas frente al número de generaciones (en valores razonables) como sugiere también el comportamiento del espaciado.



**Figura 6:** MOEA/D + EOP1. M  ticas para 10000 EV

A continuaci  n presentaremos algunas gr  ficas comparativas del comportamiento del algoritmo frente a *NSGAII*. En la figura 7 se muestran las graficas con dichas comparativas en las que podemos notar que en todos los casos el espaciado en el algoritmo NSGAII se compora mejor que en algoritmo propuesto mientras que el hipervolumen suele ser al rev  s. Como vimos en el estudio preliminar el algoritmo s   suele

converger a soluciones mejores que el algoritmo *NSGAII* pero también suele tender a concentrar más las soluciones. Sin embargo si comprobamos la métrica cover set, sí podemos ver que para el segundo y el tercer caso ( $N=100$  y  $N=200$ ) el algoritmo propuesto tiende a dominar al *NSGAII*, para el primer caso en todas las ejecuciones el frente del algoritmo *NSGAII* quedó dominado por el de *MOEA/D + EOP1* en más del 80 % de sus puntos, mientras que el frente de *NSGAII* no es apreciable que dominase al *MOEA/D + EOP1* en ni siquiera un 2 % en ninguna de las ejecuciones; y además este comportamiento es patente a partir las 5 primeras iteraciones. No es así para el caso  $N = 40$ , en el que el nuestro algoritmo sí supera en hipervolumen al *NSGAII* pero es sensiblemente peor en cuanto al espaciado y en cuanto al cover set en ningún caso ninguno de los dos frentes es dominado más de un 40 % por el otro. aunque nuestro algoritmo tienda a quedarse por debajo. Como venimos repitiendo a nuestro algoritmo le aporta más un número alto de subproblemas que de generaciones.



**Figura 7:** MOEA/D + EOP1. Comparación de métricas con NSGAII para 10000 EV.

Aunque en este apartado no hemos realizado (explícitamente) un análisis de las soluciones (población final y NSD) en el último apartado de esta sección presentaremos una comparativa conjunta de las últimas generaciones y de los conjuntos no dominados (NSD) para todos los casos y todos los algoritmos (y operadores).

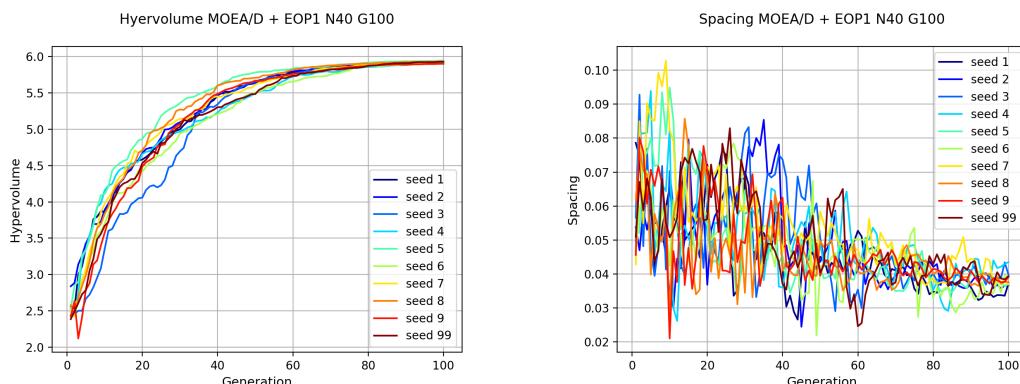
## EXPERIMENTACIÓN PARA 4000 EVALUACIONES

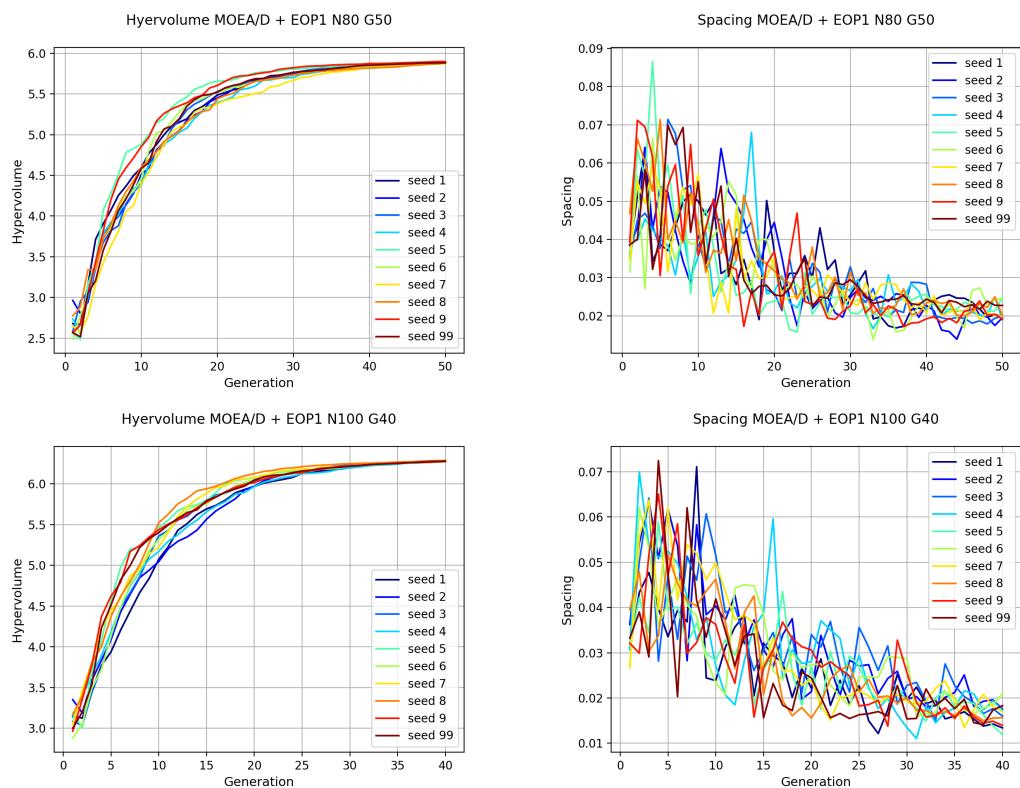
Ya presentamos para 10000 evaluaciones el comportamiento del algoritmo es adecuado, esto es, a través de las generaciones los individuos se aproximan al frente. No vamos a volver a presentarlo para el caso de 4000 evaluaciones, pues sigue lógicamente el mismo esquema (en los apéndices se presentan las gráficas que atestiguan lo expuesto). Por tanto pasaremos directamente a evaluar las métricas y a razonar directamente sobre los resultados obtenidos en dicho análisis.

En la figura 8 se presentan las gráficas del desarrollo del hypervolumen y el espaciado para cada uno de los casos considerados en las 4000 evaluaciones en contra (N = 40, G = 100), (N = 80, G = 50), (N = 100, G = 40). Como podemos apreciar el comportamiento es similar a los presentados para las 10000 evaluaciones, denotándose que la convergencia (hipervolumen) se desarrolla de forma análoga a como lo hacía para los casos previos (viendo la forma de la curva, dado que los valores no son comparables debido a la desigualdad del punto de referencia en cada caso), pero truncada a las iteraciones correspondientes (aproximadamente). De igual forma el espaciado disminuye notablemente con el aumento del número de subproblemas, lo que sin duda se adecua a los resultados presentados previamente.

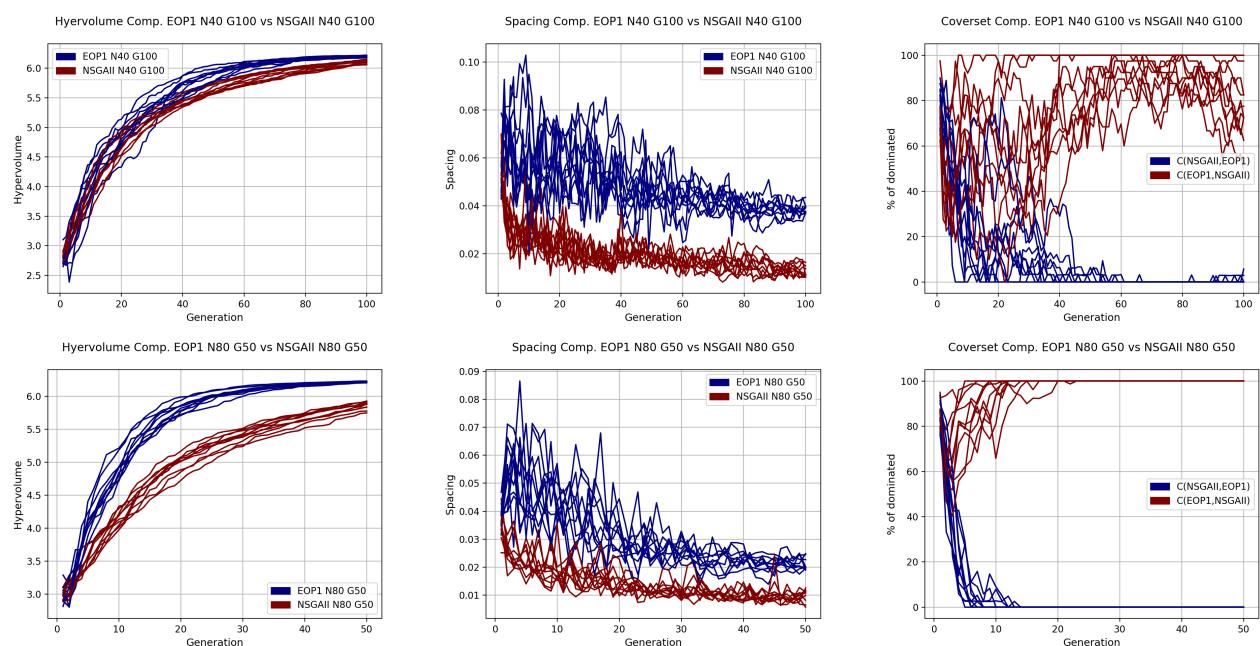
Si realizamos una comparativa (al igual que en el caso de 10000 ev) con el algoritmo *NSGAII* obtenemos conclusiones análogas a las ya presentadas previamente, descubriendo que en cuanto a convergencia en todos los casos nuestro algoritmo supera al algoritmo *NSGAII*, en espaciado ocurre al contrario y en cuanto al cover set, para el caso N = 40 el frente más del 50% del frente (final) de *NSGAII* es dominado por el de nuestro algoritmo, mientras que apenas un 5% del frente de nuestro algoritmo es dominado por el competidor. En los otros dos casos el cover set es mucho más claro de forma que aparentemente (para ambos casos) el 100% del frente es dominado (en todas las ejecuciones) por el frente final (última generación) de nuestro algoritmo y mientras que para ningún punto (o desde luego menor al 1% o 2%) se da el comportamiento contrario.

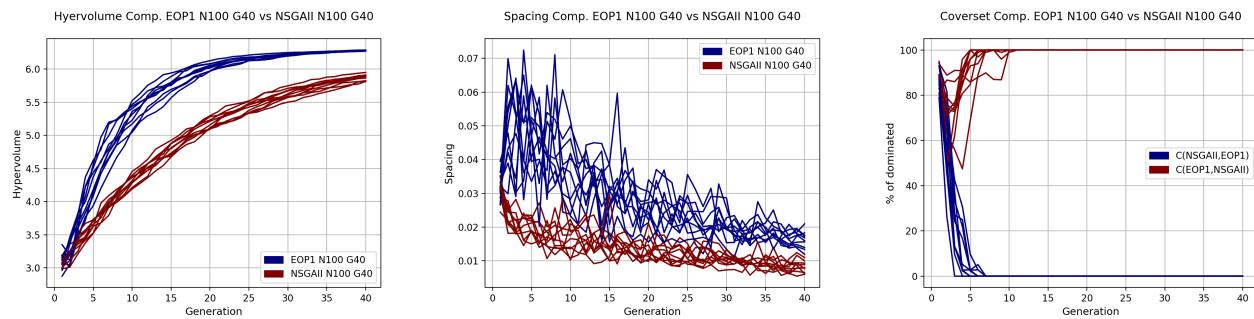
De todo ello destacamos la bondad de nuestro algoritmo en cuanto a la convergencia, no tanto para el espaciado (sobre todo si consideramos la última generación, si consideramos el frente NSD veremos que el espaciado mejora notablemente, presentado en la comparativa final de la tabla 1) y en cuanto al cover set, con el número adecuado de subproblemas (mejor que de generaciones) nuestro algoritmo domina prácticamente en su totalidad a *NSGAII*.





**Figura 8: MOEA/D + EOP1. Métricas para 4000EV**





**Figura 9:** MOEA/D + EOP1. Comparación de métricas con NSGAII para 4000 EV.

## 4.2. Evaluación MOEA/D + EOP2 con ZDT3

De igual manera a lo presentado para EOP1 vamos a realizar algunas pruebas para comprobar la efectividad del algoritmo con EOP2. Tanto utilizando una capacidad de cómputo de 10000 evaluaciones como una de 4000 evaluaciones (con los mismos repartos ya presentados).

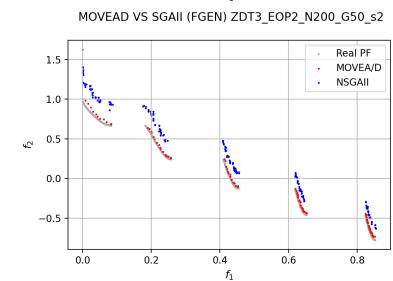
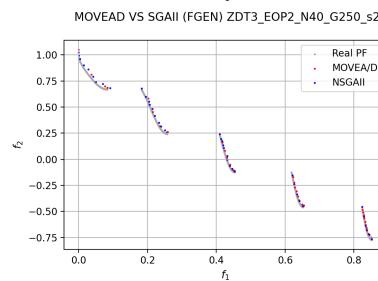
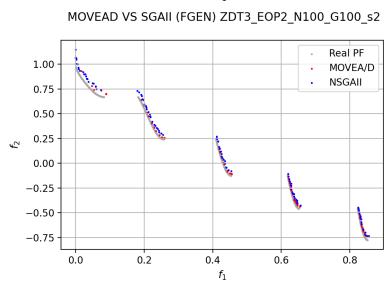
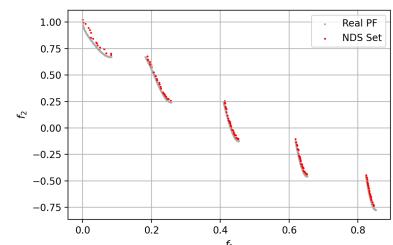
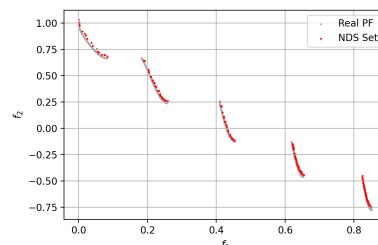
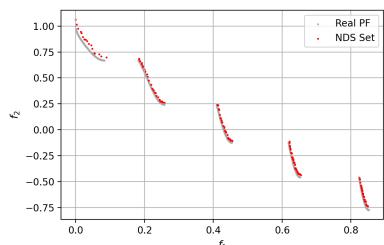
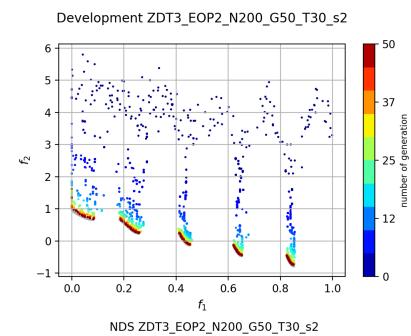
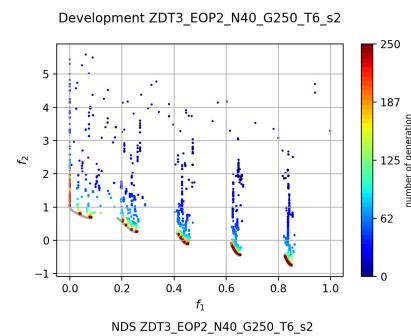
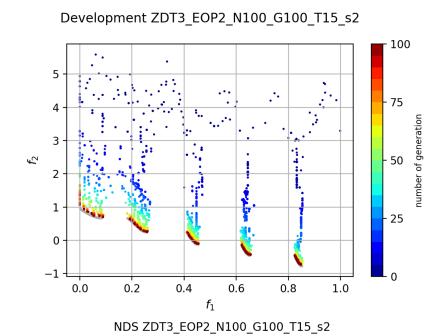
### 4.2.1. Experimentación con 10000 ev.

En la figura 10 presentamos tres gráficos pertenecientes a una ejecución característica del algoritmo con el operador evolutivo EOP3,  $N = 100, G = 100$  y  $T = 15$  (para todos los casos se ha tomado un 15% (recuérdese que en los anexos se muestran varias ejecuciones que validan las discusiones) el primer diagrama muestra el desarrollo de las soluciones en el espacio de objetivos a lo largo de la ejecución del algoritmo (en las distintas generaciones). En él podemos apreciar que, tal como esperábamos a medida que el algoritmo avanza (pasan las generaciones) las soluciones tienden a ir convergiendo hacia el frente real de Pareto (señalado en gris), sin embargo sí podemos notar que dicha convergencia es (generalmente) más lenta que en el caso anterior, aunque parecen alcanzarse buenos valores en la mayoría de las ejecuciones, aunque alguna no haga (ver s1 en los anexos). Otro aspecto a destacar es la diversidad, en los algoritmos evolutivos es corriente que las soluciones tiendan a concentrarse (se pierde diversidad); en este caso podemos ver dicha tendencia pero al mismo tiempo se puede notar que se tiende a conservar cierta dispersión en los puntos lo que denota una intención de mantener la diversidad (objetivo de todo algoritmo evolutivo).

En el segundo diagrama se presenta el conjunto de las soluciones no dominadas calculadas por el algoritmo, que parece estar bastante próximo al frente real de Pareto, tanto en proximidad (convergencia) como en cobertura (dispersión) en los puntos del frente. Lo que denota, a priori, y a falta de métricas un buen comportamiento del algoritmo. De hecho, esos puntos podrían ser considerados como la salida del algoritmo, esto es la aproximación al frente de Pareto que parece ser relativamente fiel al frente real (mostrado en gris).

Finalmente, presentamos una gráfica comparativa para nuestro algoritmo y para el algoritmo *NSGA-II*. Para ello hemos representado las soluciones en el espacio de objetivos, las cuales corresponden a la

última generación de ejecuciones en iguales condiciones ( $G = 100, N = 100$ ) para ambos algoritmos. Podemos ver que en cuanto a convergencia (proximidad al frente real) el frente del nuestro algoritmo es sensiblemente mejor que el frente proporcionado por *NSGA-II* (una mayoría de los puntos rojos están por debajo de los azules), mientras que en dispersión parece que los puntos del frente de *NSGA-II* se distribuyen más uniformemente que los de nuestro algoritmo. Pero nótese que si consideramos como salida la proporcionada por el frente de las soluciones no dominadas tanto la cobertura como la convergencia parecen aceptables.



**Figura 10:** MOEA/D + EOP2  
(N100G100)

**Figura 11:** MOEA/D + EOP2  
(N40G250)

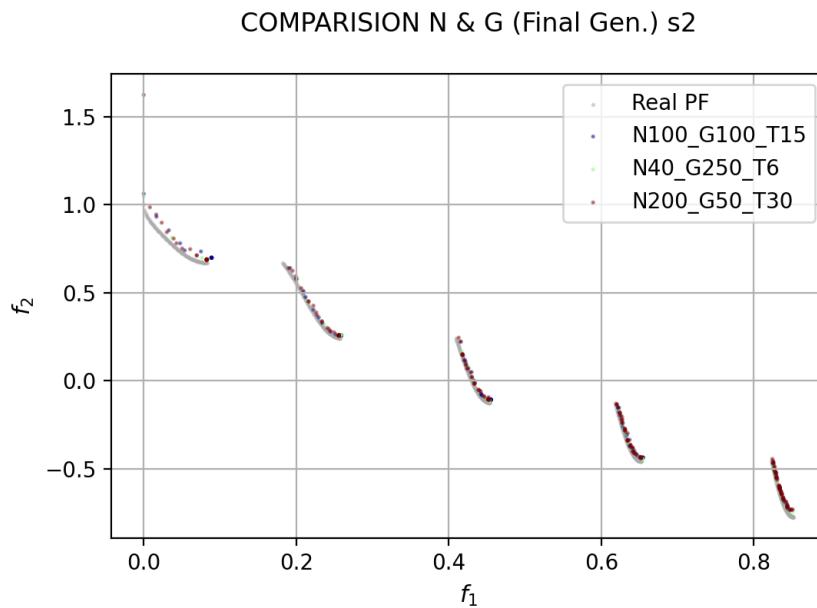
**Figura 12:** MOEA/D + EOP2  
(N200G50)

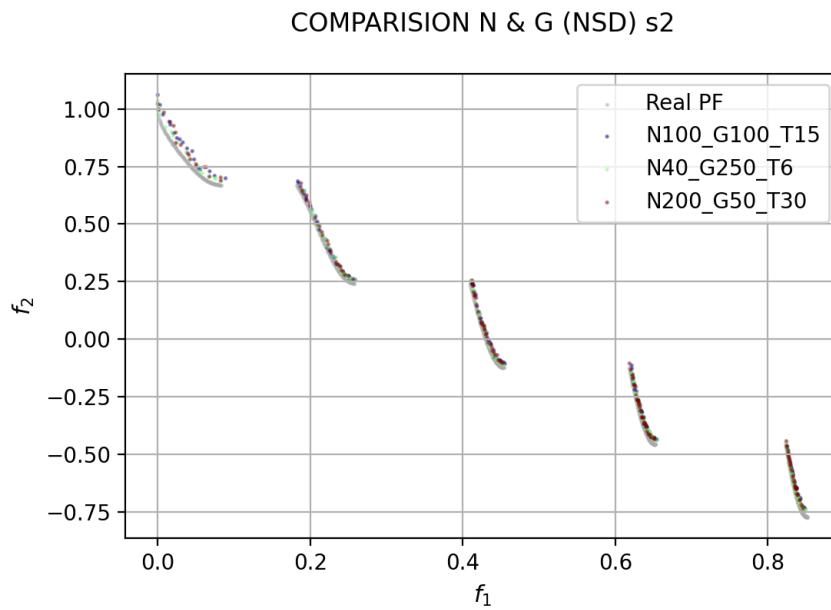
Veámos ahora unas gráficas análogas a las presentadas pero para el caso ( $G = 250, N = 40$ ) en la figura 11 podemos notar que el comportamiento es similar al caso anterior, esto es, como se espera a través de las generaciones las soluciones van aproximándose la convergencia al igual que en el caso anterior sigue siendo es buena, sin embargo se nota que la reducción en el número de subproblemas afecta de manera notable a la dispersión, sobre todo en las últimas generaciones. Sin embargo, si miramos el frente de soluciones no dominadas, vemos que tanto en convergencia (a priori, sin otro algoritmo de referencia) que tanto el grado de convergencia de las soluciones como el grado de dispersión hacen que se aproximen en buena medida al frente real de Pareto, por lo que nos incita a pensar que la pérdida de

diversidad debe darse relativamente al final porque existen soluciones buenas a lo largo prácticamente de todo el frente. Si comparamos ahora la última generación con la del algoritmo *NSGA-II* (ejecutado en las mismas condiciones) podemos ver que ahora la superioridad no es tan patente como en el casi anterior y ambos algoritmos alcanzan un grado análogo de convergencia y dispersión (en este último aspecto quizás *NSGA-II* parezca un poco mejor).

Y en último lugar comprobemos cuál es el resultado de elevar el número de subproblemas frente al número de generaciones. Dado que según hemos visto hasta ahora lo convergencia suele ser rápida, y que el número de subproblemas favorecerá la diversidad, a priori el comportamiento debería ser satisfactorio. Veámos qué ocurre para el caso ( $G = 250, N = 40$ ) en las gráficas presentadas en la figura 12. Podemos notar que la convergencia es bastante buena y la diversidad también permite que en las últimas generaciones se cubran los tramos del frente de manera más o menos uniforme. De hecho, si observamos el diagrama de las soluciones no dominadas podemos notar que, primero el ajuste es ligeramente peor que en los otros casos (normal dado que tiene menos generaciones para tratar de ajustarse) y la cobertura es buena y se reparte de forma más o menos uniforme. Si comparamos con la ejecución del *NSGA-II* en este caso nuestro algoritmo es claramente mejor en convergencia y posiblemente en cobertura.

Finalmente vamos a realizar una comparativa entre los tres casos tanto para los frentes de la última generación como para los frentes *NSD*. En la figura 13 se muestran las dos gráficas de comparativa de los frentes anteriores indicados. En cuanto a las soluciones de la generación final todas se encuentran bastante superpuestas, quizás la verde con una menor dispersión pareciendo que un número de subproblemas más elevado proporciona una mejor solución, aunque los resultados no son concluyentes y trataremos de realizar un estudio más profundo con el uso de las métricas. En cuanto al *NSD*, los resultados no son nada concluyentes, así que intentaremos clarificarlos con el uso de métricas y el correspondiente estudio estadístico de las mismas.





**Figura 13:** MOEA/D + EOP2. Comparación de casos

#### 4.2.2. Análisis de métricas para 10000 ev.

Presentado el estudio preliminar anterior vamos a tratar de profundizar para exclarecer y llevar a cabo una discusión más profunda de los casos anteriores. Para ello realizaremos un estudio de algunas métricas para los casos ya presentados. Tales métricas corresponden al hipervolumen, espaciado y cover set.

Comencemos viendo algunas gráficas asociadas a las métricas para los casos planteados en el apartado anterior. En la figura 14 se muestra la evolución de hipervolumen y el espaciado en durante las distintas generaciones para 10 ejecuciones del algoritmo. Podemos observar que en todos los casos el desarrollo del hipervolumen se comporta de manera bastante uniforme para todas las ejecuciones, sin embargo en el primer caso el espaciado sí varía considerablemente desde menos de 0.04 para algunas ejecuciones hasta casi 0.10 en otras.

Aunque no es posible hacer un análisis conjunto del hipervolumen (ya que el punto de referencia para su cálculo es distinto en cada caso) sí podemos destacar la rápida convergencia del algoritmo, menos clara que para EOP1 (hecho que destacamos también en el estudio preliminar) y el aparente estancamiento final, por lo que es preferible primar el número de subproblemas frente al número de generaciones (en valores razonables) como sugiere también el comportamiento del espaciado, que para el segundo y el tercero caso se comporta de manera más uniforme, para ambos con valores entre 0.01 y 0.03

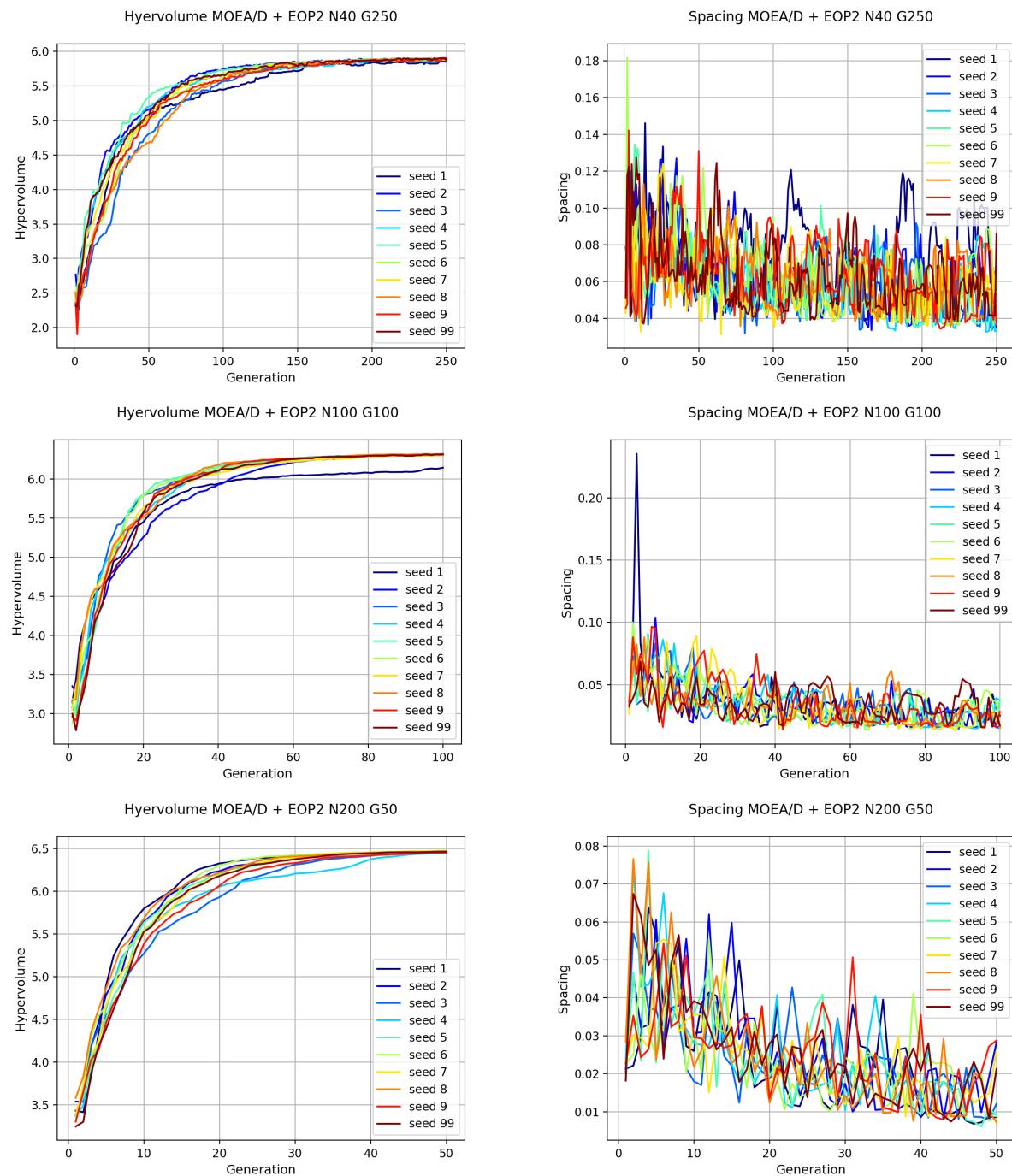
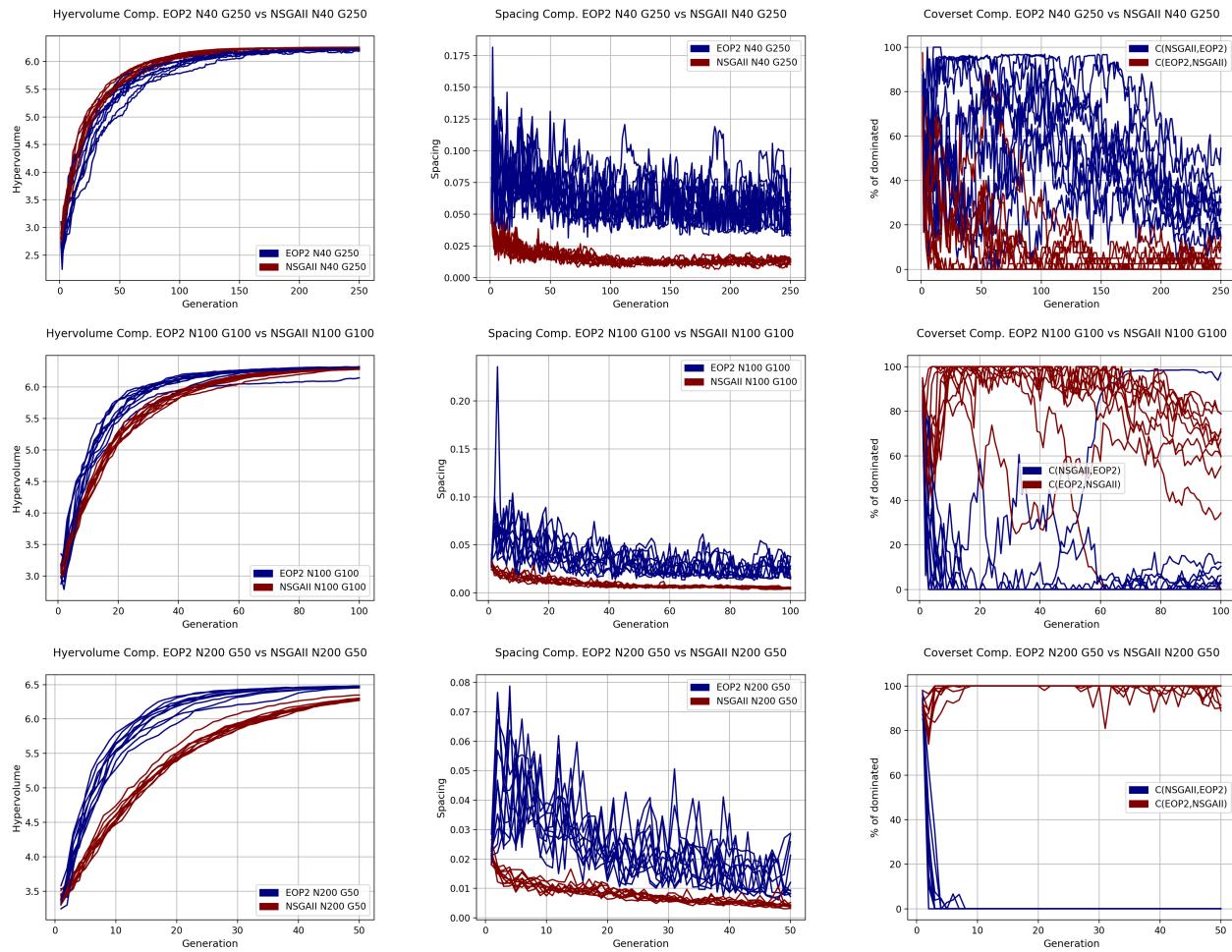


Figura 14: MOEA/D + EOP2. Métricas para 10000 EV



**Figura 15:** MOEA/D + EOP2. Comparación de métricas con NSGAII para 10000 EV.

A continuación presentaremos algunas gráficas comparativas del comportamiento del algoritmo frente a *NSGAII*. En la figura 15 se muestran las graficas con dichas comparativas en las que podemos notar que en todos los casos el espaciado en el algoritmo *NSGAII* se comporta mejor que en algoritmo propuesto mientras que el hipervolumen para el caso ( $N = 40, G = 250$ ) es favorable al algoritmo *NSGAII* y favorable a nuestro algoritmo para el resto de los casos. Como vimos en el estudio preliminar el algoritmo, con los suficientes subproblemas, sí suele converger a soluciones mejores que el algoritmo *NSGAII* pero también suele tender a concentrar más las soluciones.

Sin embargo si comprobamos la métrica cover set, sí podemos ver que en el primer caso ninguno de los frentes es dominado más de un 50 % por el otro en la mayoría de las ejecuciones, pero aún así *NSGAII* tiende a quedar por debajo. Para el segundo y el tercer caso ( $N100$  y  $N200$ ) el algoritmo propuesto tiende a dominar al *NSGAII*. Más concretamente para el segundo caso en casi todas las ejecuciones el frente del algoritmo *NSGAII* quedó dominado por el de *MOEA/D + EOP2* entre el 50 % y el 80 % de sus puntos, mientras que el frente de *NSGAII* domina al *MOEA/D + EOP2* en menos de un 20 % en prácticamente todas de las ejecuciones. En el último caso, los resultados sí son claros, el frente de *NSGAII*

queda dominado por el de nuestro algoritmo en más de un 90 % en todas las ejecuciones mientras que el caso contrario no es apreciable para ningún punto de ninguna de las ejecuciones (en más de 1 % o 2 %). De forma que queda patente los resultados que ya comentamos para el operador EOP1 pero que en este caso se manifiestan con mayor relevancia, la cantidad de subproblemas es mucho más beneficiosa que la cantidad de generaciones.

Aunque en este apartado no hemos realizado (explícitamente) un análisis de las soluciones (población final y NSD) en el último apartado de esta sección presentaremos una comparativa conjunta de las últimas generaciones y de los conjuntos no dominados (NSD) para todos los casos y todos los algoritmos.

## EXPERIMENTACIÓN PARA 4000 EVALUACIONES

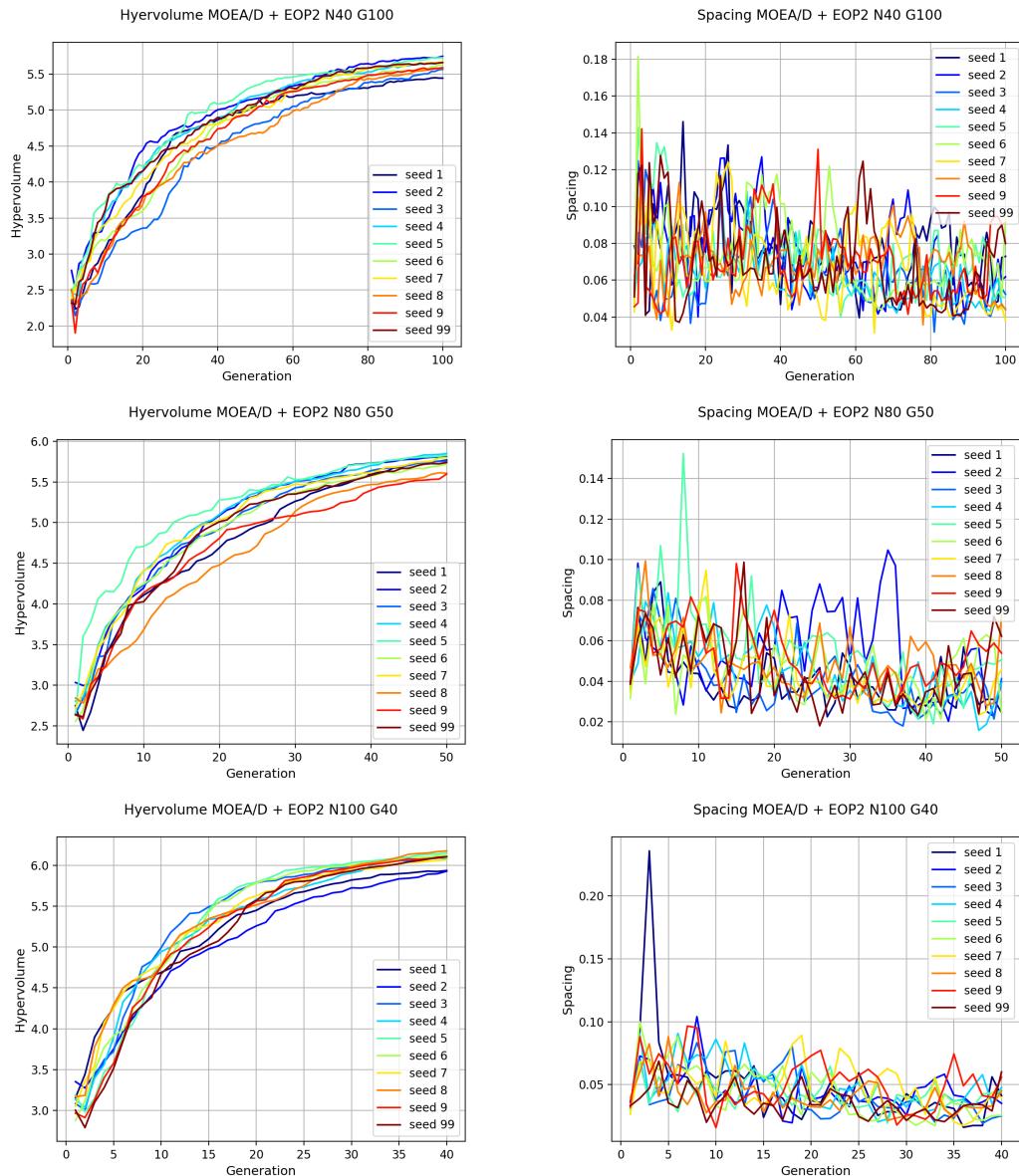
Ya presentamos para 10000 evaluaciones el comportamiento del algoritmo es adecuado, esto es, a través de las generaciones los individuos se aproximan al frente. No vamos a volver a presentarlo para el caso de 4000 evaluaciones, pues sigue lógicamente el mismo esquema (en los apéndices se presentan las gráficas que atestiguan lo expuesto). Por tanto pasaremos directamente a evaluar las métricas y a razonar directamente sobre los resultados obtenidos en dicho análisis.

En la figura 16 se presentan las gráficas del desarrollo del hipervolumen y el espaciado para cada uno de los casos considerados en las 4000 evaluaciones en contra (N = 40, G = 100), (N = 80, G = 50), (N = 100, G = 40). Como podemos apreciar el comportamiento es similar a los presentados para las 10000 evaluaciones, pero sin embargo es notable el espacio de mejora del algoritmo dado el estado de crecimiento de la curva en todos los casos, lo que parece indicar que más iteraciones y más subproblemas sí serían útiles para la mejora del algoritmo, al contrario que en EOP1 no parece haberse alcanzado el estancamiento. También es notable la gran variabilidad del espaciado dada la falta de subproblemas y generaciones, disminuyendo dicha variabilidad sensiblemente con el aumento de los subproblemas, sobre todo en el tercer caso.

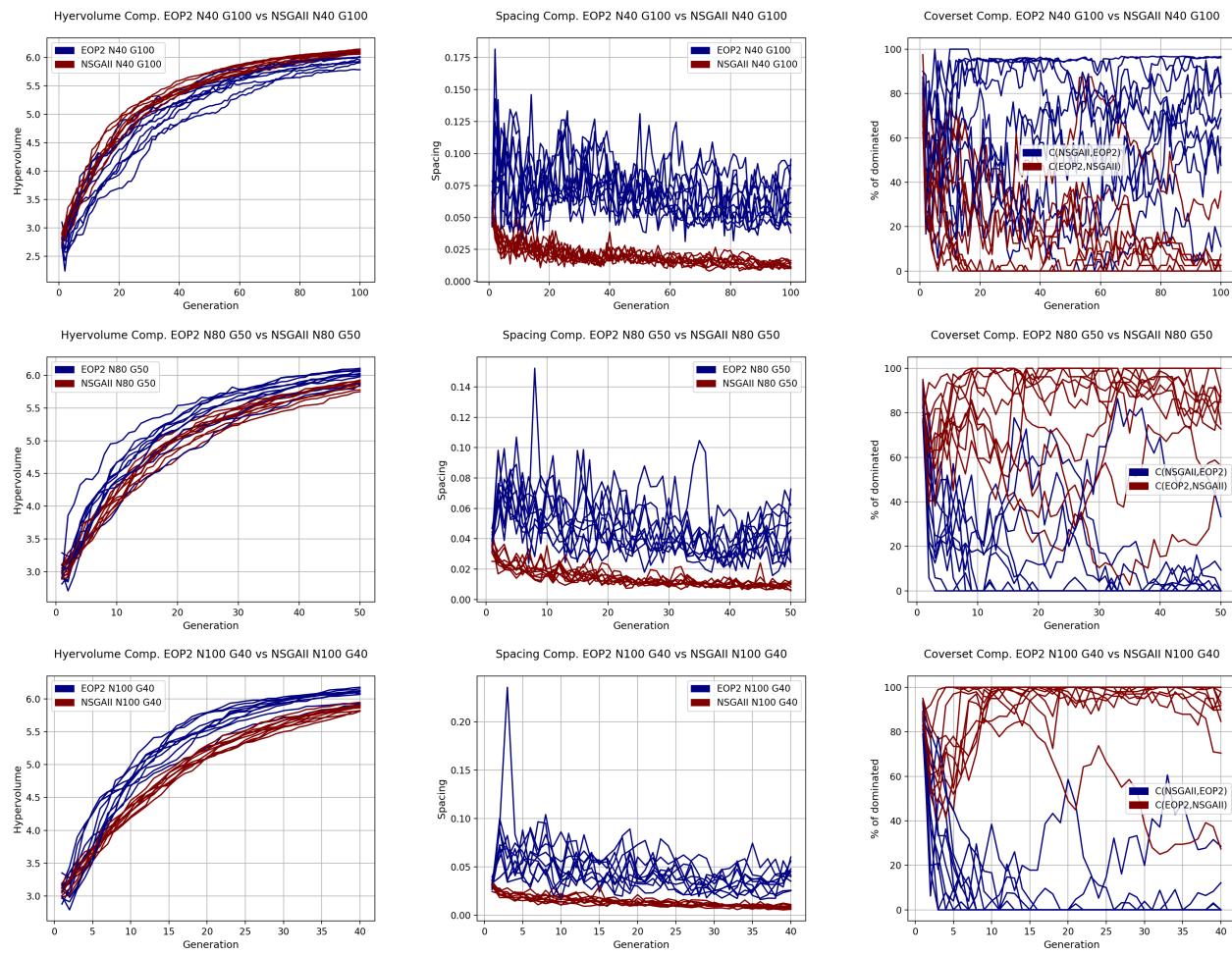
Si realizamos una comparativa (al igual que en el caso de 10000 ev) con el algoritmo *NSGAII* obtenemos conclusiones sensiblemente distintas a las presentadas para 10000 evaluaciones y para EOP1, no para el espaciado en el que *NSGAII* supera (valor más bajo) a nuestro algoritmo, aunque sí se denota una notable mejora con el aumento de los subproblemas. Pero en el caso del hipervolumen en el primer caso (N = 40, G = 100) *NSGAII* supera a nuestro algoritmo en prácticamente la totalidad de las ejecuciones, mientras que en el segundo y tercer caso nuestro algoritmo consigue superarlo en este aspecto. En cuanto al cover set, para el caso N = 40 más del 50 % del frente (final) de nuestro algoritmo es dominado por el de *NSGAII* en prácticamente todas las ejecuciones, mientras que apenas un 8 % del frente de nuestro algoritmo domina al competidor. En los otros dos casos el cover set es mucho más claro de forma que para ambos casos en buena parte de las ejecuciones más del 75 % del frente es dominado por el frente final de nuestro algoritmo, mientras que para pocas ejecuciones se produce una dominancia de *NSGAII* en más de un 1 % o 2 %.

De todo ello destacamos la bondad de nuestro algoritmo en cuanto a la convergencia, no tanto para el espaciado (sobre todo si consideramos la última generación, si consideramos el frente NSD veremos que el espaciado mejora notablemente, presentado en la comparativa final de la tabla 1) y en cuanto al cover

set, con el número adecuado de subproblemas (mejor que de generaciones) nuestro algoritmo domina prácticamente en su totalidad a *NSGAII*. Sí hemos de destacar que EOP se comportaba mejor ante la falta de generaciones (esto es su convergencia era más rápida).



**Figura 16:** MOEA/D + EOP2. Métricas para 4000EV



**Figura 17:** MOEA/D + EOP2. Comparación de métricas con NSGAII para 4000 EV.

### 4.3. Evaluación MOEA/D + EOP3 con ZDT3

De igual manera a lo presentado para EOP1 y EOP2 vamos a realizar algunas pruebas para comprobar la efectividad del algoritmo con EOP3. Tanto utilizando una capacidad de cómputo de 10000 evaluaciones como una de 4000 evaluaciones (con los mismos repartos ya presentados).

#### 4.3.1. Experimentación con 10000 ev.

En la figura 18 presentamos tres gráficos pertenecientes a una ejecución característica del algoritmo con el operador evolutivo EOP3,  $N = 100$ ,  $G = 100$  y  $T = 15$  (para todos los casos se ha tomado un 15 % (recuérdese que en los anexos se muestran varias ejecuciones que validan las discusiones) el primer diagrama muestra el desarrollo de las soluciones en el espacio de objetivos a lo largo de la ejecución del algoritmo (en las distintas generaciones). En él podemos apreciar que, tal como esperábamos a medida que el algoritmo avanza (pasan las generaciones) las soluciones tienden a ir convergiendo hacia el frente

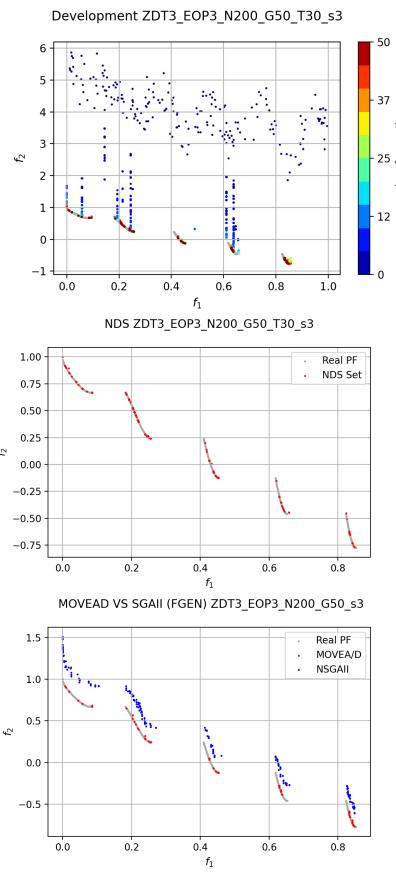
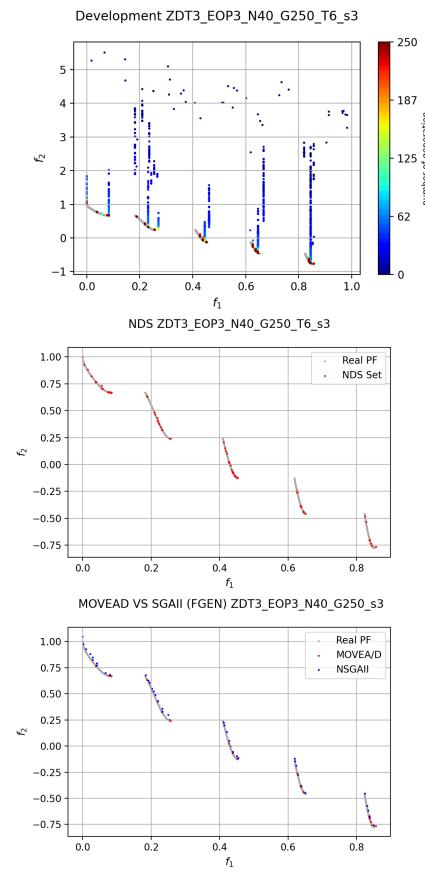
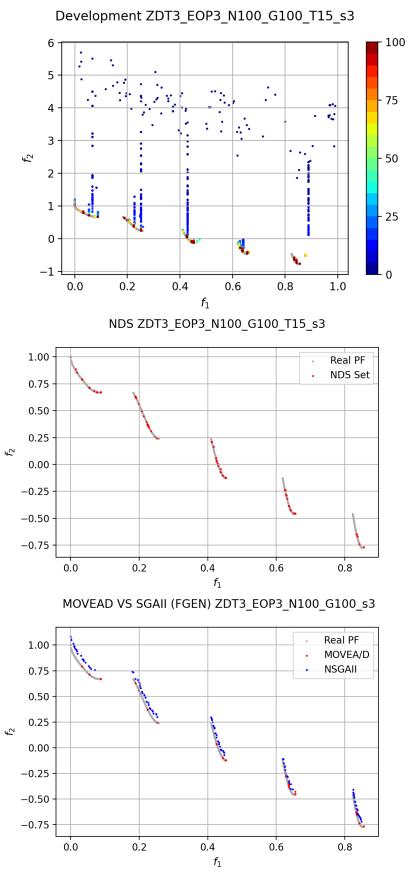
real de Pareto (señalado en gris), de forma muy rápida. Otro aspecto a destacar es la diversidad, en los algoritmos evolutivos es corriente que las soluciones tiendan a concentrarse (se pierde diversidad); en este caso podemos ver dicha tendencia claramente, lo que produce lógicamente, que las soluciones no se distribuyan uniformemente, dándose un fenómeno habitual en algoritmos genéticos (del que hemos heredado este tercer operador) conocido como convergencia temprana (efecto indeseable).

En el segundo diagrama se presenta el conjunto de las soluciones no dominadas calculadas por el algoritmo, que parece estar bastante próximo al frente real de Pareto en cuanto a convergencia, pero sí es notable la pérdida clara de uniformidad en las soluciones frente a los resultados presentados en los operadores anteriores.

Finalmente, presentamos una gráfica comparativa para nuestro algoritmo y para el algoritmo *NSGA-II*. Para ello hemos representado las soluciones en el espacio de objetivos, las cuales corresponden a la última generación de ejecuciones en iguales condiciones ( $G = 100, N = 100$ ) para ambos algoritmos. Podemos ver que en cuanto a convergencia (proximidad al frente real) el frente del nuestro algoritmo es ligeramente mejor que el frente proporcionado por *NSGA-II* (una mayoría de los puntos rojos están por debajo de los azules), mientras que en dispersión claramente los puntos del frente de *NSGA-II* se distribuyen más uniformemente que los de nuestro algoritmo.

Veámos ahora unas gráficas análogas a las presentadas pero para el caso ( $G = 250, N = 40$ ) en la *figura 19* podemos notar que el comportamiento es similar al caso anterior dándose exactamente el mismo efecto, pero en este caso la diferencia con *NSGAII* en convergencia es menor, debido a que *NSGAII* ha mejorado en ese aspecto (beneficiándose mucho más de las iteraciones que nuestro algoritmo, al que no le aportan mucho, a partir de un valor razonable). Si miramos el frente de soluciones no dominadas, vemos una situación muy similar a la anterior donde la mayoría de los puntos están situados sobre el frente pero las soluciones no están uniformemente repartidas sino que tienden a estar bastante concentradas. En comparación con *NSGAII* la convergencia de nuestro algoritmo parece ligeramente mejor mientras que en la distribución es sensiblemente peor.

Y en último lugar comprobemos cuál es el resultado de elevar el número de subproblemas frente al número de generaciones. Dado que según hemos visto hasta ahora la convergencia suele ser rápida, y que el número de subproblemas favorecerá la diversidad, a priori el comportamiento debería ser satisfactorio. Veámos qué ocurre para el caso ( $G = 250, N = 40$ ) en las gráficas presentadas en la *figura 20*. Podemos notar que la convergencia es bastante buena pero la concentración de las soluciones y la rápida pérdida de la diversidad es patente. De hecho, si observamos el diagrama de las soluciones no dominadas podemos seguir notando que la distribución no es para nada uniforme. Si comparamos con la ejecución del *NSGA-II* en este caso nuestro algoritmo es claramente mejor en convergencia y pero no en cobertura, aunque una mayoría de los puntos (lo veremos cuando analicemos las métricas) sí parecen estar dominados por los de nuestro algoritmo.



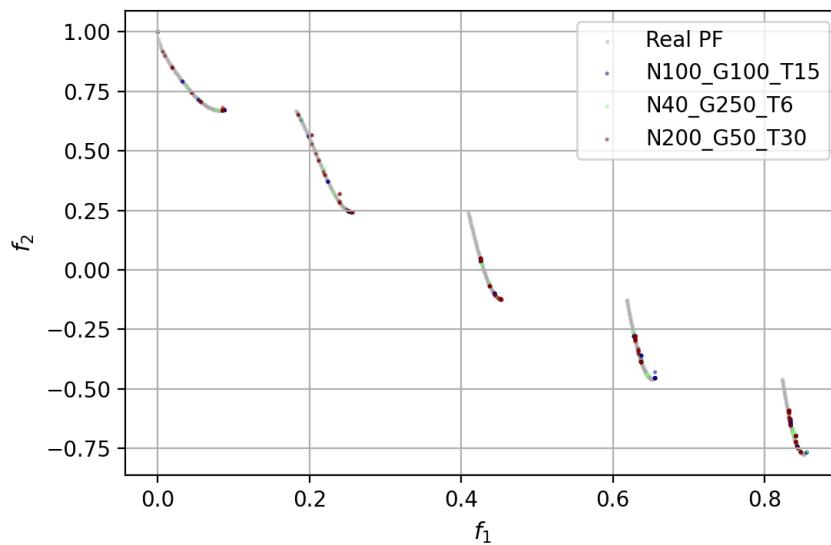
**Figura 18:** MOEA/D + EOP3  
(N100G100)

**Figura 19:** MOEA/D + EOP3  
(N40G250)

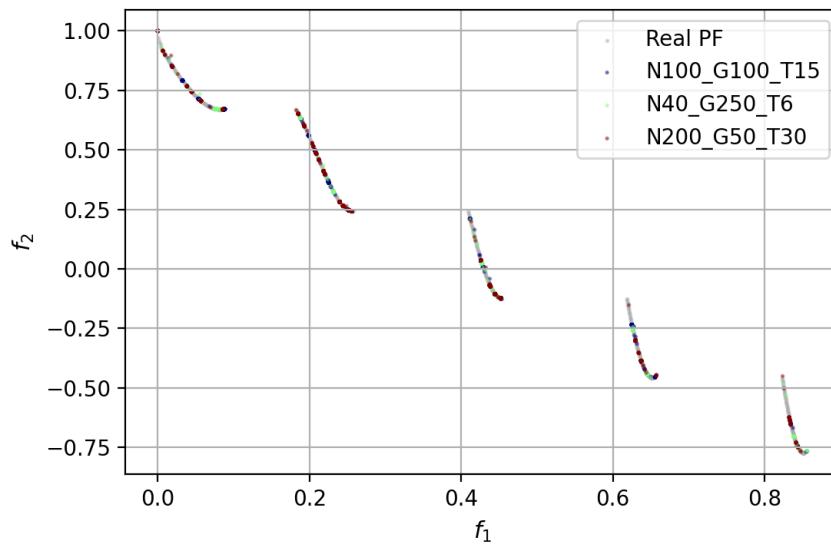
**Figura 20:** MOEA/D + EOP3  
(N200G50)

Finalmente vamos a realizar una comparativa entre los tres casos tanto para los frentes de la última generación como para los frentes NSD. En la *figura 21* se muestran las dos gráficas de comparativa de los frentes anteriormente indicados. En cuanto a las soluciones de la generación final todas se encuentran bastante superpuestas, en cuanto a convergencia todos tienen un comportamiento muy parecido, y quizás el frente rojo, correspondiente al tercer caso denote una mejor cobertura del frente. En cuanto al NSD las sensaciones son análogas y nada concluyentes, así que intentaremos clarificarlos con el uso de métricas y el correspondiente estudio estadístico de las mismas.

COMPARISION N &amp; G (Final Gen.) s3



COMPARISION N &amp; G (NSD) s3

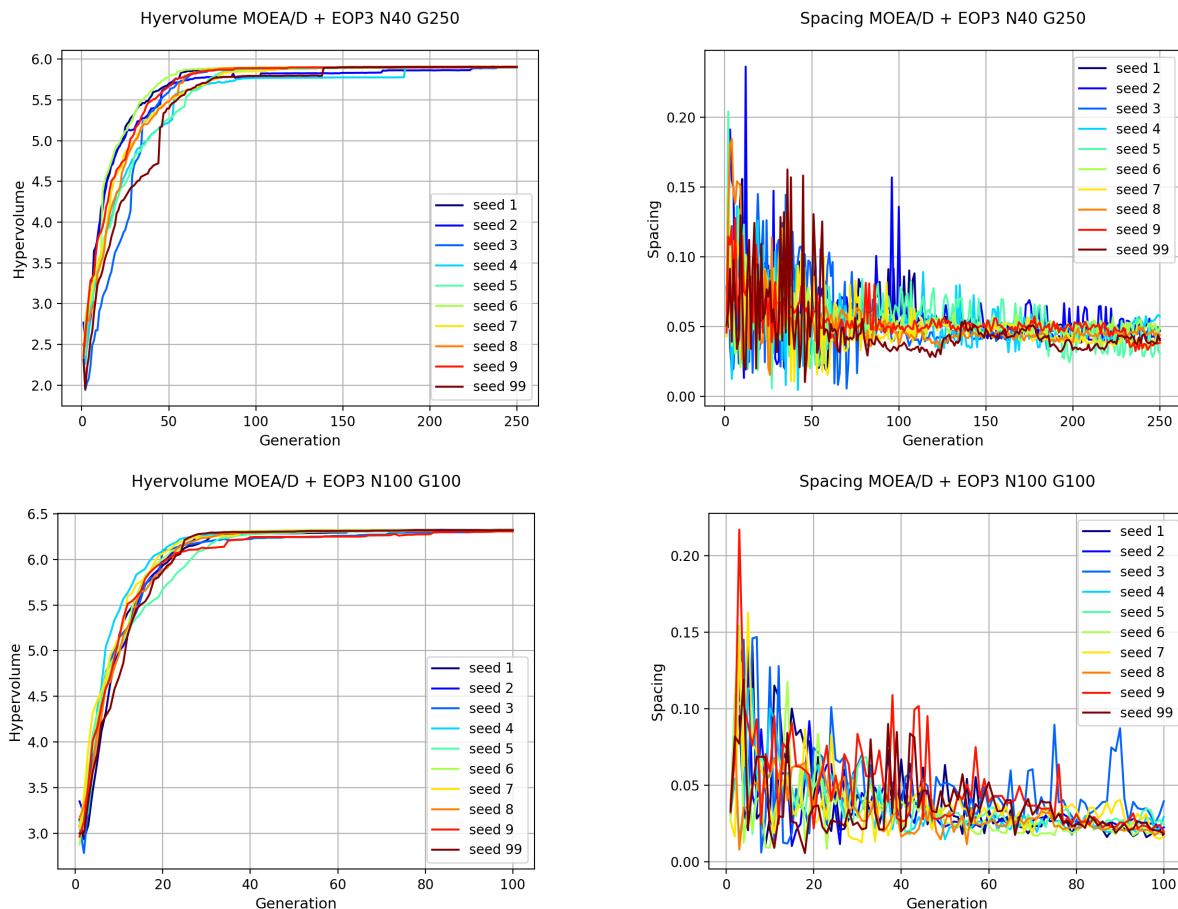

**Figura 21:** MOEA/D + EOP3. Comparación de casos

#### 4.3.2. Análisis de métricas para 10000 ev.

Presentado el estudio preliminar anterior vamos a tratar de profundizar para exclarecer y llevar a cabo una discusión más profunda de los casos anteriores. Para ello realizaremos un estudio de algunas métricas para los casos ya presentados. Tales métricas corresponden al hipervolumen, espacio y cover set.

Comencemos viendo algunas gráficas asociadas a las métricas para los casos planteados en el apartado anterior. En la figura 22 se muestra la evolución de hipervolumen y el espaciado en durante las distintas generaciones para 10 ejecuciones del algoritmo. Podemos observar que en todos los casos tanto el desarrollo del hipervolumen como el de el espaciado se comporta de forma bastante uniforme para todas las ejecuciones.

Aunque no es posible hacer un análisis conjunto del hipervolumen (ya que el punto de referencia para su cálculo es distinto en cada caso) sí podemos destacar la rápida convergencia del algoritmo, beneficiándose también del número de subproblemas más que de el número de generaciones observando una tendencia de estancamiento a partir de en torno al 40 % de las generaciones, en los tres casos. Observando además el beneficio del incremento del número de subproblemas en el espaciado, cuyo valor suele tender a disminuir con dicho aumento, de forma que parece claro que es preferible dedicar (de forma razonable) más parte de la potencia computacional al número de subproblemas en vez de al número de generaciones.



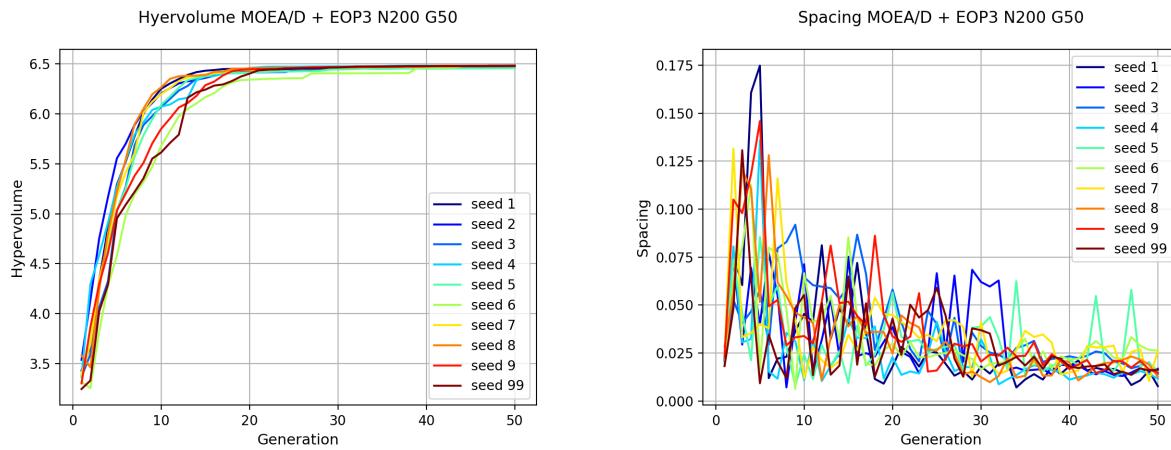


Figura 22: MOEA/D + EOP3. Métricas para 10000 EV

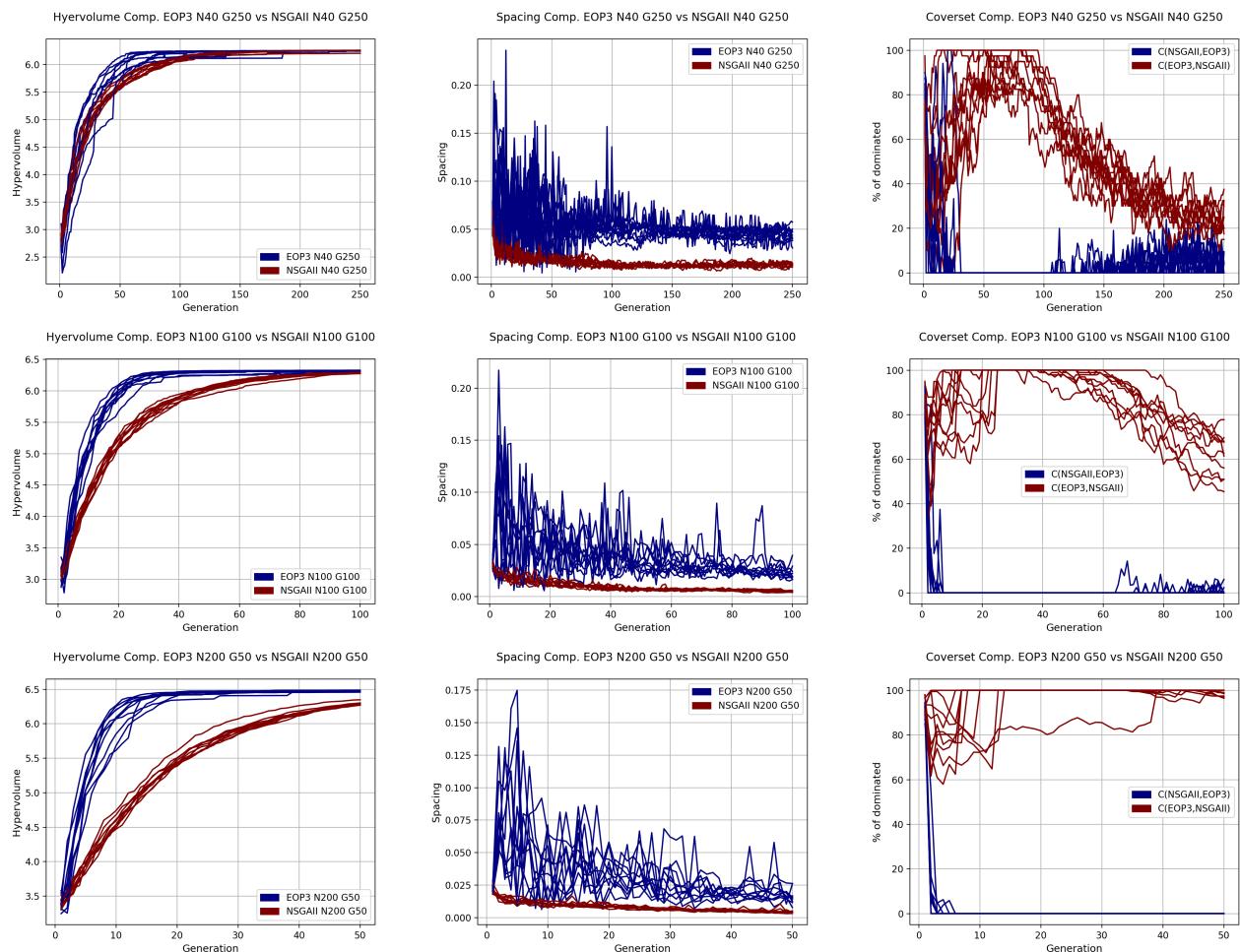


Figura 23: MOEA/D + EOP3. Comparación de métricas con NSGAII para 10000 EV.



A continuación presentaremos algunas gráficas comparativas del comportamiento del algoritmo frente a *NSGAII*. En la figura 23 se muestran las graficas con dichas comparativas en las que podemos notar que en todos los casos el espaciado en el algoritmo *NSGAII* se comporta mejor que en algoritmo propuesto mientras que el hipervolumen el comportamiento es el contrario de forma muy notable para los casos con más subproblemas. Como vimos en el estudio preliminar el algoritmo, con los suficientes subproblemas, sí suele converger a soluciones mejores que el algoritmo *NSGAII* pero también suele tender a concentrar más las soluciones.

Sin embargo si comprobamos la métrica cover set, sí podemos ver que en el primer caso ninguno de los frentes es dominado más de un 40 % por el otro en la mayoría de las ejecuciones, pero aún así nuestro algoritmo queda en todos los casos por debajo, no superando para ninguna de las ejecuciones el 17 % de dominancia. Para el segundo y tercer caso la situación es mucho clara, de forma que en casi todas las ejecuciones el frente del algoritmo *NSGAII* quedó dominado por el de *MOEA/D + EOP2* entre el 50 % y el 80 % de sus puntos, mientras que el frente de *NSGAII* domina al *MOEA/D + EOP2* en menos de un 10 % en todas de las ejecuciones y para el tercer caso, el frente de *NSGAII* queda dominado por el de nuestro algoritmo en más de un 95 % en todas las ejecuciones mientras que el caso contrario no es apreciable para ningún punto de ninguna de las ejecuciones (en más de 1 % o 2 %). De forma que queda patente los resultados que ya comentamos para el operador *EOP1* pero que en este caso se manifiestan con mayor relevancia, la cantidad de subproblemas es mucho más beneficiosa que la cantidad de generaciones.

Aunque en este apartado no hemos realizado (explícitamente) un análisis de las soluciones (población final y NSD) en el último apartado de esta sección presentaremos una comparativa conjunta de las últimas generaciones y de los conjuntos no dominados (NSD) para todos los casos y todos los algoritmos.

## EXPERIMENTACIÓN PARA 4000 EVALUACIONES

Ya presentamos para 10000 evaluaciones el comportamiento del algoritmo es adecuado, esto es, a través de las generaciones los individuos se aproximan al frente. No vamos a volver a presentarlo para el caso de 4000 evaluaciones, pues sigue lógicamente el mismo esquema (en los apéndices se presentan las gráficas que atestiguan lo expuesto). Por tanto pasaremos directamente a evaluar las métricas y a razonar directamente sobre los resultados obtenidos en dicho análisis.

En la figura 24 se presentan las gráficas del desarrollo del hipervolumen y el espaciado para cada uno de los casos considerados en las 4000 evaluaciones en contra (N = 40, G = 100), (N = 80, G = 50), (N = 100, G = 40). Como podemos apreciar el comportamiento es similar a los presentados para las 10000 evaluaciones, de hecho se empieza a atisvar la mencionada llegada a la zona de estanco y sí es notable la gran variabilidad del espaciado dada la falta de subproblemas y generaciones, disminuyendo dicha variabilidad ligeramente con el aumento de los subproblemas..

Si realizamos una comparativa (al igual que en el caso de 10000 ev) con el algoritmo *NSGAII* obtenemos conclusiones análogas a las presentadas para 10000 evaluaciones, de forma que en el espaciado *NSGAII* supera (valor más bajo) a nuestro algoritmo, aunque sí se denota una mejora con el aumento de los subproblemas. En el caso del hipervolumen nuestro algoritmo supera a *NSGAII* en prácticamente la totalidad de las ejecuciones. En cuanto al cover set, para el caso N = 40 más del 50 % del frente (final)

de *NSGAII* es dominado por nuestro algoritmo en todas las ejecuciones, mientras que apenas un no se aprecia (ni un 1% o 2%) del caso contrario. En los otros casos la situación es más clara dominando nuestro algoritmo más del 90 % del frente de *NSGAII* en todas las ejecuciones y siendo inapreciable el caso opuesto

De todo ello destacamos la bondad de nuestro algoritmo en cuanto a la convergencia, no para el espaciado y en cuanto al cover set, con el número adecuado de subproblemas (mejor que de generaciones) nuestro algoritmo domina prácticamente en su totalidad a *NSGAII*. Si destacamos frente a EOP1 y EOP2 la severa pérdida de diversidad que lo lleva a un reparto poco uniforme de las soluciones a lo largo del frente.

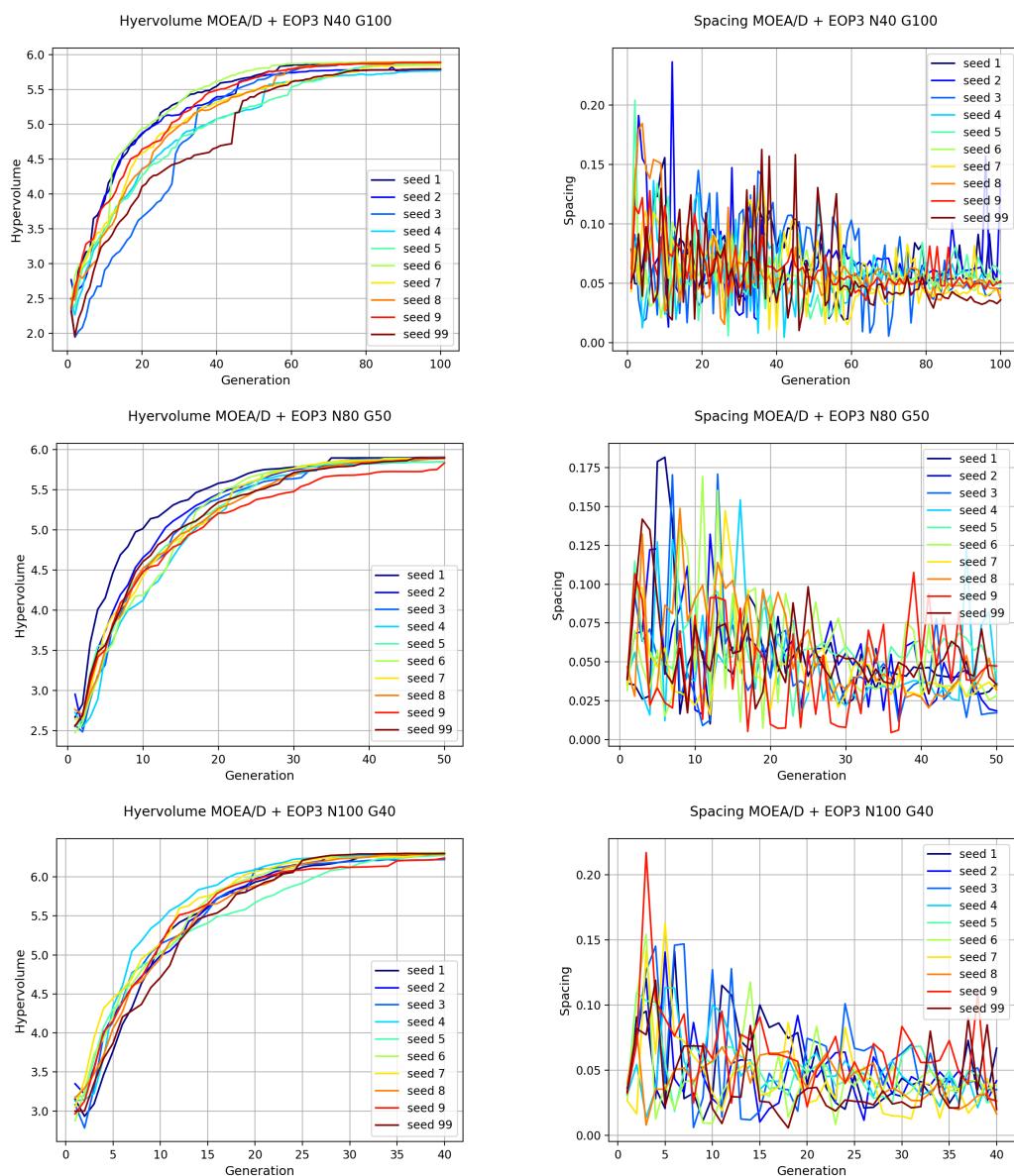


Figura 24: MOEA/D + EOP3. Métricas para 4000EV

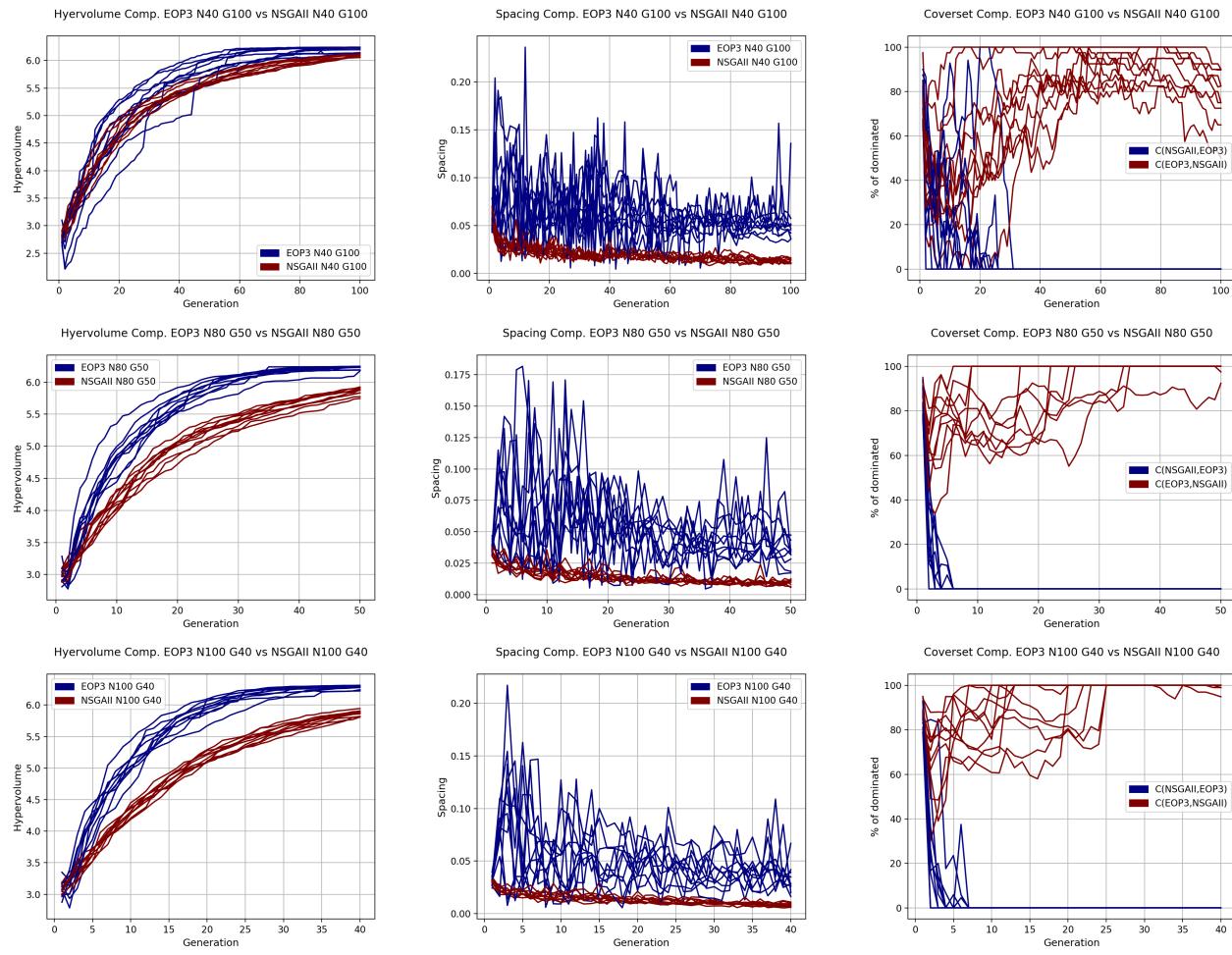


Figura 25: MOEA/D + EOP3. Comparación de métricas con NSGAII para 4000 EV.

#### 4.4. Comparativa final de los algoritmos

Para terminar el análisis vamos a presentar un análisis estadístico de las métricas de hipervolumen, espaciado y coverset sobre las soluciones proporcionadas por los distintos algoritmos, que acrediten (o desmientan) los resultados que venimos presentando hasta el momento. Comencemos presentando los resultados para las generaciones finales (incluyendo a *NSGAII*) y después presentaremos también los resultados dados por NDS. Para cada una de las tablas se ha tomado el mismo punto de referencia para la evaluación del hipervolumen, de forma que los resultados aquí presentados son totalmente comparables entre sí).

En la tabla 1 se presentan los resultados de la comparativa de las métricas de hipervolumen y espaciado para 4000 EV. Podemos ver en la misma que el algoritmo *MOEA/D EOP1* (con  $N = 100, G = 40$ ), aunque en el espaciado como venimos presentando el mejor es *NSGAII*. Sin embargo, es mucho más considerable la ganancia hipervolumen de nuestro algoritmo frente a la pérdida de espaciado. Tanto es así que si acudimos a la tabla 5 vemos que en cuanto al cover set, vemos que el frente de nuestro algoritmo domina

al 100 % del frente de *NSGAII* en el 100 % de las ejecuciones. Por lo tanto podemos considerar que bajo 4000 evaluaciones nuestro algoritmo es mejor que *NSGAII* y además que, como ya hemos resaltado, es mucho más conveniente primar el número de subproblemas frente al número de generaciones. Además los resultados de las tablas refuerzan todos los análisis que hemos venido presentando tanto para *EOP1*, *EOP2* y *EOP3*.

Si pasamos ahora a evaluar estas mismas métricas sobre 10000 evaluaciones, en la tabla 2 se presentan los resultados de la comparativa de las métricas de hipervolumen y espaciado y en la tabla 6 el cover set, podemos ver reflejada una situación análoga a la presentada para 4000EV. En el que *EOP1* vuelve a ser el mejor en cuanto a hipervolumen y *NSGAII* en cuanto a espaciado, repitiéndose también la dominancia de nuestro algoritmo sobre el competidor en más del 99 % del frente casi la totalidad de las ejecuciones y dándose el caso contrario para el 0 % del frente en la totalidad de las ejecuciones, por lo que nuestro algoritmo vence también a *NSGAII* con un mayor poder computacional y siempre mejorando con el mayor uso un mayor número de subproblemas (aunque en este caso los casos *N100* y *N200* son prácticamente equivalentes).

Finalmente vamos a comentar algunos resultados acerca de las soluciones dada por los frentes NDS. En la tabla 2 se presentan los resultados de la comparativa de las métricas de hipervolumen y espaciado y en la tabla 6 el cover set para los casos con 4000EV. Podemos observar que en este aspecto *EOP1* vuelve a ser el mejor valorado frente a *EOP2* y *EOP3* tanto en hipervolumen como en espaciado. En hipervolumen si es notable que el operador *EOP3* es prácticamente equivalente a *EOP1* pero en cuanto al espaciado *EOP1* es claramente vencedor, logrando los mejores valores para el caso con 100 subproblemas, logrando valores comparables en espaciado a los presentados para la generación final de *NSGAII*. En cuanto al cover set, vemos que es clara la dominancia de *EOP1* y *EOP3* sobre *EOP2*, pero entre ellos es menos clara (a pesar de que *EOP3* tiende a dominar a *EOP1* en más del 40 % del frente). En el caso de 10000EV, los resultados son similares a los presentados para 40000EV, aunque disminuyendo la dominancia en coverset de *EOP3* a *EOP1*.

## 5. Conclusiones

Después de los resultados presentados de la experimentación realizada, es destacable la mejora de nuestro algoritmo frente a *NSGAII* especialmente en el caso de *EOP1*. Destacando, además, la rápida convergencia de nuestro algoritmo comprobando complementariamente el beneficio del aumento del número de subproblemas tanto para la convergencia, como por supuesto para el espaciado, de forma que a un poder computacional de 4000EV el algoritmo con mejor comportamiento corresponde a *MOEA/D + EOP1* con una población de 100 individuos y 40 generaciones mientras que para 10000EV el mejor resultado se alcanzó con ese mismo algoritmo y con 200 individuos y 50 generaciones. Además los otros dos operadores evaluados demostraron un comportamiento aceptable, sobre todo en las 10000 evaluaciones, concretamente *EOP3* quien (aunque con falta de diversidad) consiguió resultados muy buenos. Por último destacar el beneficio del NDS, en cuanto a la cobertura del frente, sin descompensar la convergencia, que proporcionó, sobre todo para *EOP1* una aproximación muy cercana al frente real de Pareto.

ALGORITHM	N	G	Hv_mean	Hv_std	Spc_mean	Spc_std
MOEA/D EOP1	40	100	2.088514	0.015041	0.038675	0.001963
MOEA/D EOP1	80	50	2.107404	0.009019	0.020931	0.002098
MOEA/D EOP1	100	40	<b>2.103889</b>	<b>0.004384</b>	0.016192	0.002490
MOEA/D EOP2	40	100	1.899492	0.059507	0.063062	0.019576
MOEA/D EOP2	80	50	1.929259	0.064898	0.045417	0.014444
MOEA/D EOP2	100	40	1.938250	0.070247	0.042124	0.010837
MOEA/D EOP3	40	100	2.106688	0.034493	0.055301	0.027549
MOEA/D EOP3	80	50	2.121596	0.023985	0.032382	0.009011
MOEA/D EOP3	100	40	2.119973	0.010458	0.032826	0.013621
NSGAII	40	100	2.012664	0.023211	0.012071	0.002019
NSGAII	80	50	1.790951	0.047534	0.009881	0.002221
NSGAII	100	40	1.745769	0.036844	<b>0.008174</b>	<b>0.001572</b>

Tabla 1: ZDT3- Comparativa completa Hipervol. y Spacing (FGEN, 4000EV)

ALGORITHM	N	G	Hv_mean	Hv_std	Spc_mean	Spc_std
MOEA/D EOP1	40	250	1.684281	0.001819	0.037795	0.001208
MOEA/D EOP1	100	100	1.693973	0.001051	0.015474	0.000347
MOEA/D EOP1	200	50	<b>1.694149</b>	<b>0.002867</b>	0.008268	0.000891
MOEA/D EOP2	40	250	1.660451	0.008558	0.052216	0.015527
MOEA/D EOP2	100	100	1.660273	0.047921	0.022429	0.008742
MOEA/D EOP2	200	50	1.675938	0.006607	0.015855	0.008601
MOEA/D EOP3	40	250	1.682167	0.002680	0.043343	0.007077
MOEA/D EOP3	100	100	1.688566	0.002852	0.022407	0.006816
MOEA/D EOP3	200	50	1.686791	0.005307	0.016195	0.005595
NSGAII	40	250	1.685467	0.003161	0.013168	0.001379
NSGAII	100	100	1.656455	0.006912	0.004888	0.000580
NSGAII	200	50	1.522302	0.017832	<b>0.004105</b>	<b>0.000507</b>

Tabla 2: ZDT3- Comparativa completa Hipervol. y Spacing (FGEN, 10000EV)

ALGORITHM	N	G	Hv_mean	Hv_std	Spc_mean	Spc_std
MOEA/D EOP1	40	100	1.451592	0.030917	0.027312	0.017374
MOEA/D EOP1	80	50	<b>1.497310</b>	<b>0.007318</b>	0.009470	0.002279
MOEA/D EOP1	100	40	1.496890	0.004449	<b>0.008987</b>	<b>0.001489</b>
MOEA/D EOP2	40	100	1.302208	0.055153	0.024994	0.009511
MOEA/D EOP2	80	50	1.334811	0.064589	0.022065	0.007143
MOEA/D EOP2	100	40	1.344495	0.072631	0.021256	0.011216
MOEA/D EOP3	40	100	1.449614	0.083014	0.098708	0.068756
MOEA/D EOP3	80	50	1.496368	0.043114	0.079722	0.058070
MOEA/D EOP3	100	40	1.498124	0.029528	0.062702	0.025272

**Tabla 3:** ZDT3- Comparativa completa Hipervol. y Spacing (NDS, 4000EV)

ALGORITHM	N	G	Hv_mean	Hv_std	Spc_mean	Spc_std
MOEA/D EOP1	40	250	1.235404	0.248486	0.014032	0.016374
MOEA/D EOP1	100	100	1.259206	0.266413	0.004887	0.003128
MOEA/D EOP1	200	50	<b>1.263475</b>	<b>0.271422</b>	<b>0.003192</b>	<b>0.000780</b>
MOEA/D EOP2	40	250	1.223784	0.249435	0.017460	0.015928
MOEA/D EOP2	100	100	1.232368	0.273584	0.006281	0.001798
MOEA/D EOP2	200	50	1.247745	0.273041	0.005406	0.000924
MOEA/D EOP3	40	250	1.213916	0.233524	0.064273	0.069826
MOEA/D EOP3	100	100	1.236160	0.251089	0.055110	0.052205
MOEA/D EOP3	200	50	1.250131	0.262513	0.042286	0.036711

**Tabla 4:** ZDT3- Comparativa completa Hipervol. y Spacing (NDS, 10000EV)

ALG1	ALG2	N	G	c(ALG2, ALG1)		c(ALG1, ALG2)	
				Mean	Std	Mean	Std
EOP1	EOP2	40	100	0.000000	0.000000	0.974490	1.110223e-16
EOP1	EOP2	80	50	0.000000	0.000000	0.974490	1.110223e-16
EOP1	EOP2	100	40	0.000000	0.000000	0.974490	1.110223e-16
EOP1	EOP3	40	100	0.000000	0.000000	0.974490	1.110223e-16
EOP1	EOP3	80	50	0.000000	0.000000	0.974490	1.110223e-16
EOP1	EOP3	100	40	0.000000	0.000000	0.974490	1.110223e-16
EOP1	NSGAII	40	100	0.008655	0.018388	0.769936	1.416819e-01
EOP1	NSGAII	80	50	0.000000	0.000000	1.000000	0.000000e+00
EOP1	NSGAII	100	40	0.000000	0.000000	1.000000	0.000000e+00
EOP2	EOP3	40	100	0.000000	0.000000	1.000000	0.000000e+00
EOP2	EOP3	80	50	0.000000	0.000000	1.000000	0.000000e+00
EOP2	EOP3	100	40	0.000000	0.000000	1.000000	0.000000e+00
EOP2	NSGAII	40	100	0.696330	0.245986	0.050897	1.056916e-01
EOP2	NSGAII	80	50	0.042708	0.100827	0.794910	1.437527e-01
EOP2	NSGAII	100	40	0.040767	0.089379	0.842722	2.059881e-01
EOP3	NSGAII	40	100	0.000000	0.000000	0.772244	1.208985e-01
EOP3	NSGAII	80	50	0.000000	0.000000	0.989873	2.320291e-02
EOP3	NSGAII	100	40	0.000000	0.000000	0.992815	1.531966e-02

Tabla 5: ZDT3- Comparativa completa Cover Set (FGEN, 4000EV)

ALG1	ALG2	N	G	c(ALG2, ALG1)		c(ALG1, ALG2)	
				Mean	Std	Mean	Std
EOP1	EOP2	40	250	0.000000	0.000000	0.989899	1.110223e-16
EOP1	EOP2	100	100	0.000000	0.000000	0.989899	1.110223e-16
EOP1	EOP2	200	50	0.000000	0.000000	0.989899	1.110223e-16
EOP1	EOP3	40	250	0.000000	0.000000	0.989899	1.110223e-16
EOP1	EOP3	100	100	0.000000	0.000000	0.989899	1.110223e-16
EOP1	EOP3	200	50	0.000000	0.000000	0.989899	1.110223e-16
EOP1	NSGAII	40	250	0.053730	0.054245	0.220000	7.483315e-02
EOP1	NSGAII	100	100	0.000000	0.000000	0.907915	3.822231e-02
EOP1	NSGAII	200	50	0.000000	0.000000	0.992347	2.295918e-02
EOP2	EOP3	40	250	0.000000	0.000000	1.000000	0.000000e+00
EOP2	EOP3	100	100	0.000000	0.000000	1.000000	0.000000e+00
EOP2	EOP3	200	50	0.000000	0.000000	1.000000	0.000000e+00
EOP2	NSGAII	40	250	0.260153	0.118778	0.040000	4.636809e-02
EOP2	NSGAII	100	100	0.136465	0.281695	0.576990	2.292387e-01
EOP2	NSGAII	200	50	0.000000	0.000000	0.965804	4.469600e-02
EOP3	NSGAII	40	250	0.040089	0.032455	0.255000	7.053368e-02
EOP3	NSGAII	100	100	0.008386	0.018737	0.610813	9.586192e-02
EOP3	NSGAII	200	50	0.000000	0.000000	0.987302	1.287349e-02

**Tabla 6:** ZDT3- Comparativa completa Cover Set (FGEN, 10000EV)

ALG1	ALG2	N	G	c(ALG2, ALG1)		c(ALG1, ALG2)	
				Mean	Std	Mean	Std
EOP1	EOP2	40	100	0.000000	0.000000	0.995505	0.008991
EOP1	EOP2	80	50	0.000000	0.000000	0.997826	0.006522
EOP1	EOP2	100	40	0.002469	0.007407	0.993750	0.018750
EOP1	EOP3	40	100	0.491354	0.096355	0.048016	0.060577
EOP1	EOP3	80	50	0.454795	0.087867	0.027795	0.028894
EOP1	EOP3	100	40	0.414962	0.051668	0.049302	0.048648
EOP2	EOP3	40	100	0.915397	0.097093	0.003846	0.011538
EOP2	EOP3	80	50	0.921391	0.085503	0.000000	0.000000
EOP2	EOP3	100	40	0.842845	0.108491	0.005556	0.016667

**Tabla 7:** ZDT3- Comparativa completa Cover Set (NDS, 4000EV)

ALG1	ALG2	N	G	c(ALG2, ALG1)		c(ALG1, ALG2)	
				Mean	Std	Mean	Std
EOP1	EOP2	40	250	0.024549	0.020495	0.873728	0.080861
EOP1	EOP2	100	100	0.018487	0.022481	0.910285	0.068369
EOP1	EOP2	200	50	0.020209	0.019274	0.877289	0.088478
EOP1	EOP3	40	250	0.247131	0.056417	0.131976	0.067172
EOP1	EOP3	100	100	0.247053	0.049638	0.084933	0.036117
EOP1	EOP3	200	50	0.290046	0.064070	0.082858	0.032555
EOP2	EOP3	40	250	0.541899	0.074144	0.032363	0.023784
EOP2	EOP3	100	100	0.599404	0.152908	0.019685	0.016970
EOP2	EOP3	200	50	0.603076	0.116967	0.025308	0.022203

**Tabla 8:** ZDT3- Comparativa completa Cover Set (NDS, 10000EV)

## Referencias

- [1] H. Li and Q. Zhang, “Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii,” *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.
- [2] Q. Zhang, W. Liu, and H. Li, “The performance of a new version of moea/d on cec09 unconstrained mop test instances,” in *2009 IEEE Congress on Evolutionary Computation, CEC 2009*, 2009, pp. 203–208, 2009 IEEE Congress on Evolutionary Computation, CEC 2009 ; Conference date: 18-05-2009 Through 21-05-2009.
- [3] Q. Zhang and H. Li, “Moea/d: A multiobjective evolutionary algorithm based on decomposition,” *Evolutionary Computation, IEEE Transactions on*, vol. 11, pp. 712 – 731, 01 2008.
- [4] Y. Li, A. Zhou, and G. Zhang, “An moea/d with multiple differential evolution mutation operators,” 07 2014, pp. 397–404.
- [5] B. Liu, F. V. Fernandez, Q. Zhang, M. Pak, S. Sipahi, and G. Gielen, “An enhanced moea/d-de and its application to multiobjective analog cell sizing,” 08 2010, pp. 1 – 7.
- [6] “ETH - SOP - Downloads/Material - Supplementary Material - Testproblems - ZDT 3.” [Online]. Available: <https://sop.tik.ee.ethz.ch/download/supplementary/testproblems/zdt3/index.php>
- [7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: Nsga-ii,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.