Optimización de Enjambre de Partículas para Problemas de Muchos Objetivos

Mateo Torres
Universidad Católica "Nuestra Señora de la Asunción"
Asunción, Paraguay
Email: torresmateo@gmail.com

Benjamín Barán
Universidad Católica "Nuestra Señora de la Asunción"
Universidad Nacional del Este
Email: bbaran@pol.una.py

***Abstract*—The difficulty to solve problems with many, possibly conflicting, objectives logically increases with the number of objectives, what makes them difficult to solve using multi-objective algorithms like the well known NSGA-II. Therefore, this work proposes the use of a particle swarm optimization (PSO) algorithm to solve many-objective problems. The main premise of this work is that MOPSO may be a good option for solving many-objective problems, presenting experimental evidence that supports this premise using the well known DTLZ benchmark with different performance metrics such as hypervolume, coverage and generational distance, among others.**

***Resumen*—La dificultad de resolver problemas considerando muchos objetivos (*many-objective*) posiblemente contradictorios, lógicamente crece con el numero de objetivos, lo que dificulta su solución con algoritmos multiobjetivo reconocidos como el NSGA-II. En consecuencia, este trabajo propone utilizar una optimización por enjambre de partículas (PSO) para resolver problemas de muchos objetivos. La premisa principal del trabajo es que un MOPSO puede ser una buena opción para este tipo de problemas, aportando evidencia experimental que soporta esta premisa utilizando los reconocidos problemas de prueba DTLZ con diferentes métricas de comparación como hipervolúmen, cobertura y distancia generacional, entre otras.**

I. Introducción

En los problemas multiobjetivo (MOPs - *Multi-Objective Problems*), existen por lo general conflictos entre los diversos objetivos considerados, lo que usualmente impide obtener una única solución óptima y en cambio se encuentra un conjunto de soluciones de compromiso llamadas conjunto Pareto óptimo. En este contexto, se dice que una solución domina a otra si no es peor en ningún objetivo, y es estrictamente mejor en al menos uno de los objetivos [1], [2], [3].

En los últimos años, ha quedado establecido que la dificultad de resolver un problema aumenta para problemas *many-objective*; es decir, problemas con cuatro o más objetivos [1], [4], [5], [6], [7], [8], [9]. En el caso de algoritmos basados en dominancia Pareto, dicha dificultad está intrínsecamente relacionada al hecho que a medida que el número de objetivos aumenta, la proporción de conjuntos de soluciones no dominadas crece, haciendo que sea cada vez más difícil encontrar buenas soluciones utilizando únicamente la relación de dominancia [3], [5]. Además, la mayoría de los algoritmos evolutivos se basan en estructuras de datos que crecen exponencialmente en complejidad a medida que se incrementa la cantidad de objetivos [4], [10].

Una vez que un algoritmo encuentra un conjunto Pareto, se asume que el tomador de decisiones selecciona una solución de este conjunto. Esta toma de decisiones no será tratada en este trabajo, el cual se limitará al cálculo del conjunto Pareto. En esta etapa, la visualización de las alternativas se vuelve importante. Aunque se han propuesto varios métodos [11], [12], [13], [14], [15], todavía falta una manera simple e intuitiva de representar soluciones en un espacio objetivo con cuatro o más objetivos [1].

Se puede apreciar que al aumentar el número de objetivos, se dan los siguientes fenómenos:

- los algoritmos basados en dominancia Pareto no proveen la presión de selección requerida hacia mejores soluciones, dificultando una búsqueda evolutiva eficiente e incluso el elitismo puede ser difícil [1], [3];
- dado un conjunto de soluciones aleatorias, la proporción e de soluciones no comparables es:

$$e = \frac{2^M - 2}{2^M} \tag{1}$$

  A medida que la cantidad de objetivos M crece, la proporción e tiende a uno. Este fenómeno deja ver que la dominancia Pareto puede resultar inadecuada para seleccionar buenas soluciones en problemas *many-objective* [1], [5].

Este trabajo muestra pruebas analíticas y experimentales de la complejidad temporal de dos algoritmos multiobjetivo: el reconocido NSGA-II (*Non-dominated Sorting Genetic Algorithm II*) [16] y la propuesta alternativa de este trabajo, el MOPSO (*Multi-Objective Particle Swarm Optimization*) [10]. La propuesta principal del trabajo es que el algoritmo MOPSO puede ser una buena opción para problemas *many-objective*.

En las siguientes secciones se introducen conceptos importantes para el entendimiento de los problemas de muchos objetivos. Luego se describen a dos algoritmos multiobjetivo, el NSGA-II [16] y el MOPSO [10], [17] junto con un análisis de complejidad de estos algoritmos. Seguidamente se muestra el conjunto de problemas de prueba y las definiciones de las métricas de calidad utilizadas. Luego, se especifica el método utilizado para realizar los experimentos que validan dicho análisis y se presentan los resultados experimentales correspondientes. Estos resultados experimentales son utilizados para comparar los algoritmos NSGA-II y MOPSO utilizando diversas métricas de calidad. Finalmente, se presentan las conclusiones de este trabajo y las propuestas de trabajos

978-1-4673-9143-6/15/$31.00

futuros.

II. Problemas Multiobjetivo

Los problemas que se tratan en este trabajo son problemas multiobjetivo. Muchas veces éstos problemas tienen objetivos conflictivos, razón por la cual no es trivial resolverlos. Cuando se tiene más de un objetivo se vuelve usual la necesidad de elegir entre un conjunto de soluciones no comparables, debido a que puede no existir una única solución óptima del problema.

II-A. Definición

Definición 1: Problema Multi-Objetivo [1]: Sea F un conjunto de M funciones objetivo $\{f_1, f_2, \cdots, f_M\}, f_i : \mathbb{R}^n \Rightarrow \mathbb{R}$, se define un Problema Multi-Objetivo (MOP) como:

$$\begin{aligned} \text{Optimizar } \textit{(Max/Min)}\ y = F(x) = (f_1(x), f_2(x), \cdots, f_M(x)) \\ x = (x_1, x_2, \cdots, x_n) \in \mathcal{X} \subseteq \mathbb{R}^n \\ y = (y_1, y_2, \cdots, y_M) \in \mathcal{Y} \subseteq \mathbb{R}^M \end{aligned} \tag{2}$$

sujeto a

$$x_i^{(L)} \leq x_i \leq x_i^{(U)}\ \forall\ i \in \{1, 2, \cdots, n\} \tag{3}$$

$$r(x) = (r_1(x), r_2(x), \cdots, r_l(x)) \leq 0 \tag{4}$$

donde x es un vector con n variables de decisión, mientras que y representa un vector objetivo M-dimensional. Las restricciones (3) y (4) representan límites que ayudan a definir el espacio factible de decisión o espacio de búsqueda $\mathcal{X}$. Las funciones objetivo constituyen un espacio multidimensional llamado "espacio objetivo" representado por el símbolo $\mathcal{Y}$. El vector r está compuesto por l funciones (4) que restringen el espacio de decisión. Las soluciones que no cumplen con las funciones (4) y/o límites de rango de las variables (3) son llamadas "soluciones no factibles", mientras que las soluciones que cumplen con todas las restricciones (3) y (4) se conocen como "soluciones factibles". El conjunto de todas las soluciones factibles $\mathcal{X}_f$ es conocido como el "espacio de decisión factible". El dominio de cada f_i es $\mathcal{X}_f$. Para cada solución $x \in \mathcal{X}_f$ existe un punto y en el espacio objetivo. La imagen de $\mathcal{X}_f$ define el "espacio objetivo factible" $\mathcal{Y}_f$

$$\mathcal{Y}_f = F(\mathcal{X}_f) = \bigcup_{x \in \mathcal{X}_f} \{F(x)\} \tag{5}$$

Como fuera antes expresado, un problema con $M \geq 4$ se conoce como *many-objective problem* (MaOP) [18], en caso contrario, se denomina simplemente MOP.

II-B. Dominancia Pareto

En un conjunto de soluciones factibles $\mathcal{X}_f$, dadas 2 soluciones $u, v \in \mathcal{X}_f$, se dice que u domina a v (expresado como $u \succ v$) si no es peor en ningún objetivo y es estrictamente mejor en al menos un objetivo [1], [3].

Definición 2: Conjunto Pareto Óptimo: Para un MOP dado, el Conjunto Pareto Óptimo, expresado como $\mathcal{P}^*$, se define como el conjunto de todas las soluciones factibles no dominadas:

$$\mathcal{P}^* = \{x \in \mathcal{X}_f | \nexists\ x' \in \mathcal{X}_f \text{ tal que } x' \succ x\}$$

```
input : P      // P es un conjunto de
                  soluciones conocido
                  como Población
foreach p ∈ P do
    S_p = ∅
    n_p = 0
    foreach q ∈ P do
        if p ≻ q then          // Si p domina a q
            S_p = S_p ∪ {q}// Agregar q al set
                           de soluciones
                           dominadas por p
        else if q ≻ p then
            n_p = n_p + 1 // Sumar 1 al
                          contador de
                          dominación de p
    if n_p = 0 then // p pertence al primer
                    frente
        p_rank = 1
        F_1 = F_1 ∪ {p}
i = 1
while F_i ≠ ∅ do
    Q = ∅ // utilizado para establecer
          el siguiente frente
    foreach p ∈ F_i do
        foreach q ∈ S_p do
            n_q = n_q − 1
            if n_q = 0 then // q pertence al
                            siguiente frente
                q_rank = i + 1
                Q = Q ∪ {q}
    i = i + 1
    F_i = Q
Retornar F     // F es el conjunto de
               todos los frentes F_i
```

Algoritmo 1: Fast Nondominated Sort utilizado por el algoritmo NSGA-II

Definición 3: Frente Pareto Óptimo: Dado un MOP, el Frente Pareto Óptimo, expresado como $\mathcal{PF}^*$, se define como la imagen en el espacio objetivo del Conjunto Pareto Óptimo $\mathcal{P}^*$:

$$\mathcal{PF}^* = \{y = F(x) \in \mathcal{Y}_f | x \in \mathcal{P}^*\}$$

III. *NSGA-II*

El principal algoritmo de referencia para resolver problemas multiobjetivo de la literatura actual es el NSGA-II [16]. Este algoritmo propone un procedimiento para clasificar a los individuos de la población en varios frentes no dominados, llamado *fast nondominated sorting procedure* que se desarrolla de la siguiente manera: primeramente se determina para cada individuo i de una población P la cantidad de individuos que lo dominan n_i y el conjunto D_i de los individuos dominados por i. Todo individuo i cuyo $n_i = 0$ pertenece al primer frente $\mathcal{F}_1$. Para todos los elementos de los conjuntos D_i de los individuos del primer frente, el valor n_i se decrementa en uno. Aquellos individuos cuyo n_i se vuelve cero luego de un decremento, forman parte del segundo frente $\mathcal{F}_2$. Este proceso es repetido hasta clasificar todos los elementos de la población en sus respectivos frentes $\mathcal{F}_i$. En el Algoritmo 1 se puede ver el pseudocódigo de este procedimiento.

Además, a fin de mejorar la diversidad, se propone un valor que representa la densidad de la población cercana a una solución del espacio objetivo sin la necesidad de establecer un parámetro extra al algoritmo. Este valor se denomina *crowding distance* (d_i). Se define entonces un operador para comparar

input : $\mathcal{I}$ // $\mathcal{I}$ es un conjunto de soluciones de un mismo frente $\mathcal{F}_i$
$l = |\mathcal{I}|$ // cardinalidad del conjunto
foreach $i \in \mathcal{I}$ **do**
  $\mathcal{I}_{i_{\text{distance}}} = 0$ // inicializar las distancias
foreach objective $m \in M$ **do**
  $\mathcal{I} = \text{sort}(\mathcal{I}, m)$ // ordenar utilizando cada objetivo
  $\mathcal{I}_{1_{\text{distance}}} = \mathcal{I}_{l_{\text{distance}}} = \infty$ // crowding de extremos
  for $i = 2$ **to** $(l-1)$ **do**
    $\mathcal{I}_{i_{\text{distance}}} = \mathcal{I}_{i_{\text{distance}}} + \frac{\mathcal{I}_{i+1}.m - \mathcal{I}_{i-1}.m}{f_m^{\max} - f_m^{\min}}$
Retornar $\mathcal{I}$ con distancias asignadas

Algoritmo 2: *Crowding Distance Assignment* o asignación de valores de densidad. La notación $\mathcal{I}_i.m$ representa al componente m del vector de espacio de decisión asociado a la i-ésima solución del conjunto $\mathcal{I}$

individuos llamado *crowded comparison operator* ($\prec_n$) que define un orden parcial entre los individuos y se define de la siguiente manera:

$$i \prec_n j \iff (\mathcal{F}_i < \mathcal{F}_j) \vee (\mathcal{F}_i = \mathcal{F}_j \wedge d_i > d_j) \qquad (6)$$

donde $\mathcal{F}_i$ representa el frente al cual pertenece un individuo i. El pseudocódigo para el cálculo del *crowding distance* de cada individuo para un frente dado se muestra en el Algoritmo 2. Dado un frente $\mathcal{F}_i$, se inicializan las distancias a 0 para todos los elementos del frente. Luego, se ordena el frente para cada objetivo y se asigna una distancia infinita a las soluciones de los extremos, mientras que a las demás soluciones se asigna un promedio del cuboide que encierra a esta solución sin incluir a otra solución de su frente.

El bucle principal del NSGA-II para M objetivos se muestra en el Algoritmo 3. Se combina la población padre con la población hija de la generación anterior en un conjunto R_t (Población combinada en la iteración t). Luego, dicho conjunto es separado en frentes no dominados utilizando el método `fast-nondominated-sort` (Algoritmo 1) en el conjunto de frentes $\mathcal{F}$. Se selecciona el conjunto de las N mejores soluciones (N es un parámetro que limita el tamaño de la población), asignando el valor d_i a las soluciones de cada frente, utilizando el operador $\prec_n$ para seleccionar los elementos del último frente necesario para completar el conjunto padre P_{t+1}. Dicho conjunto es luego sometido a los operadores usuales de los algoritmos genéticos (selección, cruzamiento, y mutación) para generar una población descendiente Q_{t+1}. Este proceso se repite hasta que se cumpla la condición de parada.

Teniendo en cuenta el estudio de complejidad realizado por los autores [16], la complejidad de una iteración es la siguiente:

1. *fast-non-dominated sorting*: complejidad temporal $M(2N)^2$ por los bucles anidados utilizados para dividir a la población en frentes no dominados y la comparación de las soluciones (Algoritmo 1).
2. *crowding distance assignment*: complejidad temporal $M(2N)\log(2N)$ ya que cada frente debe ordenarse considerando cada objetivo (M veces) y al estar incluido en el loop hace que la suma de los frentes sea $2N$.
3. *ordenar usando* $\prec_n$: complejidad temporal $2N\log(2N)$. Dada por el algoritmo utilizado para ordenar (ejemplo: *quicksort*).

input : N, M
$P = \texttt{generar\_poblacion}(N, M)$
$t = 1$
while *not* `condicion_de_parada()` **do**
  $R_t = P_t \cup Q_t$ // combinar la población padre con la población descendiente
  $\mathcal{F} = \texttt{fast-non-dominated-sort}(R_t)$
  // $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \cdots)$ (Algoritmo 1)
  $P_{t+1} = \emptyset$
  $i = 1$
  while $|P_{t+1}| + |\mathcal{F}_i| \leq N$ **do**
    `crowding-distance-assignment`$(\mathcal{F}_i)$
    // Algoritmo 2
    $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$
    $i = i + 1$
  sort$(\mathcal{F}_i, \prec_n)$ // ordenar de forma descendiente usando $\prec_n$
  $P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$ // elegir los primeros $N - |P_{t+1}|$ elmentos de $\mathcal{F}_i$
  $Q_{t+1} = \texttt{make-new-pop}(P_{t+1})$ $t = t + 1$
Retornar la última población generada P_t

Algoritmo 3: NSGA-II

La complejidad final del algoritmo NSGA-II es así $M(2N)^2$, dada por la separación en frentes no dominados, como se ilustra en el Algoritmo 1.

IV. *MOPSO*

IV-A. Optimización por Enjambre de Partículas

La metaheurística de Optimización por Enjambre de partículas (PSO - *Particle Swarm Optimization*), desarrollada por Kennedy y Eberhart en 1995 [19], se inspira en el comportamiento social de individuos como peces, aves y abejas, que realizan exploración de una zona en grupo, detectando alimentos o guiando al conjunto (cardumen, parvada, enjambre) de manera colectiva a posiciones ventajosas. La metaheurística consiste básicamente en un algoritmo iterativo basado en una población de individuos llamada "enjambre", en la que cada individuo denominado "partícula", se dice que sobrevuela el espacio de decisión en busca de soluciones óptimas.

En un espacio de búsqueda n-dimensional, cada partícula i del enjambre conoce su posición actual $x_i = [x_{i_1}, x_{i_2}, \cdots, x_{i_n}]$, su velocidad $v_i = [v_{i_1}, v_{i_2}, \cdots, v_{i_n}]$ (con la cual ha llegado a dicha posición) y la mejor posición $p_i = [p_{i_1}, p_{i_2}, \cdots, p_{i_n}]$ en la que ha estado, denominada "mejor posición personal". Además, todas las partículas conocen la

mejor posición de entre todas las mejores posiciones personales en el enjambre $g = [g_1, g_2, \cdots, g_n]$, a la cual se denomina "mejor posición global".

En cada iteración t del algoritmo, cada componente j de la posición y la velocidad de cada partícula i del enjambre se actualiza conforme a las siguientes ecuaciones [20]:

$$v_{ij}^{t+1} = \omega \times v_{ij}^{t} + C_1 \times \textit{rand()} \times (p_{ij}^{t} - x_{ij}^{t}) + C_2 \times \textit{rand()} \times (g_{ij}^{t} - x_{ij}^{t}) \quad (7)$$

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1} \quad (8)$$

donde ω es el parámetro de inercia, C_1 es un parámetro cognitivo, C_2 es un parámetro social mientras que *rand()* es una función que retorna un número aleatorio en el intervalo $[0, 1]$.

La ecuación (7) calcula un nuevo vector de velocidad para la i-ésima partícula a partir de su velocidad actual, la distancia euclidiana a su mejor posición personal, y la distancia euclidiana a la mejor posición global.

En la ecuación (8) las componentes del vector de posición de la i-ésima partícula se actualizan de acuerdo con cada componente de su nueva velocidad. El rol de las distancias a las mejores posiciones (personal y global) en la ecuación (7) es influir en que la partícula sea "atraída" tanto hacia la mejor posición en la que se ha encontrado, como a la posición más favorable encontrada por el resto del enjambre. Al uso de estas distancias se denomina respectivamente "factor cognitivo" y "factor social". Los parámetros C_1 y C_2 son constantes de aceleración [19], cuyos valores determinan la influencia máxima de los factores en el cálculo de la nueva velocidad. El uso de la función *rand()* sirve para determinar una influencia aleatoria de los factores, siendo su efecto mejorar la exploración.

El parámetro de inercia ω introducido por Eberhart y Shi [21] es utilizado para controlar el impacto de las velocidades anteriores en el cálculo de la nueva velocidad. ω controla el balance entre la exploración global y personal del espacio de búsqueda [22]. Valores grandes para este parámetro favorecen la exploración de áreas nuevas del espacio de búsqueda (*exploración*) pues la partícula no es tan influenciada por los mejores (global y local), mientras que valores pequeños del parámetro fomentan una búsqueda en zonas cercanas a los mejores (local y global), lo que implica una mayor *explotación*. Es usual limitar el crecimiento de la velocidad estableciendo un valor máximo $v_{\max}$ [22], y en ese caso luego de calcular la velocidad, se restringe a esta conforme a la ecuación (9).

$$\begin{aligned} &\text{si } v_{ij} > v_{\max} \text{ entonces } v_{ij} = v_{\max} \\ &\text{sino si } v_{ij} < -v_{\max} \text{ entonces } v_{ij} = -v_{\max} \end{aligned} \quad (9)$$

IV-B. Optimización por Enjambre de Partículas en Problemas de Optimización multiobjetivo

Adaptar el algoritmo PSO para realizar una optimización multiobjetivo requiere un nuevo cálculo de velocidad. Dicho cálculo debe tener en cuenta que la mejor posición global debe ser necesariamente una solución no dominada de compromiso encontrada por el enjambre [20]. De manera similar, la mejor solución local debe ser una solución de compromiso encontrada por la partícula. Por brevedad, en esta sección solo se muestra una de las dos variantes del algoritmo PSO adaptado a optimizaciones multiobjetivo (MOPSO). Otra variante no utilizada fue descartada por su mayor complejidad exponencial al aumentar la cantidad de objetivos, como puede verse en [23].

```
input : ω, C1, C2, v_max, N
for i = 1 to N do
    Inicializar la i-ésima partícula en una velocidad y posición aleatoria
    evaluar(i)
    Actualizar el repositorio global (G)
    Actualizar el repositorio personal de la i-ésima partícula (P_i)
while not condicion_de_parada() do
    for i = 1 to N do
        Seleccionar mejor posición personal (elemento aleatorio en P_i)
        Seleccionar mejor posición global (elemento aleatorio en G)
        Calcular velocidad de la i-ésima partícula // ecuación (7)
        Limitar las componentes de la velocidad a v_max // ecuación (9)
        Calcular la nueva posición de la i-ésima partícula // ecuación (8)
    for i = 1 to N do
        evaluar(i)
        Actualizar el repositorio global (G)
        Actualizar el repositorio personal de la i-ésima partícula (P_i)
Retornar soluciones_no_dominadas(G)
```

Algoritmo 4: MOPSO Moore y Chapman

1) Moore y Chapman (1999): Moore y Chapman [10] proponen un método en el cual cada partícula almacena en un repositorio personal todas las soluciones no dominadas que ha encontrado la partícula, así como un repositorio para todas las soluciones no dominadas encontradas por el enjambre. Para el cálculo de la nueva velocidad, cada partícula toma como mejor posición personal a cualquier miembro del repositorio de mejores posiciones personales, y como mejor posición global, a cualquier miembro del repositorio de mejores posiciones globales. El Algoritmo 4 muestra el pseudocódigo de la propuesta MOPSO de Moore y Chapman.

La complejidad del algoritmo se da principalmente por el segundo bucle que recorre toda la población (de tamaño N), ya que la complejidad de mantener los repositorios de las partículas es de MN (dado por la comparación de pares de soluciones en el repositorio para M objetivos). Por lo tanto, la complejidad de una iteración es de MN^2.

V. Problemas de Prueba Utilizados

Los problemas de prueba utilizados para los experimentos de este trabajo pertenecen a un conjunto reconocido de problemas (*benchmarks*) que son escalables en la cantidad de objetivos, conocidos como problemas DTLZ (por sus autores Deb, Thiele, Laumanns y Zitzler) [24]. Este conjunto propone diversos problemas tipo. En esta sección se muestran las definiciones de cada problema, explicando sus particularidades.

En este trabajo fueron implementados 5 problemas de los 9 propuestos por los autores originales teniendo en cuenta que con estos 5 problemas se cubren las diversas dificultades que se encuentran en la colección de problemas DTLZ [24].

V-A. DTLZ1

$$\left.\begin{array}{ll} \text{Min} & f_1(\mathbf{x}) = \frac{1}{2}x_1x_2x\cdots x_{M-1}\big(1+g(\mathbf{x}_M)\big), \\ \text{Min} & f_2(\mathbf{x}) = \frac{1}{2}x_1x_2x\cdots(1-x_{M-1})\big(1+g(\mathbf{x}_M)\big), \\ \vdots & \vdots \\ \text{Min} & f_{M-1}(\mathbf{x}) = \frac{1}{2}x_1(1-x_2)\big(1+g(\mathbf{x}_M)\big) \\ \text{Min} & f_M(\mathbf{x}) = \frac{1}{2}(1-x_1)\big(1+g(\mathbf{x}_M)\big) \end{array}\right\} \quad \text{sujeto a } 0 \le x_i \le 1, \text{para } i = 1,2,\cdots,n \tag{10}$$

donde $\mathbf{x}_M = \{x_M, x_{M+1}, \cdots, x_n\}$

La función $g(\mathbf{x}_M)$ requiere $|\mathbf{x}_M| = k$ variables y debe necesariamente ser definida con una función $g(\mathbf{x}_M) \ge 0$. Se ha utilizado la función sugerida por los mismos autores [24]:

$$g(\mathbf{x}_M) = 100\left(k + \sum_{x_i \in \mathbf{x}_M} (x_i - 0{,}5)^2 - \cos\big(20\pi(x_i - 0{,}5)\big)\right) \tag{11}$$

Las soluciones Pareto óptimas cumplen con la relación $x_i^* = 0{,}5$ $(x_i^* \in \mathbf{x}_M)$ y la función objetivo está sobre el hiperplano lineal con la relación $\sum_{m=1}^{M} f_m^* = 0{,}5$. Para DTLZ1, la cantidad total de variables es $n = M + k - 1$. La dificultad del problema es converger al hiperplano óptimo. El espacio de búsqueda contiene $(11^k - 1)$ frentes Pareto óptimos locales ubicados en hiperplanos dentro del espacio de búsqueda, pero con más altura, que pueden atraer a un algoritmo evolutivo multiobjetivo (MOEA - *Multi Objectvive Evolutionary Algorithm*) [24].

El problema puede hacerse más difícil usando otras funciones multimodales g (incluso utilizando una k más grande) y/o reemplazando x_i por un mapeo no lineal $x_i = L_i(y_i)$ y utilizando las y_i como variables de decisión [24].

V-B. DTLZ2

Los problemas DTLZ2, DTLZ3 y DTLZ4 utilizan la siguiente forma genérica:

$$\left.\begin{array}{ll} \text{Min} & f_1(\mathbf{x}) = \big(1+g(\mathbf{x}_M)\big)\cos(x_1^\alpha\pi/2)\cdots\cos(x_{M-1}^\alpha\pi/2), \\ \text{Min} & f_2(\mathbf{x}) = \big(1+g(\mathbf{x}_M)\big)\cos(x_1^\alpha\pi/2)\cdots\sin(x_{M-1}^\alpha\pi/2), \\ \text{Min} & f_3(\mathbf{x}) = \big(1+g(\mathbf{x}_M)\big)\cos(x_1^\alpha\pi/2)\cdots\sin(x_{M-2}^\alpha\pi/2), \\ \vdots & \vdots \\ \text{Min} & f_M(\mathbf{x}) = \big(1+g(\mathbf{x}_M)\big)\sin(x_1^\alpha\pi/2) \end{array}\right\} \quad \text{sujeto a } 0 \le x_i \le 1, \text{para } i = 1,2,\cdots,n \tag{12}$$

En particular, para el DTLZ2 se define:

$$g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} (x_i - 0{,}5)^2 \tag{13}$$

Para el problema DTLZ2, $\alpha = 1$. Las soluciones Pareto óptimas corresponden a $x_i^* = 0{,}5$ $(x_i^* \in \mathbf{x}_M)$. Además, todas las funciones objetivo deben satisfacer $\sum_{m=1}^{M} f_m^* = 1$ y por lo tanto el frente tiene la forma de una hiperesfera M-dimensional de radio 1. Como en el caso de DTLZ1, este problema requiere de un vector de $n = M + k - 1$ variables [24].

V-C. DTLZ3

El DTLZ3 se construye utilizando las mismas definiciones dadas en (12) pero utilizando la función $g(\mathbf{x}_M)$ del DTLZ1 como se define en la ecuación (11).

Para este problema, como en el caso anterior, $\alpha = 1$. Esta función $g(\mathbf{x}_M)$ introduce $(3^k - 1)$ frentes Pareto locales y un frente Pareto óptimo global. Todos los frentes Pareto locales son paralelos al frente Pareto óptimo global y un MOEA puede estancarse en cualquiera de los frentes Pareto óptimos locales antes de converger al frente Pareto óptimo global (en $g(\mathbf{x}_M) = 0$). El frente Pareto óptimo global corresponde a $x_i^* = 0{,}5$ para $x_i^* \in \mathbf{x}_M$ [24].

V-D. DTLZ4

El DTLZ4 se construye utilizando las mismas definiciones dadas en (12) y (13) pero utilizando un valor de α distinto. Los autores sugieren un valor de $\alpha = 100$. Esta modificación permite la existencia de un conjunto denso de soluciones cerca del plano $f_M - f_1$. En este problema, la población final es dependiente de la población inicial. La dificultad del problema es que los MOEAs tienden a generar soluciones en los planos con alta densidad y no en todo el frente Pareto óptimo [24].

V-E. DTLZ5

$$\left.\begin{array}{ll} \text{Min} & f_1(\mathbf{x}) = \big(1+g(\mathbf{x}_M)\big)\cos(\theta_1\pi/2)\cdots\cos(\theta_{M-1}\pi/2), \\ \text{Min} & f_2(\mathbf{x}) = \big(1+g(\mathbf{x}_M)\big)\cos(\theta_1\pi/2)\cdots\sin(\theta_{M-1}\pi/2), \\ \text{Min} & f_3(\mathbf{x}) = \big(1+g(\mathbf{x}_M)\big)\cos(\theta_1\pi/2)\cdots\sin(\theta_{M-2}\pi/2), \\ \vdots & \vdots \\ \text{Min} & f_M(\mathbf{x}) = \big(1+g(\mathbf{x}_M)\big)\sin(\theta_1\pi/2) \end{array}\right\} \quad \text{sujeto a } 0 \le x_i \le 1, \text{para } i = 1,2,\cdots,n \tag{14}$$

con

$$\theta_i = \frac{\pi}{4\,(1+g(\mathbf{x}_M))}\,(1 + 2g(\mathbf{x}_M)x_i) \quad \text{para } i = 2,3,\cdots,(M-1) \tag{15}$$

$$g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} x_i^{0,1} \tag{16}$$

El frente Pareto óptimo corresponde a $x_i^* = 0$ para todo $x_i^* \in \mathbf{x}_M$. El tamaño del vector $\mathbf{x}_M$ sugerido es 10 [24]. La dificultad de este problema es encontrar el verdadero frente Pareto óptimo, que en este caso es una curva, lo cual lo hace más sencillo de visualizar (simplemente dibujando f_M con cualquier otro objetivo). En las pruebas de los autores, los algoritmos evolutivos mostraron una consistente tendencia a converger a una superficie, en lugar de a una curva [24].

VI. MÉTRICAS UTILIZADAS

Con el objetivo de determinar que algoritmo genera un mejor resultado, en este trabajo se utilizan las siguientes métricas de calidad:

- Generación total de vectores no dominados (ONVG - *Overall Non-dominated Vector Generation*) [25].
- Hipervolúmen (HV - *Hypervolume indicator*) [26].
- Indicador ϵ (ϵ-*Indicator*) [27].
- Cobertura (C - *Coverage*) [25].
- Cobertura Complementaria ($\overline{C}$) (propuesta del presente trabajo).
- Distancia Generacional (GD - *Generational Distance*) [28].
- Distancia Mínima (MD - *Minimum Distance*) [28].

1) Generación Total de Vectores No Dominados: La métrica de generación de vectores no dominados (ONVG - *Overall Non-dominated Vector Generation*) se utiliza para medir la cantidad de soluciones generadas en una aproximación A por un Algoritmo Evolutivo y se define como [25]:

$$ONVG = |A| \tag{17}$$

donde $|A|$ representa la cardinalidad del conjunto de soluciones calculado por el algoritmo en cuestión. En general, se prefiere a los conjuntos con el mayor valor posible de $ONVG$, lo que será adoptado en este trabajo, aunque su valor en la práctica sea acotado a un máximo que en este trabajo se tomará como 1000.

2) Hipervolúmen: Una de las métricas más populares es el hipervolúmen, también conocido como métrica S (*S-metric*) o medida de Lebesgue [26], [29].

El indicador unario del hipervolúmen es una medida de la calidad de un conjunto $P = \{p^{(1)}, p^{(2)}, \cdots, p^{(\overline{k})}\}$ de $\overline{k}$ soluciones no dominadas que son resultado de la ejecución de un algoritmo optimizador multiobjetivo [29]. Asumiendo que el MOP es de minimización y de M objetivos, el indicador consiste en la región que es dominada por P, limitada en la parte superior por un punto de referencia $p_r \in \mathbb{R}^M$ tal que $p_r \geq (\max_p p_1, \cdots, \max_p p_M)$, donde $p = (p_1, \cdots, p_M) \in P \subset \mathbb{R}^M$, y la relación $\geq$ se aplica a cada componente. La región definida es equivalente a la unión de $\overline{k}$ hiper-rectángulos alineados con un vértice común (el punto de referencia p_r) [26].

Esta métrica, a pesar de ser la preferida en la literatura especializada [1], tiene el inconveniente de su alta complejidad de cómputo:

- limite inferior conocido $\Omega(\overline{k} \log \overline{k})$ [30];
- peor caso $O(\overline{k}^{d/3} \log^{O(1)} \overline{k})$ [31];
- complejidad computacional del algoritmo recursivo $O(\overline{k}^{d-2} \log \overline{k})$ [32],

donde $\overline{k}$ es la cantidad de soluciones de la aproximación y d es la cantidad de dimensiones (usualmente equivalente a la cantidad de objetivos M del problema). Para dos conjuntos de soluciones A y B generadas por los algoritmos $\mathcal{A}$ y $\mathcal{B}$ respectivamente, se dice que $\mathcal{A}$ es mejor que $\mathcal{B}$ si su hipervolúmen es mayor, lo que se denota como $HV(A) > HV(B)$.

Esta complejidad se considera computacionalmente muy costosa [32] por lo que su uso para una cantidad grande de objetivos no es viable. Consecuentemente, en este trabajo se utiliza esta métrica solo en problemas de hasta 6 objetivos.

3) Indicador ϵ: Suponiendo un problema de minimización con M objetivos positivos $Z \subseteq \mathbb{R}^{+^M}$, se dice que un vector objetivo $a = (a_1, a_2, \cdots, a_M) \in Z$ domina en ϵ a otro vector objetivo $b = (b_1, b_2, \cdots, b_M) \in Z$, denotado como $a \succeq_\epsilon b$ si y solo si $\forall i \in 1, \cdots, M : a_i \leq \epsilon \cdot b_i$.

Dado un $\epsilon \geq 0$, se define el Indicador ϵ binario I_ϵ [27] como

$$I_\epsilon(A, B) = \inf_{\epsilon \in \mathbb{R}} \{\forall b \in B \ \exists a \in A : a \succeq_\epsilon b\} \tag{18}$$

para cualquier par de aproximaciones A, B.

De forma simple, a domina en ϵ a b ($a \succeq_\epsilon b$) si se puede multiplicar cada componente en b por ϵ y el vector resultante sigue siendo dominado por a. Por lo tanto, el indicador ϵ es el factor por el cual una aproximación es peor que otra teniendo en cuenta todos los objetivos, o más precisamente: $I_\epsilon(A, B)$ es igual al factor mínimo ϵ para el cual cualquier solución en B es dominada en ϵ por al menos un vector de A [27]. Se prefiere el menor valor de ϵ por lo que si $I_\epsilon(A, B) < I_\epsilon(B, A)$ se dice que $\mathcal{A}$ es mejor.

4) Cobertura: En [25] se define la métrica cobertura (*coverage*) como:

$$C(A, B) = \frac{|\{B_i | \exists A_j, A_j \succ B_i\}|}{|B|} \tag{19}$$

donde A y B son las aproximaciones propuestas por los algoritmos $\mathcal{A}$ y $\mathcal{B}$ respectivamente, $|B|$ representa la cardinalidad del conjunto B. Expresado con más sencillez, la cobertura es la proporción de soluciones en B dominadas por al menos un elemento en A. Un valor de 1 indica que todas las soluciones en B son dominadas por las soluciones existentes en A, mientras un valor de 0 indica que ninguna solución de B es dominada por soluciones en A, por lo que si $C(A, B) < C(B, A)$ se dice que $\mathcal{A}$ es mejor.

5) Cobertura Complementaria: Similar a la métrica anterior, el presente trabajo propone definir a la cobertura complementaria $\overline{C}$ como:

$$\overline{C}(A, B) = \frac{|\{B_i | \exists A_j, B_i \succ A_j\}|}{|B|} \tag{20}$$

donde $|B|$ representa la cardinalidad del conjunto B. Expresado con más sencillez, la cobertura complementaria $\overline{C}$ es la proporción de soluciones en B que dominan a al menos un elemento de A. Un valor de 1 indica que todas las soluciones en B dominan a al menos una solución en A, mientras un valor de 0 indica que no existen soluciones en B que dominen a soluciones en A, por lo que si $\overline{C}(A, B) > \overline{C}(B, A)$ se dice que $\mathcal{A}$ es mejor.

6) Distancia Generacional: La métrica de esta sección y la siguiente difieren de las anteriores en el hecho de que requieren que el frente Pareto del MOP sea conocido, lo que no es

un inconveniente con el *benchmark* DTLZ utilizado en este trabajo.

La distancia generacional representa que tan lejos se encuentra una aproximación A del frente Pareto óptimo $\mathcal{PF}^*$ y se define como [28]:

$$GD = \frac{\sqrt{\sum_{i=1}^{|A|} d_i^{\text{mín}}}}{|A|} \tag{21}$$

donde $d_i^{\text{mín}}$ es la distancia Euclidiana entre cada vector objetivo en A y su correspondiente más cercano en $\mathcal{PF}^*$. Entre dos conjuntos de soluciones A y B, se prefiere aquel que tenga menor distancia generacional GD.

7) Distancia Mínima: La métrica representa la distancia mínima existente entre una aproximación A del frente Pareto óptimo $\mathcal{PF}^*$ y se define como [28]:

$$MD = \text{mín}\left(\{d_i^{\text{mín}} | 1 \leq i \leq |A|\}\right) \tag{22}$$

donde $d_i^{\text{mín}}$ es la distancia Euclidiana entre cada vector objetivo en A y su correspondiente más cercano en $\mathcal{PF}^*$. Entre dos conjuntos de soluciones A y B, se prefiere aquel que tenga menor distancia mínima MD.

VII. Análisis de Resultados

El *hardware* utilizado en estos experimentos es un computador con procesador Intel® Core™i7 CPU 970 @ 3,20 GHz con 12 GB RAM y sistema operativo Fedora 20. El lenguaje de programación utilizado para la implementación de los algoritmos y el cálculo de las métricas es Python a excepción del cálculo del hipervolúmen, para lo cual se utilizó el programa creado por Carlos M. Fonseca, Manuel López-Ibáñez, Luís Paquete, y Andreia P. Guerreiro que se encuentra disponible en http://iridia.ulb.ac.be/~manuel/hypervolume al momento de escribir este artículo.

La implementación de los experimentos se hizo tomando en cuenta la descripción de los algoritmos originales [10], [16], [17] . Además se consideró el análisis de complejidad de las implementaciones realizadas para esta investigación. En ambos casos la complejidad de los algoritmos fue consistente con lo sugerido por los autores:

- NSGA-II: Complejidad temporal $M(2N)^2$.
- MOPSO: Complejidad temporal MN^2.

Sea la relación $\mathcal{U}$:

$$\mathcal{U} = \frac{\texttt{complejidad(NSGA)}}{\texttt{complejidad(MOPSO)}} = \frac{M(2N)^2}{MN^2} = 4 \tag{23}$$

De (23), es intuitivo esperar que el tiempo de ejecución de MOPSO sea alrededor de 4 veces menor al del algoritmo NSGA-II; por lo tanto, utilizar MOPSO para resolver problemas *many-objective* podría resultar ventajoso cuando el contexto del problema imponga límites temporales. Para validar esta hipótesis, fueron realizados diversos experimentos que se muestran en la siguiente sección.

VII-A. Experimentos relacionados con la Complejidad

Para validar el análisis de complejidad realizado, fue preparado el siguiente experimento utilizando el problema DTLZ1:

Variando M (cantidad de objetivos), n (dimensión del vector de decisión), N (tamaño de la población) y E (cantidad límite de evaluaciones de la función objetivo) se comparan los tiempos de ejecución.

Siendo t_{MOPSO} y t_{NSGA} los tiempos de ejecución para los algoritmos con los mismos parámetros, se calcula la relación τ:

$$\tau = \frac{t_{\text{NSGA}}}{t_{\text{MOPSO}}} \tag{24}$$

Se ejecutaron 3.000 combinaciones de parámetros:

- $N \in \{100, 200, 300, \cdots, 1000\}$
- $M \in \{4, 6, 8, 10, 12\}$
- $n \in \{14, 15, 16, 17, 18, 19\}$
- $E \in \{1000, 2000, 3000, \cdots, 10000\}$

Una vez terminadas las ejecuciones se calcularon las correlaciones entre cada variable arriba mencionada y τ. Como ejemplo, se muestra la fórmula utilizada para la variable N:

$$\rho_{\tau,N} = \frac{\sum(\tau_i - \overline{\tau})(N_i - \overline{N})}{\sqrt{\sum(\tau_i - \overline{\tau})^2 \sum(N_i - \overline{N})^2}} \tag{25}$$

donde $\overline{N}$ es el promedio de valores de N y $\overline{\tau}$ es el promedio de valores de τ manteniendo constantes las demás variables (M, n, E) para ambas corridas.

Los tiempos de ejecución promedio se muestran en la Tabla I, confirmando la hipótesis inicial que el MOPSO sería más rápido que el NSGA-II para un mismo número de generaciones y evaluaciones de la función objetivo.

Tabla I: Tiempos de ejecución en promedio. Datos experimentales pueden verse en [23].

Algoritmo	Tiempo de Ejecución [s]
NSGA-II	35,087
MOPSO	3,063

La influencia que cada variable tiene sobre los tiempos de ejecución fueron estudiados mediante las correlaciones entre τ y cada variable. Las correlaciones se muestran en la Tabla II que sigue.

De los cálculos, queda claro que incrementando N, M y E se favorece el uso del MOPSO para problemas *many-objective*. Sin embargo, al incrementar n NSGA-II sigue siendo una buena opción. Es importante notar que valores grandes de n aumentan naturalmente el valor de E, lo que a su vez favorece al MOPSO.

Los resultados obtenidos en el análisis de complejidad temporal indican que MOPSO es un buen candidato para resolver problemas *many-objective*. Para verificar este indicio, se procedió a probar también la calidad de las soluciones generadas por el algoritmo en distintos problemas y con distintas configuraciones, aumentado principalmente la cantidad de objetivos M. Para ello, se realizaron experimentos ejecutando los algoritmos NSGA-II y MOPSO configurados para $M \in \{2, 3, 4, \cdots, 20\}$ y los problemas DTLZ1 al DTLZ5

Tabla II: Correlaciones Experimentales

Variable	Correlación	Comentario	Algoritmo Favorecido al Crecer la Variable
$\rho_{\tau,N}$	0,796	Influencia fuerte de N	MOPSO
$\rho_{\tau,M}$	0,257	Influencia débil de M	MOPSO
$\rho_{\tau,n}$	$-0{,}912$	Influencia fuerte de n	NSGA-II
$\rho_{\tau,E}$	0,787	Influencia fuerte de E	MOPSO

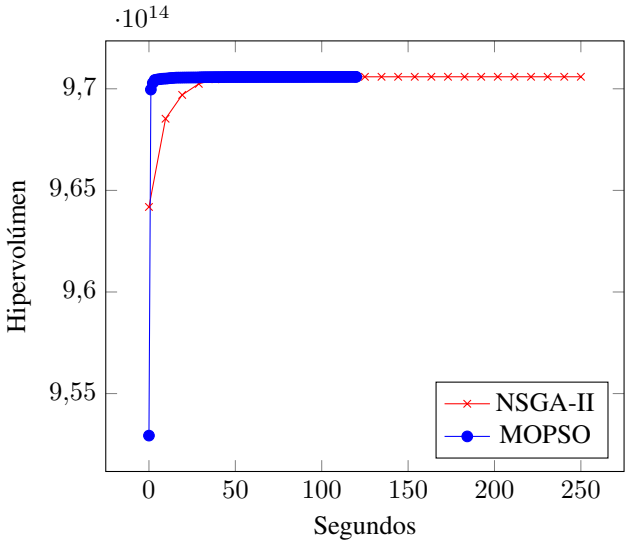

Figura 1: Ejecución de MOPSO y NSGA-II para DTLZ3 con 5 objetivos. Para esta corrida en particular, el algoritmo MOPSO generó 337 soluciones, mientras que NSGA-II llega al límite de 1000. El hipervolúmen final del MOPSO es de $97.058.68 \times 10^{10}$ mientras el de NSGA-II es $97.059.20 \times 10^{10}$ (con lo que NSGA tiene un valor $0{,}00054\,\%$ mejor que el MOPSO). El hecho de que el MOPSO haya generado con mayor rapidez una calidad mayor de soluciones es un claro indicio de que la afirmación de este trabajo es acertada, y que el MOPSO merece atención al momento de resolver problemas *many-objective*.

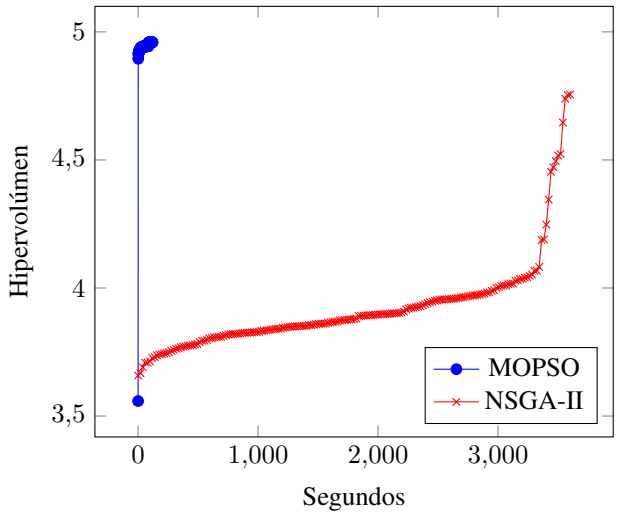

Figura 2: Métrica hipervolúmen para $M = 2$ y DTLZ5. Las pruebas con este problema en particular tuvieron que ejecutarse varias veces, descartando resultados, ya que el NSGA-II no parecía estabilizarse mientras MOPSO sí lo hacía. Los autores destacan la dificultad del DTLZ5 al ser su frente Pareto óptimo una curva con $M - 1$ dimensiones (un único punto en $M = 2$) [24]. Esto indica que para este tipo de problemas es conveniente aprovechar la cualidad del MOPSO de explotar una zona deseable. En casos como el DTLZ5 dicha cualidad resulta muy ventajosa, pues el algoritmo MOPSO concentra las partículas cerca del punto óptimo. El algoritmo NSGA-II en su esfuerzo por mantener un frente Pareto distribuido y mantener la diversidad, encuentra en esta clase de problemas un obstáculo difícil de sortear.

arriba descritos. Las comparaciones que se muestran en la siguiente sección son el resultado de la ejecución de cada una de las configuraciones mencionadas con diferentes parámetros, seleccionando para la comparación los experimentos que obtuvieran mejores resultados en la mayor cantidad de métricas.

VII-B. Experimentos con Métricas de Calidad

El comportamiento típico del hipervolúmen puede verse en la Figura 1. El MOPSO genera rápidamente buenos conjuntos de solución, sin embargo el NSGA-II obtiene mejores soluciones a largo plazo para $M \leq 4$. Este comportamiento se mantiene para los problemas fáciles del conjunto como el DTLZ1 y valores pequeños de M. En cambio, en otros problemas más complejos o con más objetivos, el MOPSO muestra una ventaja contundente como puede apreciarse en la Figura 2.

VII-C. Análisis de Tendencias

En la Figura 3 se muestra la tendencia de la tasa de mejora del hipervolúmen entre 2 y 6 objetivos que se presenta con la variable δ_{HV} definida como:

$$\delta_{HV} = \frac{HV(\text{NSGA-II})}{HV(\text{MOPSO})} \tag{26}$$

La línea horizontal verde indica el valor 1, por debajo de la cual el valor del hipervolúmen del MOPSO supera al del NSGA-II. La línea de tendencia se calcula utilizando el método de mínimos cuadrados, que minimiza el error cuadrático medio de los datos experimentales (puntos) obtenidos en las pruebas realizadas. Claramente, la tendencia al aumentar el número de objetivos favorece al MOPSO.

La Figura 4 muestra la tendencia de la tasa de mejora del Indicador ϵ entre 2 y 20 objetivos, para lo cual fue definida la

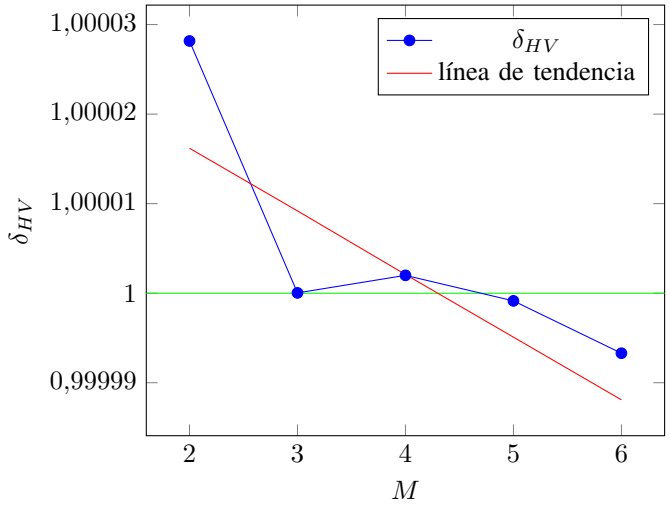

Figura 3: Resultados consolidados para la métrica hipervolúmen para el problema DTLZ1. Puede verse que para este problema para valores pequeños de M los valores están sobre la línea del 1, indicando que los valores obtenidos por NSGA-II son mejores que los obtenidos por MOPSO. Es importante notar en cambio que los valores son en todos los casos muy similares, y que la tendencia indica que a medida que se incrementa la cantidad de objetivos (y el problema se vuelve *many-objective*) el conjunto generado por MOPSO se vuelve ventajoso. Estos resultados validan la propuesta inicial de este trabajo en el sentido que el MOPSO es una buena opción para MaOPs.

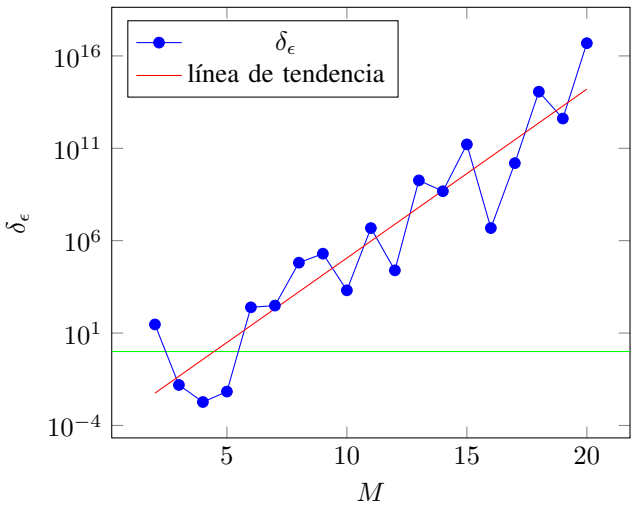

Figura 4: Resultados consolidados para el Indicador ϵ para el problema DTLZ1. Debido a la gran diversidad de ordenes de magnitud de los puntos se utiliza un eje logarítmico. Puede observarse que a medida que aumenta la cantidad de objetivos, el factor por el cual las soluciones generadas por NSGA-II dominarían a todas las soluciones generadas por MOPSO aumenta drásticamente, lo cual es un indicio de la validez de nuestra afirmación que el MOPSO es un algoritmo interesante para problemas *many-objective*.

variable δ_ϵ como:

$$\delta_\epsilon = \frac{I_\epsilon(\text{NSGA-II, MOPSO})}{I_\epsilon(\text{MOPSO, NSGA-II})} \tag{27}$$

Nuevamente se observa la ventaja de utilizar el MOPSO frente al NSGA-II

La Figura 5 muestra la tendencia de la tasa de mejora del hipervolúmen entre 2 y 6 objetivos para el problema DTLZ4. Puede verse en esta figura como el NSGA-II aún mantiene ventaja en ciertos escenarios en los que la exploración es un elemento clave, condición en la que el NSGA-II puede superar al un MOPSO incluso al aumentar el número de objetivos.

En la tabla III pueden verse los resultados para todo el conjunto de métricas para $M = 6$ utilizando el DTLZ1. Para facilitar la lectura se agregan los valores de las soluciones dominantes y soluciones dominadas de cada algoritmo y la columna "Mejor Algoritmo" para las métricas.

En la tabla IV se pueden ver resultados para el número máximo de objetivos tratado en este trabajo. Es una muestra clara de que para ciertos escenarios no todas las métricas son útiles. En este caso en particular, la unión de ambos conjuntos es no dominado y por lo tanto las métricas de cobertura no ofrecen información acerca de cual es el mejor algoritmo. Además en el caso del DTLZ4 los valores del Indicador ϵ se dispararon a magnitudes tales que superaban el límite superior estándar de números de punto flotante de Python ($1.797.693.134.862.315.7 \times 10^{308}$, valor obtenido utilizando `sys.float_info` en el computador en donde

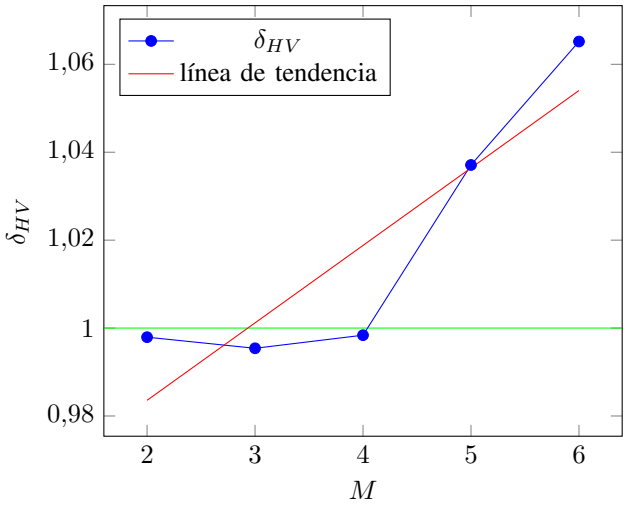

Figura 5: Resultados consolidados para la métrica hipervolúmen para el problema DTLZ4. La dificultad de este problema es la tendencia de los algoritmos de generar soluciones en los planos de alta densidad. Esta dificultad es un punto débil del MOPSO, al ser el este un algoritmo con mejor explotación que exploración. Puede verse en el gráfico como progresivamente el NSGA-II obtiene ventaja en un problema donde la exploración es clave para enfrentar la tendencia a concentrar soluciones en una zona de alta densidad.

Tabla III: Métricas para $M = 6$ y DTLZ1.

Método	**A =NSGA-II**	**B =MOPSO**	**Mejor Algoritmo**
Cant. Soluciones	1000	847	NSGA-II
Cant. Soluciones Dominantes	739	10	NSGA-II
Cant. Soluciones Dominandas	68	649	NSGA-II
Cobertura	$C(B,A) = \frac{68}{1000} = 0{,}068$	$C(A,B) = \frac{649}{847} = 0{,}766$	NSGA-II
Cobertura Complementaria	$\overline{C}(B,A) = \frac{739}{1000} = 0{,}739$	$\overline{C}(A,B) = \frac{10}{847} = 0{,}0118$	NSGA-II
Hypervolúmen Final	$HV(A) = 59.045.78 \times 10^{11}$	$HV(B) = 59.046.18 \times 10^{11}$	MOPSO
Distancia Generacional	$GD(A) = 6{,}745$	$GD(B) = 43{,}255$	NSGA-II
Distancia Mínima	$MD(A) = 2{,}164$	$MD(B) = 0{,}433$	MOPSO
Indicador ϵ	$I_\epsilon(A,B) = 2{,}845 \times 10^{+11}$	$I_\epsilon(B,A) = 1{,}151 \times 10^{+09}$	MOPSO

Tabla IV: Métricas para $M = 20$ y DTLZ4.

Método	**A =NSGA-II**	**B =MOPSO**	**Mejor Algoritmo**
Cant. Soluciones	1000	1000	
Cant. Soluciones Dominantes	0	0	
Cant. Soluciones Dominadas	0	0	
Cobertura	$C(B,A) = \frac{0}{1000} = 0{,}000$	$C(A,B) = \frac{0}{1000} = 0{,}000$	
Cobertura Complementaria	$\overline{C}(B,A) = \frac{0}{1000} = 0{,}0$	$\overline{C}(A,B) = \frac{0}{1000} = 0{,}0$	
Distancia Generacional	$GD(A) = 0{,}372$	$GD(B) = 0{,}622$	NSGA-II
Distancia Mínima	$MD(A) = 0{,}339$	$MD(B) = 2{,}576 \times 10^{-05}$	MOPSO
Indicador ϵ	No comparable en la precisión deseasa.		

fueron ejecutadas las pruebas).

VIII. Conclusiones

El punto de partida de este trabajo fue la premisa de que el algoritmo MOPSO puede representar una alternativa atractiva para resolver problemas de muchos objetivos (*many-objective problems*), lo cual quedó claramente establecido con los resultados experimentales presentados en la sección anterior. En efecto, la tendencia en la métrica más utilizada (el hipervolúmen) [1] fue claramente favorable al MOPSO en la mayoría de los problemas. Incluso, el MOPSO converge rápidamente a muy buenas soluciones debido a su demostrada capacidad de explotación y menor complejidad computacional.

Se destaca que tanto el NSGA-II como el MOPSO se ven afectados por las dificultades de los problemas más difíciles del *benchmark* DTLZ considerado en este trabajo. Como ejemplo, ninguno de los algoritmos pudo converger a la curva óptima en el problema DTLZ5. Notablemente, el MOPSO logró una ventaja importante con respecto al resultado del NSGA-II, por lo que se puede afirmar que el MOPSO resulta atractivo en problemas con dificultad de convergencia a una zona limitada del espacio objetivos como el DTLZ5. A su vez, basados en los resultados obtenidos con el DTLZ4, donde el NSGA-II tuvo una evidente ventaja, se puede afirmar de igual manera que el NSGA-II resulta más conveniente en escenarios en los que una alta densidad de soluciones no dominadas entre sí atraen a las partículas del MOPSO.

Aunque el hipervolúmen sea la métrica más utilizada en tradicionales problemas multiobjetivo (MOPs) con un bajo número de funciones objetivo, queda claro de los experimentos que no es escalable para problemas *many-objective*, por lo que en este trabajo solo se lo utilizó con problemas de hasta 6 objetivos. Consecuentemente, queda por establecerse cuál será la métrica a ser utilizada como referencia en problemas de muchos objetivos, no existiendo aún ninguna métrica única que haya sido adoptada por la comunidad científica, lo que llevó a los autores de este trabajo a proponer una nueva métrica (ecuación 20) que dimos en llamar *cobertura complementaria*.

Es notable también que mientras el MOPSO tiene la ventaja de la rapidez, el NSGA-II consistentemente genera frentes más uniformes a lo largo del frente Pareto. Esto se evidencia al considerar las métricas de cobertura, donde el NSGA lleva una ventaja evidente. El uso por parte del NSGA-II de un valor de *crowding distance*, ciertamente mantiene un buen nivel de distribución que el MOPSO no pudo lograr en los experimentos realizados.

Como trabajo futuro se propone estudiar también el algoritmo NSGA-III [33] y compararlo con el MOPSO utilizando los experimentos arriba descritos. Además, se sugieren realizar modificaciones al algoritmo MOPSO para aumentar la calidad de las soluciones que encuentra. Por ejemplo, introducir el concepto de densidad poblacional (*crowding*) al MOPSO al momento de:

- la selección de los mejores representantes de los repositorios;
- la eliminación de elementos de un repositorio lleno.

Referencias Bibliográficas

[1] C. von Lücken, B. Barán, y C. Brizuela, "A survey on multi-objective evolutionary algorithms for many-objective problems" *Computational Optimization and Applications*, páginas. 1–50, 2014.

[2] C. A. Coello Coello, G. B. Lamont, y D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer, 2nd ed., Sept. 2007.

[3] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[4] D. W. Corne, J. D. Knowles, y M. J. Oates, "The pareto envelope-based selection algorithm for multiobjective optimization" en *Proceedings of the Parallel Problem Solving from Nature VI Conference*, páginas. 839–848, Springer, 2000.

[5] M. Farina y P. Amato, "On the optimal solution definition for many-criteria optimization problems" en *Fuzzy Information Processing Society, 2002. Proceedings. NAFIPS. 2002 Annual Meeting of the North American*, páginas. 233–238, 2002.

[6] E. Hughes, "Evolutionary many-objective optimisation: many once or one many?" en *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 1, páginas. 222–227 Vol.1, Sept 2005.

[7] V. Khare, X. Yao, y K. Deb, "Performance scaling of multi-objective evolutionary algorithms" en *Evolutionary Multi-Criterion Optimization* (C. Fonseca, P. Fleming, E. Zitzler, L. Thiele, y K. Deb, eds.), vol. 2632 of *Lecture Notes in Computer Science*, páginas. 376–390, Springer Berlin Heidelberg, 2003.

[8] R. Purshouse y P. Fleming, "Evolutionary many-objective optimisation: an exploratory analysis" en *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, vol. 3, páginas. 2066–2073 Vol.3, Dec 2003.

[9] R. Purshouse y P. Fleming, "On the evolutionary optimization of many conflicting objectives" *Evolutionary Computation, IEEE Transactions on*, vol. 11, páginas. 770–784, Dec 2007.

[10] J. Moore y R. Chapman, "Application of particle swarm to multiobjective optimization" *Department of Computer Science and Software Engineering, Auburn University*, 1999.

[11] P. Fleming, R. Purshouse, y R. Lygoe, "Many-objective optimization: An engineering design perspective" en *Evolutionary Multi-Criterion Optimization* (C. Coello Coello, A. Hernández Aguirre, y E. Zitzler, eds.), vol. 3410 of *Lecture Notes in Computer Science*, páginas. 14–32, Springer Berlin Heidelberg, 2005.

[12] S. Mostaghim, J. Branke, y H. Schmeck, "Multi-objective particle swarm optimization on computer grids" en *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, páginas. 869–875, ACM, 2007.

[13] S. Obayashi y D. Sasaki, "Visualization and data mining of pareto solutions using self-organizing map" en *Evolutionary multi-criterion optimization*, páginas. 796–809, Springer, 2003.

[14] A. Pryke, S. Mostaghim, y A. Nazemi, "Heatmap visualization of population based multi objective algorithms" en *Evolutionary multi-criterion optimization*, páginas. 361–375, Springer, 2007.

[15] D. Walker, R. Everson, y J. Fieldsend, "Visualisation and ordering of many-objective populations" en *Evolutionary Computation (CEC), 2010 IEEE Congress on*, páginas. 1–8, July 2010.

[16] K. Deb, A. Pratap, S. Agarwal, y T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii" *Evolutionary Computation, IEEE Transactions on*, vol. 6, páginas. 182–197, Apr 2002.

[17] C. A. Coello Coello y M. Lechuga, "Mopso: a proposal for multiple objective particle swarm optimization" en *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, vol. 2, páginas. 1051–1056, 2002.

[18] M. Farina y P. Amato, "On the optimal solution definition for many-criteria optimization problems" en *Fuzzy Information Processing Society, 2002. Proceedings. NAFIPS. 2002 Annual Meeting of the North American*, páginas. 233–238, 2002.

[19] J. Kennedy y R. Eberhart, "Particle swarm optimization" en *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, páginas. 1942–1948 vol.4, Nov 1995.

[20] J. Lima y B. Barán, "Optimización de enjambre de partículas aplicada al problema del cajero viajante bi–objetivo" *Revista Iberoamericana de Inteligencia Artificial*, 2006.

[21] Y. Shi y R. Eberhart, "A modified particle swarm optimizer" en *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, páginas. 69–73, May 1998.

[22] Y. Shi y R. Eberhart, "Parameter selection in particle swarm optimization" en *Evolutionary Programming VII* (V. Porto, N. Saravanan, D. Waagen, y A. Eiben, eds.), vol. 1447 of *Lecture Notes in Computer Science*, páginas. 591–600, Springer Berlin Heidelberg, 1998.

[23] M. Torres, B. Barán, y U. Yael, "Particle swarm optimisation algorithms for solving many-objective problems" en *3rd Conference of Computational Interdisciplinary Sciences*, páginas. 144–155, 2014.

[24] K. Deb, L. Thiele, M. Laumanns, y E. Zitzler, "Scalable Multi-Objective Optimization Test Problems" en *Congress on Evolutionary Computation (CEC 2002)*, páginas. 825–830, IEEE Press, 2002.

[25] E. Zitzler, K. Deb, y L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results" *Evolutionary Computation*, vol. 8, no. 2, páginas. 173–195, 2000.

[26] E. Zitzler y L. Thiele, "Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study" en *Conference on Parallel Problem Solving from Nature (PPSN V)*, (Amsterdam), páginas. 292–301, 1998.

[27] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, y V. Grunert da Fonseca, "Performance Assessment of Multiobjective Optimizers: An Analysis and Review" *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, páginas. 117–132, 2003.

[28] D. A. Van Veldhuizen, "Multiobjective evolutionary algorithms: classifications, analyses, and new innovations" tech. rep., DTIC Document, 1999.

[29] E. Zitzler, *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, ETH Zurich, Switzerland, 1999.

[30] N. Beume, C. Fonseca, M. López-Ibañez, L. Paquete, y J. Vahrenhold, "On the complexity of computing the hypervolume indicator" *Evolutionary Computation, IEEE Transactions on*, vol. 13, páginas. 1075–1082, Oct 2009.

[31] T. Chan, "Klee's measure problem made easy" en *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, páginas. 410–419, Oct 2013.

[32] C. Fonseca, L. Paquete, y M. Lopez-Ibanez, "An improved dimension-sweep algorithm for the hypervolume indicator" en *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, páginas. 1157–1163, July 2006.

[33] K. Deb y H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints" *Evolutionary Computation, IEEE Transactions on*, vol. 18, páginas. 577–601, Aug 2014.