IMPLEMENTACIÓN DE UN COMPARADOR INTERACTIVO DE LOS ALGORITMOS DE OPTIMIZACIÓN MULTI-CRITERIO NSGA-II Y MOPSO EN UNA INTERFAZ GRÁFICA PARA USUARIOS NO EXPERTOS

DAVID FRANCISCO DORADO SEVILLA

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
INGENIERÍA ELECTRÓNICA
PASTO
2018

IMPLEMENTACIÓN DE UN COMPARADOR INTERACTIVO DE LOS ALGORITMOS DE OPTIMIZACIÓN MULTI-CRITERIO NSGA-II Y MOPSO EN UNA INTERFAZ GRÁFICA PARA USUARIOS NO EXPERTOS

DAVID FRANCISCO DORADO SEVILLA

Trabajo de grado para optar el título de Ingeniero Electrónico

ASESOR
DIEGO HERNÁN PELUFFO ORDÓÑEZ

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
INGENIERÍA ELECTRÓNICA
PASTO
2018

Nota de aceptación

_____

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

Pasto, 5 de Junio de 2018

DEDICATORIA

Primero, agradezco y doy gloria a Dios por haberme regalado una vida maravillosa llena de felicidad y constante aprendizaje. A mis padres, por ser mi ejemplo a seguir y el apoyo incondicional en este hermoso y difícil camino y enseñarme siempre a superar barreras y alcanzar grandes logros. A mi señora, por su amor y apoyo incondicional bajo cualquier circunstancia. A mi hermano por su amistad y preocupación. A mi profesor Diego Hernán Peluffo, por haberme apoyado en este trabajo y por haberme dado el impulso necesario para concluir con mi carrera.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ALGORITMOS

TABLA DE ACRÓNIMOS

NSGA-II:	Non-dominated Genetic Algorithm – II.
MOPSO:	Multiobjective Particle Swarm Optimization.
SPEA:	Strenght Pareto Evolutionary Algorithm.
PAES:	Pareto-Archived Evolution Strategy
VEPSO:	Vector Evaluated Particle Swarm Optimization

LISTA DE ANEXOS

RESUMEN

Un problema de optimización multiobjetivo, según su enfoque, puede significar minimizar o maximizar un grupo de al menos dos funciones objetivo, con el fin de encontrar el mejor conjunto de soluciones posibles a dichas funciones. Existen varios métodos de optimización multiobjetivo, pero la calidad de las soluciones encontradas varía según el método utilizado y la complejidad del problema planteado. Un recorrido bibliográfico ha permitido notar que los métodos derivados de la Computación Evolutiva entregan buenos resultados y son de uso común en trabajos de investigación. Aunque se han podido encontrar estudios comparativos entre dichos métodos de optimización, las conclusiones que estos ofrecen al lector no permiten definir una regla general que determine que un método es mejor que otro. Por lo anterior, la elección de un método de optimización que se adapte a un problema planteado puede significar una dificultad para no expertos en el tema.

En esta tesis se propone implementar una interfaz gráfica que permita a usuarios no expertos en optimización multi-criterio, comparar de forma interactiva el desempeño de los algoritmos NSGA-II y MOPSO, que han sido escogidos de forma cualitativa de un grupo de cinco algoritmos preseleccionados como los mejores representantes de algoritmos evolutivos e inteligencia de enjambres. Luego, en este trabajo se propone una metodología de comparación que permite al usuario, analizar resultados gráficos y numéricos para observar el comportamiento de los algoritmos y determinar cuál de los dos se acopla mejor a sus necesidades.

Palabras clave: Optimización multiobjetivo, computación evolutiva, algoritmos evolutivos, inteligencia de enjambres.

ABSTRACT

A problem of multiobjective optimization, according to its approach, can mean either minimizing or maximizing a group of at least two objective functions, in order to find the best possible set of solutions to such functions. There are several methods of multiobjective optimization, in which the resulting solutions' quality varies depending on the method used and the complexity of the posed problem. A bibliographical review allowed us to notice, that the methods derived from the Evolutionary Computation deliver good results and are commonly used in research works. Although comparative studies among these optimization methods have been found, the conclusions that these ones offer to the reader do not allow to define a general rule that determines when one method is better than another. Therefore, the choice of well adapted optimization method can be a difficult task for non-experts in the field.

In this thesis, it is proposed to implement a graphical interface that allows non-expert users in multi-objective optimization, to interactively compare the performance of the NSGA-II and MOPSO algorithms, which have been chosen qualitatively from a group of five preselected algorithms as members of evolutionary algorithms and swarm intelligence. Therefore, a comparison methodology is proposed to allow the user to analyze graphical and numerical results to observe the behavior of the algorithms and determine which of the both is best suited to their needs.

Keywords: Multiobjective optimization, evolutionary computation, evolutionary algorithms, swarm intelligence.

1 INTRODUCCIÓN

Los problemas a los que las personas se pueden ver enfrentados deben abordarse, en la mayor parte de los casos, teniendo en cuenta más de un objetivo simultáneamente. En este tipo de problemas no existe una solución única que satisfaga todos los objetivos planteados, sino más bien un conjunto de soluciones posibles. Dicho conjunto es muy extenso y si lo que se desea es obtener los mejores resultados, se deben optimizar las funciones objetivo para encontrar el subconjunto que contenga las mejores soluciones como se muestra en la Figura 1. Actualmente, los procesos de optimización son muy comunes debido a que son requeridos en distintas áreas de trabajo, por ejemplo, el alto desarrollo industrial actual se basa en una economía lineal en la cual se explotan recursos para producir y luego consumir y desechar, sin tener en cuenta que la mayoría de los recursos utilizados son limitados, luego, la optimización de los procesos industriales puede ayudar a la disminución de desperdicios y así reducir impactos negativos en el medio ambiente.

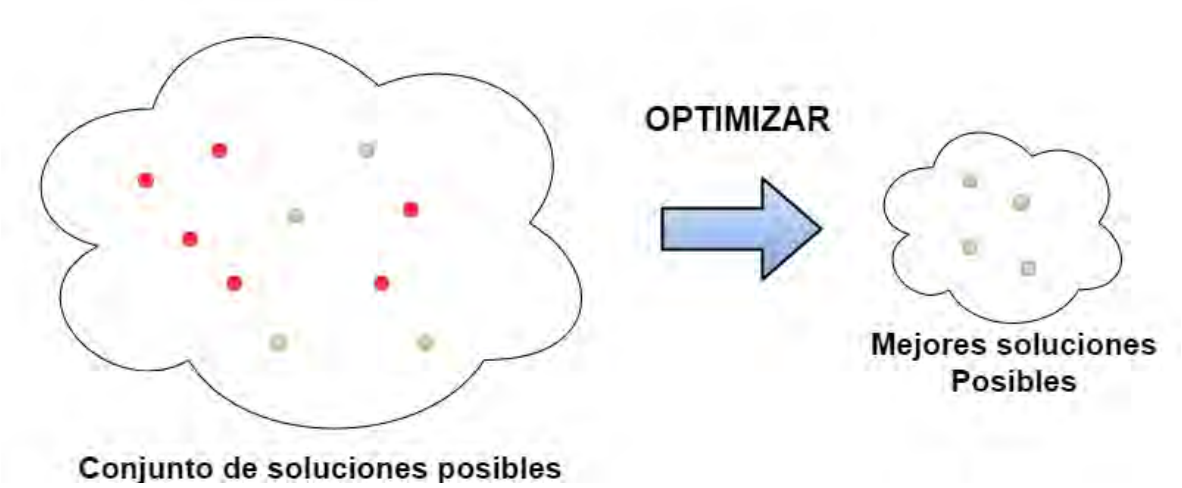


Figura 1. Optimización de un problema multi-criterio

Dentro del amplio catálogo de métodos de optimización multi-criterio se destacan, los Algoritmos Genéticos y los algoritmos basados en Inteligencia Colectiva, al ser comúnmente utilizados en la literatura gracias a sus buenos resultados. La calidad del conjunto de soluciones encontrado puede variar según el método aplicado, teniendo en cuenta que no existe una regla general que permita definir a un método A, como mejor que un método B. En el mejor de los casos podría decirse que un método A no es peor que un método B, respecto a un problema multi-criterio planteado. Por lo tanto, la selección de un método que se acople a las necesidades del problema de optimización planteado puede definirse como un problema abierto.

En este trabajo de grado, se describe el desarrollo de una interfaz comparativa interactiva de los métodos de optimización NSGA-II [1] Y MOPSO [2], que han sido seleccionados luego de un estado del arte, como representantes de dos de las ramas de optimización más utilizadas: Los Algoritmos inspirados en teorías Evolutivas y los inspirados en Inteligencia de Enjambres. La función de la interfaz desarrollada en Matlab es permitir a su usuario aplicar los algoritmos mencionados a cinco diferentes problemas de prueba, y entregar la información necesaria para que este pueda concluir que método se adapta mejor a sus necesidades.

Es importante resaltar que la investigación realizada en este trabajo se inspiró un objetivo específico de la tesis doctoral en educación *"Análisis de factores determinantes en las técnicas de enseñanza aprendizaje de las matemáticas en estudiantes de educación media desde un enfoque multicriterio"*, que plantea escoger un método de optimización adecuado. También, se puede relacionar este trabajo con el área de investigación de análisis de datos basado en múltiples expertos [3] [4].

1.1 JUSTIFICACIÓN

Actualmente, es común el uso de métodos de optimización debido a la necesidad de resolver problemas multi-criterio. Teniendo en cuenta que los resultados de los algoritmos de optimización varían según el método utilizado, y que por tanto se busca encontrar un conjunto óptimo de soluciones, es evidente la necesidad de establecer una herramienta computacional que permita un análisis rápido e interactivo, de tal forma que ofrezca información útil para evaluar el desempeño de los algoritmos frente a diferentes características propias del problema de optimización planteado.

Los algoritmos NSGA-II y MOPSO se escogieron como base del desarrollo de la interfaz, debido a que sus métodos de búsqueda son de fácil entendimiento para un usuario no experto y porque una comparación cualitativa con otros tres algoritmos que se mencionaran más adelante permitió definirlos como los mejores representantes de Algoritmos Evolutivos e Inteligencia de Enjambres.

1.2 OBJETIVOS

1.2.1 Objetivo General

Realizar un estudio comparativo de métodos recientes de optimización multi-criterio con el fin de determinar su desempeño, coste computacional y versatilidad de forma interactiva.

1.2.2 Objetivos Específicos

- Definir los métodos de optimización multi-criterio que serán implementados en programación de medio nivel.
- Definir las medidas de desempeño que serán utilizadas para evaluar los métodos de optimización seleccionados, en términos de desempeño y versatilidad.
- Proponer una metodología de comparación entre los métodos de optimización seleccionados.
- Desarrollar una interfaz comparativa interactiva que permita determinar el comportamiento de los algoritmos seleccionados, basado en la metodología de comparación definida.

1.3 ORGANIZACIÓN DEL DOCUMENTO

Este documento está compuesto por 7 secciones nombradas a continuación: Introducción, marco teórico, metodología, resultados, conclusiones, bibliografía y anexos. El contenido es el que sigue:

- En la Sección 2, se presenta un recorrido conceptual que permite entender qué es la optimización multiobjetivo y entender algunos de los métodos de optimización más comunes derivados de la computación evolutiva.
- En la Sección 3, se describe la metodología de comparación de métodos de optimización basada en medidas de desempeño y problemas de prueba.
- En la Sección 4, se muestran los resultados experimentales y en la Sección 5 se podrán ver las conclusiones de la investigación.
- En las dos últimas secciones se encuentra la bibliografía consultada en la investigación y los anexos de aportes de este trabajo.

2 MARCO TEÓRICO

La tarea de resolver problemas planteados con un enfoque multi-criterio es difícil de alcanzar para quien desea hacerlo, si no se cuentan con las herramientas adecuadas, sobre todo, si lo que se busca es lograr una solución óptima. En esta sección se menciona la teoría necesaria para entender qué es un problema de optimización multi-criterio y como podría llegar a ser resuelto por algunos de los caminos existentes en la literatura.

2.1 OPTIMIZACIÓN MULTI-CRITERIO

Muchos problemas cotidianos exigen alcanzar un fin evaluando simultáneamente diferentes criterios, por lo general contrapuestos. Un ejemplo simple, es que al comprar vivienda se suelen evaluar factores como el sector en donde se ubica el inmueble, los cimientos de la construcción, la empresa constructora, el tamaño del área construida y el costo que se puede afrontar, en resumen, una vivienda de calidad al menor precio posible. En el campo industrial, teniendo en cuenta que hoy en día los recursos naturales no renovables están siendo explotados en exceso, una empresa automotriz, por ejemplo, puede buscar el desarrollo de un vehículo que sea veloz y a la vez amigable con el ambiente. La optimización multi-criterio, pretende ayudarnos a alcanzar determinada meta buscando al conjunto de soluciones que mejor se adapten a los criterios del problema planteado. Dependiendo de las características del problema, optimizar puede significar maximizar o minimizar los objetivos. Por ejemplo, en el problema del automóvil, si se piensa en que el consumo de combustible sea el menor posible, se habla de minimizar. Luego, un problema de optimización multi-criterio en términos de minimización puede definirse formalmente como [5]:

$$
\begin{aligned}
&\textit{Optimizar} && \boldsymbol{y = F(x) = (f_1(x), f_2(x), \dots, f_k(x))} \\
&\textit{Sujeto a restricciones} && \boldsymbol{g(x) = (g_1(x), \dots, g_m(x)) \leq 0} \\
&\textit{Donde} && \boldsymbol{x = (x_1, \dots, x_n) \in X \subseteq R^n} \\
& && \boldsymbol{y = (y_1, \dots, y_k) \in Y \subseteq R^k}
\end{aligned}
\tag{1}
$$

La función $\boldsymbol{F(x)}$ depende de $\boldsymbol{k}$ funciones objetivo y puede representar números reales, binarios, listas, tareas a realizar, etc. El vector de decisión $\boldsymbol{x}$ contiene $\boldsymbol{n}$ variables de decisión que identifican cada solución en el espacio del problema $\boldsymbol{X}$, que es el conjunto de todos los elementos posibles del problema. Las $\boldsymbol{m}$ restricciones de $\boldsymbol{g(x)}$ limitan zonas de búsqueda factibles en las que se encuentra el vector $\boldsymbol{x}$ como se observa en la Figura 2. El vector objetivo $\boldsymbol{y}$ con $\boldsymbol{k}$ objetivos pertenece a el espacio objetivo $\boldsymbol{Y}$ que es el codominio de las funciones objetivo. El mapeo del proceso de optimización se ve en la Figura 3. A los valores encontrados

al resolver las funciones objetivo con las variables de decisión se les conocerá en este trabajo como *valor fitness.*

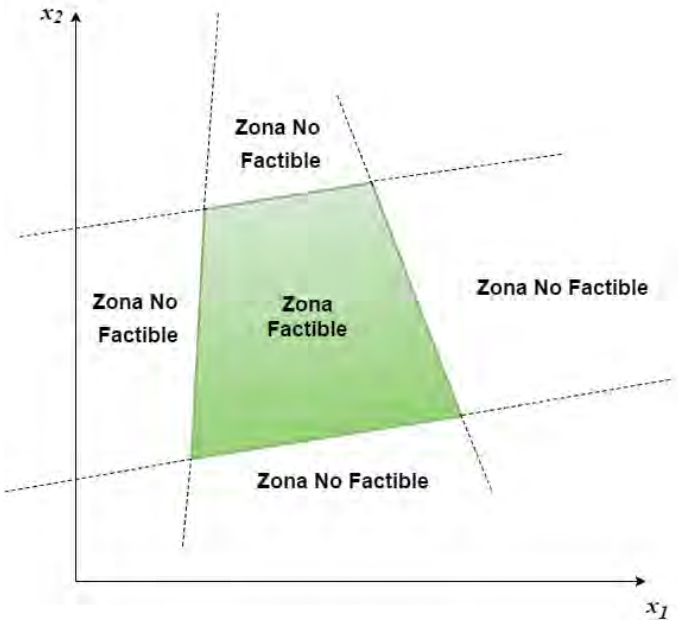


Figura 2. Zona Factible o Espacio de Búsqueda

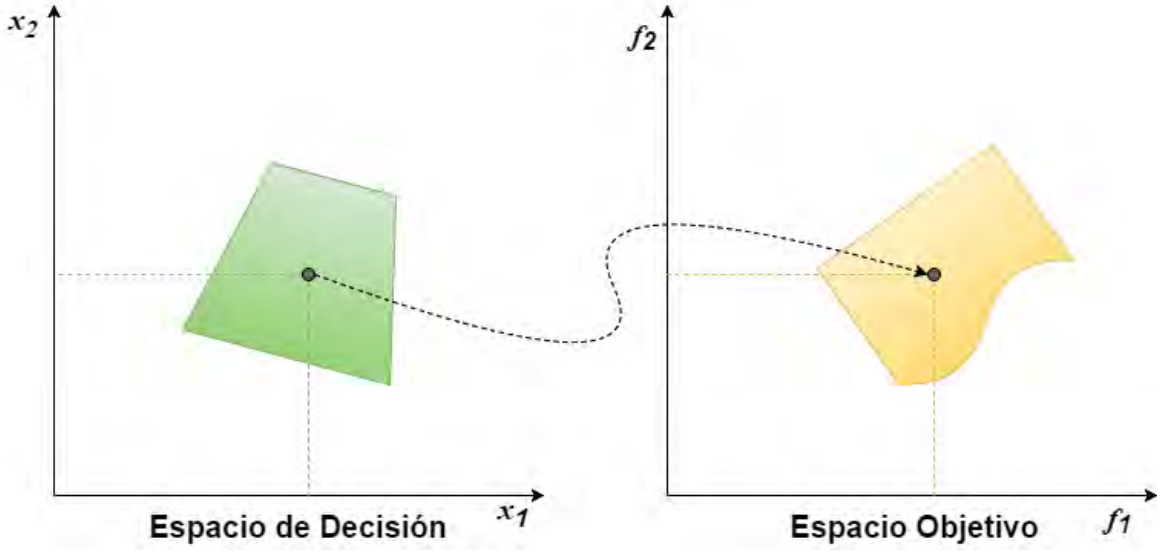


Figura 3. Mapeo de la Optimización

Para poder clasificar del conjunto de soluciones posibles cuáles son las mejores soluciones, se utiliza el termino de dominancia (Vilfredo Pareto,1896), que dice que una solución es Pareto óptima si se encuentra un equilibrio en el que no se pueda mejorar dicha solución si deteriorar otra. Formalmente, siendo $\boldsymbol{u}$ y $\boldsymbol{v}$ vectores

contenidos en el espacio de decisión y $\boldsymbol{f}(\boldsymbol{u})$ y $\boldsymbol{f}(\boldsymbol{v})$ *valores* fitness correspondientes, se puede decir en términos de minimización que:

- El vector dominante será aquel que tenga el menor *valor fitness.* Luego:
  $\boldsymbol{u} \prec \boldsymbol{v}$ ($\boldsymbol{u}$ domina a $\boldsymbol{v}$) si y solo si $\boldsymbol{f}(\boldsymbol{u}) < \boldsymbol{f}(\boldsymbol{v})$.
  $\boldsymbol{v} \prec \boldsymbol{u}$ ($\boldsymbol{v}$ domina a $\boldsymbol{u}$) si y solo si $\boldsymbol{f}(\boldsymbol{v}) < \boldsymbol{f}(\boldsymbol{u})$.
- Las soluciones no son comparables si ninguno de los vectores se domina entre sí. Esto es:
  $\boldsymbol{u} \sim \boldsymbol{v}$ ($\boldsymbol{u}$ y $\boldsymbol{v}$ no son comparables) si y solo si $\boldsymbol{f}(\boldsymbol{u}) \nleq \boldsymbol{f}(\boldsymbol{v}) \ \wedge \ \boldsymbol{f}(\boldsymbol{v}) \nleq \boldsymbol{f}(\boldsymbol{u})$.

A las soluciones se les asigna un rango según su nivel de no dominancia conocido como Frente. El primer Frente, será conformado por las soluciones que no son dominadas por ningún individuo de la población, continuando con el segundo Frente que son las soluciones no dominadas sin tener en cuenta el primer Frente, y se continúa asignando los siguientes rangos sin tener en cuenta las soluciones del Frente anterior. Los métodos de optimización buscan encontrar en el espacio de decisión, al conjunto denominado Pareto óptimo definido como $\boldsymbol{X_{true}} = \{\boldsymbol{x} \in \boldsymbol{X} \mid \boldsymbol{x}$ es no dominado respecto a $\boldsymbol{X}\}$, para luego, alcanzar en el espacio objetivo el Frente de Pareto definido como $Y_{true} = F(X_{true})$ [6]. Para entender mejor los conceptos de dominancia y Frente en la Figura 4 se puede observar un conjunto de soluciones con frentes asignados.

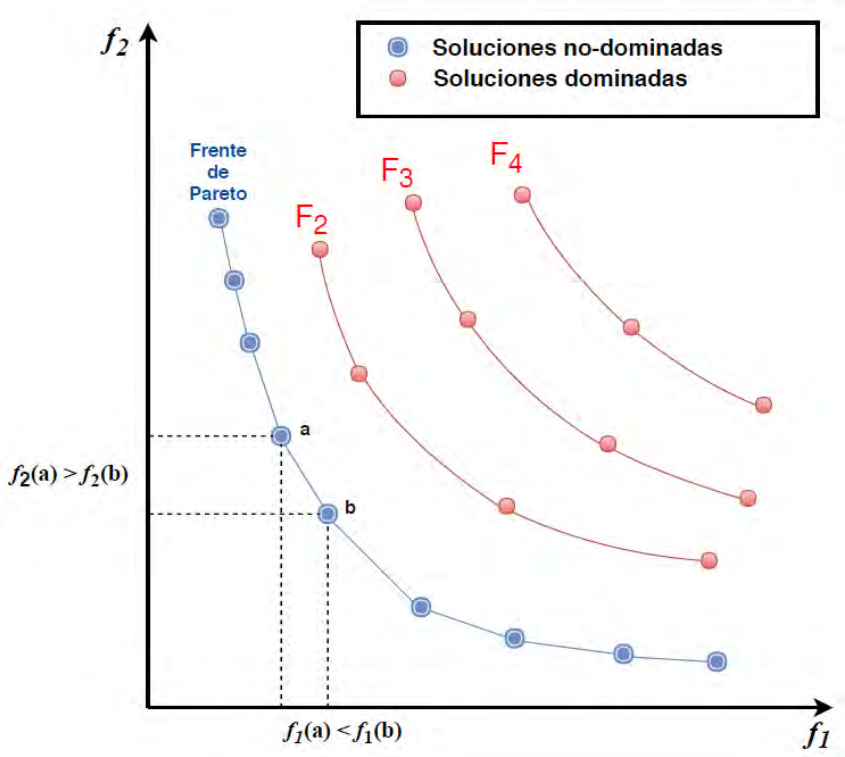


Figura 4. Minimización de f_1 y f_2

2.2 METAHEURÍSTICAS

Para los fines de este trabajo, es necesario enfocarse en Algoritmos de optimización multi-criterio aplicables en herramientas computacionales. Por lo anterior, aquí se abordan las metaheurísticas, que son algoritmos que modifican variables a lo largo del tiempo, guiados por un conocimiento experto para recorrer la zona factible del espacio de decisión de forma iterativa. Los mejores resultados se obtienen aplicando mejoras a una población de soluciones iniciales, basándose en el concepto de dominancia mencionado en la sección anterior con el fin de descartar las soluciones menos aptas [7].

2.2.1 Computación Evolutiva

Son algoritmos que realizan procedimientos de búsqueda inspirados en comportamientos observables en especies de la naturaleza, modificando individuos de la población de forma iterativa tras una serie de procesos determinados. Estos algoritmos se pueden clasificar como se ve en la Figura 5.

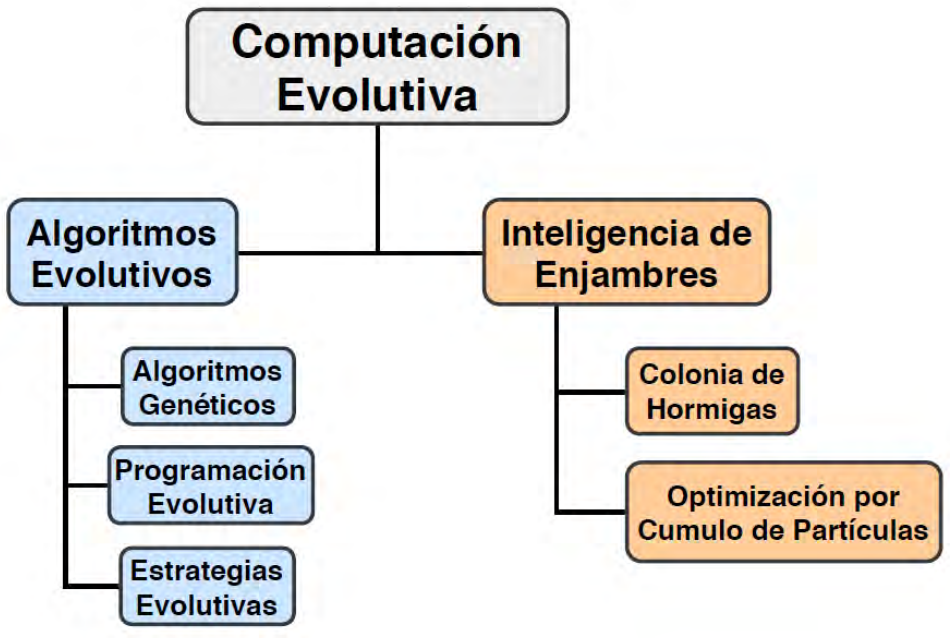


Figura 5. Ramas de la Computación Evolutiva

2.2.1.1 Algoritmos Evolutivos

Los algoritmos evolutivos enfocados a resolver problemas multi-criterio, se empezaron a desarrollar a mediados de los años 80 con la aparición de las computadoras personales de buen rendimiento y bajo costo, como la Macintosh 128k, y ha habido varias publicaciones de este tipo de algoritmos desde entonces.

Son algoritmos inspirados en la teoría darwiniana de la evolución natural, que actúan sobre una población de soluciones teniendo en cuenta los factores de diversidad y la supervivencia del más fuerte. Pueden dividirse en 3 clases [8]:

- **Algoritmos Genéticos:** Estos algoritmos toman la teoría de cruce de especies como operador de búsqueda principal y usan un operador de mutación para mantener la diversidad en las soluciones.
- **Programación Evolutiva:** Este algoritmo basa su proceso de búsqueda en la mutación sin tener en cuenta la recombinación.
- **Estrategias Evolutivas:** Estos algoritmos hacen evolucionar una población de números reales que codifican las posibles soluciones y los tamaños de salto.

En un recorrido bibliográfico, se puede entender que estos algoritmos tienen buenas formas para recorrer el espacio de búsqueda de un problema, pero requieren de varias generaciones o iteraciones para lograrlo, por lo que toma un tiempo largo para alcanzar su meta [9].

2.2.1.2 Inteligencia de Enjambres

Son algoritmos basados en el comportamiento de especies de la naturaleza, que actúan con una inteligencia colectiva conforme a los fines que tenga el grupo. Las principales clases se mencionan a continuación [10]:

- **Optimización por Cúmulo de Partículas:** Estos algoritmos, basan su proceso de búsqueda en el comportamiento social de cardúmenes y bandadas, que se mueven de forma colectiva en búsqueda de alimentos o para huir de un depredador.
- **Colonia de Hormigas:** Estos algoritmos basan su proceso de búsqueda en el comportamiento de las hormigas, que se mueven por el espacio siguiendo rastros de feromonas dejados por el colectivo para encontrar alimento o su nido.

Este tipo de algoritmos pueden presentar buenos resultados en términos de convergencia, pero pueden verse comprometidos en cuanto a la exploración del espacio de búsqueda [9].

2.3 ALGORITMOS DE OPTIMIZACIÓN

Existe un amplio catálogo de algoritmos de optimización derivados de la computación evolutiva. Luego, con el fin de delimitar el alcance de este trabajo se plantea comparar el desempeño de un algoritmo evolutivo conta un algoritmo de inteligencia de enjambres. Por lo anterior, se preseleccionan cinco algoritmos populares en la literatura y se describen a continuación para poder escoger de forma cualitativa los dos algoritmos que serán aplicados en programación de medio nivel.

2.3.1 NSGA-II (Non-dominated Sorting Genetic Algorithm)

Es un algoritmo genético representativo de los algoritmos evolutivos. Es ampliamente utilizado en la literatura para la resolución de problemas de optimización multi-criterio reales, como puede verse por ejemplo en [11] [12] [13] [14]. Es uno de los mejores métodos gracias a sus estrategias para mantener el elitismo y la diversidad en la búsqueda de soluciones óptimas, utilizando como analogía la selección natural darwiniana, de la cual se entiende que sólo los individuos más aptos sobreviven y se reproducen para dar paso a una nueva generación con aspectos mejorados. El pseudocódigo propuesto en [1] se abordará en la Sección 3.1.

2.3.2 MOPSO (Multi-Objective Particle Swarm Optimization)

Es un algoritmo representativo de la inteligencia de enjambres, popular en la literatura para abordar problemas de optimización multi-criterio, debido a su buen desempeño como puede verse en [15] [16] [17]. Es un algoritmo de inteligencia colectiva con un comportamiento de búsqueda similar al vuelo de los estorninos. Estas aves se mueven en bandadas y relacionan la dirección y velocidad de sus vuelos, de modo que un subgrupo de la población, respondiendo a un estímulo externo, transmite inmediatamente y con claridad el estado de su movimiento al resto del grupo. Cada individuo mantiene una susceptibilidad máxima a reaccionar frente a cualquier cambio en el vuelo de sus vecinos quienes, a su vez, reaccionan rápidamente a estímulos externos y transmiten a todo el grupo la información [18]. El pseudocódigo propuesto en [2] se abordará en la Sección 3.2.

2.3.3 SPEA (Strength Pareto Evolutionary Algorithm)

Este algoritmo recorre la zona factible del espacio de decisión utilizando operadores de cruce y mutación, haciendo selección de individuos a partir de la dominancia de Pareto, manteniendo en un repositorio a los miembros no dominados de la población con el fin de proporcionar elitismo en la búsqueda. El pseudocódigo propuesto en [19] se puede observar en el Algoritmo I.

ALGORITMO I

Algoritmo SPEA

```
1:  Inicializar una población:
2:      Generar aleatoriamente una población P.
3:      Crear un repositorio E de tamaño N.
4:      Asignar un nivel basado en la dominancia de Pareto – "ordenar".
5:  Para t = 1 hasta el máximo de iteraciones Hacer
6:      Extraer los individuos no-dominados de P y guardarlos en E.
7:      Determinar dominancia dentro del repositorio.
8:      Si el tamaño del repositorio es mayor que N Hacer
9:          Reducir el tamaño del repositorio con la técnica Clustering.
10:     Fin Si
11:     Asignar valor fitness a los miembros de P y E.
12:     Selección por Torneo Binario de una población P+E.
13:     Aplicar cruce y mutación.
14:     Actualizar población P.
15:  Fin Para
```

2.3.4 PAES (Pareto-Archived Evolution Strategy)

Este algoritmo realiza un procedimiento de búsqueda local para encontrar soluciones óptimas partiendo de una única solución. Dicha solución es tomada como un padre para generar a partir de ella una única descendencia utilizando un operador de mutación, luego compara padre e hijo según la dominancia de Pareto para crear un nuevo descendiente. El procedimiento de [20] se puede ver claramente en el pseudocódigo del Algoritmo II.

ALGORITMO II

Algoritmo PAES

```
1:  Inicializar:
2:      Generar aleatoriamente una solución C.
3:      Guardar C en un repositorio.
4:  Para t = 1 hasta el máximo de iteraciones Hacer
5:      Mutar C para generar una descendencia m.
6:      Evaluar fitness de m.
```

7: **Si** *C* domina a *m,* **Entonces**
8: Descartar *m*.
9: **De otro modo Si** *m* domina a *C* **Entonces**
10: *C = m.*
11: Agregar a *m* al *repositorio*.
12: **De otro modo Si** *m* es dominado por algún miembro del *repositorio* **Entonces**
13: Descartar *m*.
14: **De otro modo**
15: Evaluar dominancia entre *C*, *m* y el *repositorio*.
16: La solución no-dominada será la nueva *C*.
17: Agregar la nueva solución al *repositorio*.
18: **Fin Para**

2.3.5 VEPSO (Vector Evaluated Particle Swarm Optimization)

Este algoritmo de inteligencia de enjambres es similar al PSO original [21] con un cambio en el procedimiento de búsqueda. En lugar de utilizar un solo enjambre para optimizar simultáneamente el número de funciones objetivo planteadas en el problema de optimización, utiliza múltiples enjambres de tal modo que a cada enjambre se le asigna una función objetivo para ser evaluada. Al evaluar las funciones objetivo por separado, no se debe utilizar la dominancia de Pareto porque el problema se ha convertido en mono-objetivo. Luego, la mejor solución es la que tenga el menor valor fitness si se habla en términos de minimización. Para mayor entendimiento del proceso de búsqueda del algoritmo [22] se muestra su pseudocódigo en el Algoritmo III.

ALGORITMO III

Algoritmo MOPSO

1: **Fase de inicialización:**
2: **Para** i=1 hasta N **hacer**
3: Aplicar a *posición1$_i$.* un valor aleatorio en el rango [Xmin, Xmax].
4: Aplicar a *posición2$_i$.* un valor aleatorio en el rango [Xmin, Xmax].
5: Asignar ceros a *velocidad1$_i$.*
6: Asignar ceros a *velocidad2$_i$.*
7: Evaluar *fitness1$_i$.*
8: Evaluar *fitness2$_i$.*
9: *Mejor posición1$_i$* igual a *posición1$_i$.*
10: *Mejor posición2$_i$* igual a *posición2$_i$.*
11: *Mejor fitness1$_i$ igual a fitness1$_i$.*
12: *Mejor fitness2$_i$ igual a fitness1$_i$.*
13: **Fin Para**
14: Elegir el *mejor global* de cada enjambre
15: Guardar en *repositorio1* a *mejor posición1$_i$*
16: Guardar en *repositorio2* a *mejor posición2$_i$*
17: **Para** it=1 hasta el máximo de Iteraciones **Hacer**
18: **Para** i=0 hasta N **Hacer**

19: Elegir el *mejor global* de cada enjambre.
20: Calcular *velocidad1*$_i$ con *mejor global2.*
21: Calcular *velocidad2*$_i$ con *mejor global1.*
22: Actualizar *posición1*$_i$ *y posición2*$_i$.
23: **Fin Para**
24: Actualizar *mejor posición 1 y 2* según *repositorio*.
25: **Fin Para**

3 METODOLOGÍA

Hasta aquí, se puede entender que existen varios algoritmos de optimización multi-criterio y que cada uno propone un procedimiento de búsqueda diferente para encontrar soluciones aproximadas a valores fitness óptimos. Por lo anterior, si a un mismo problema de optimización se aplican varios algoritmos para su resolución, se podrá ver que la calidad de los resultados varía y que el desempeño de unos será mejor que el de otros. Luego, si una persona no experta en el tema busca resolver un problema, su primera dificultad será determinar el método que mejor se adapte a sus necesidades.

En la Sección 2.3. se destacan algunos de los algoritmos de optimización básicos más populares en el área de optimización, de los cuales tres son Algoritmos Evolutivos (NSGA-II, SPEA y PAES) y dos son basados en Inteligencia de Enjambres (MOPSO y VEPSO). La idea de este trabajo de grado es proveer una interfaz intuitiva que le permita a un usuario no necesariamente experto, comparar el desempeño de las dos ramas de la Computación Evolutiva seleccionando un algoritmo que represente cada una de ellas. En el estado del arte se pudo concluir que el NSGA-II es superior a sus pares SPEA y PAES. En [1] se puede evidenciar que el NSGA-II es superior a SPEA. El algoritmo PAES se descarta porque su método de búsqueda a partir de una única solución es inferior a los métodos basados en poblaciones. Luego, el NSGA-II se escoge como representante de los Algoritmos Evolutivos. Por otro lado, el MOPSO se escoge como representante de la Inteligencia de Enjambres debido a que el método VEPSO aborda los objetivos del problema con un enfoque mono-objetivo.

En las secciones 3.1 y 3.2, se detallan a mayor profundidad los procesos de búsqueda de los algoritmos NSGA-II y MOPSO respectivamente. En la sección 3.3, se definen los problemas de prueba que serán optimizados. En la sección 3.4, se definen las métricas de rendimiento que determinaran la calidad de las soluciones encontradas.

3.1 PROCESO DE BÚSQUEDA DEL NSGA-II

El pseudo código del NSGA-II se puede observar en el ALGORITMO IV. A continuación, se realiza una descripción de como este hace su proceso de búsqueda.

ALGORITMO IV

Algoritmo NSGA-II

```
1:  Inicializar una población:
2:      Generar aleatoriamente una población P.
3:      Evaluar la aptitud.
4:      Asignar un nivel basado en la dominancia de Pareto – "ordenar".
5:      Generar una población P siguiente:
6:          Selección mediante un torneo binario.
7:          Recombinación y mutación.
8:  Para i=1 hasta el número de generaciones Hacer
9:      Para la población padre e hijo Hacer
10:         Asignar un nivel basado en la dominancia de Pareto y ordenar.
11:         Generar el conjunto de frentes no dominados.
12:         Sumar soluciones a la siguiente generación, empezando por la
            primera jerárquica y utilizar el factor de apilamiento (crowding)
            en cada frente.
13:     Fin Para
14:     Seleccionar los puntos en el frente más bajo y que estén fuera de la
        distancia del factor de apilamiento.
15:     Crear la siguiente generación:
16:         Seleccionar mediante un torneo binario.
17:         Recombinación y mutación.
18: Fin Para
```

En el ALGORITMO IV las primeras siete líneas del pseudo código sirven para crear una población inicial. Primero el algoritmo crea de forma aleatoria una población de soluciones factibles P_0 de tamaño N. De esta población se escoge por torneo binario a un conjunto de padres que, a partir de un operador de cruce, tendrán una descendencia Q_0 también de tamaño N. El siguiente paso es combinar las dos poblaciones en una nueva población $R_0= P_0+Q_0$ y asignar un Frente a cada individuo de dicha población. Luego, los individuos se organizan según su Frente de forma ascendente y se escogen los N primeros individuos de R_0 para ser los padres P_1 de la siguiente generación. El proceso se repite en las siguientes generaciones hasta alcanzar un máximo de iteraciones según el *ciclo para* que inicia en la línea 8 del pseudocódigo. Para mantener la diversidad el NSGA-II aplica un factor de mutación en un número de individuos determinado por el usuario del algoritmo. En la Figura 6 se puede ver gráficamente el proceso de búsqueda, en donde t es el número de iteración o generación. Los términos de selección por torneo binario, cruce y mutación se profundizan más adelante.

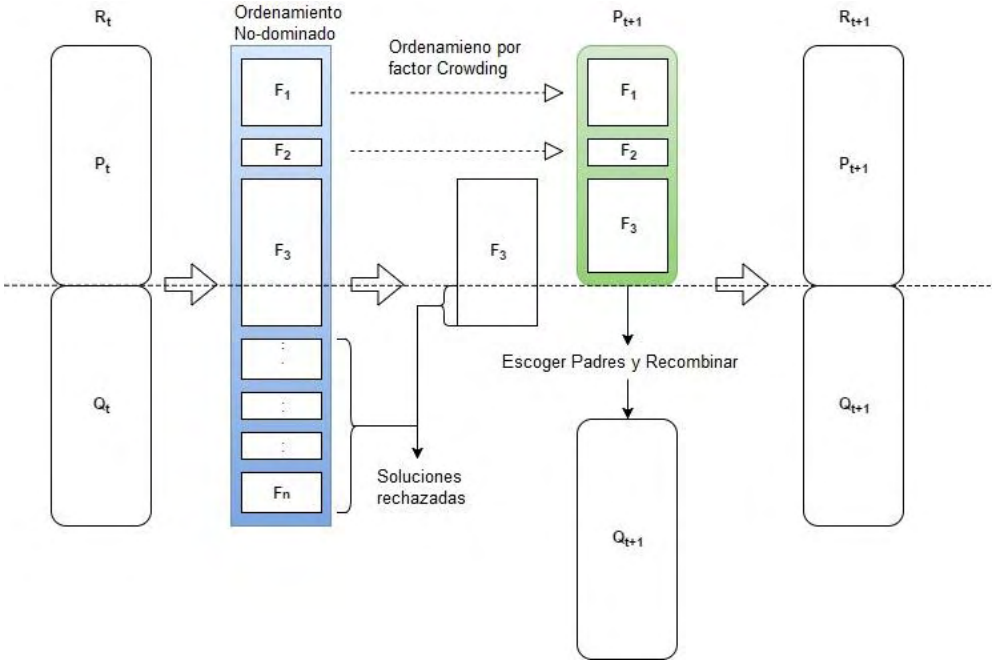


Figura 6. Proceso de búsqueda del algoritmo NSGA-II

3.1.1 Selección por Torneo Binario

Para mantener el elitismo, el algoritmo toma de forma aleatoria dos individuos de la población y los compara a partir del Frente o nivel de dominancia que tengan asignado. Sean dos individuos p_1 y p_2, se escoge un padre a partir de las siguientes condiciones:

- ***Si*** Frente de p_1 < Frente de p_2, ***Entonces*** p_1 se escoge como padre.
- ***Si*** Frente de p_2 < Frente de p_1, ***Entonces*** p_2 se escoge como padre.
- ***Si*** Frente de p_1 = Frente de p_2, ***Entonces*** el padre se escoge según el valor *Crowding Distance*.

El factor *Crowding Distance* o distancia de apilamiento, sirve para medir la distancia entre un individuo y sus vecinos más cercanos del mismo Frente. Este se obtiene calculando la longitud lateral promedio de la cuadricula que se forma utilizando los vecinos como vértices, esto se puede ver en la Figura 7.

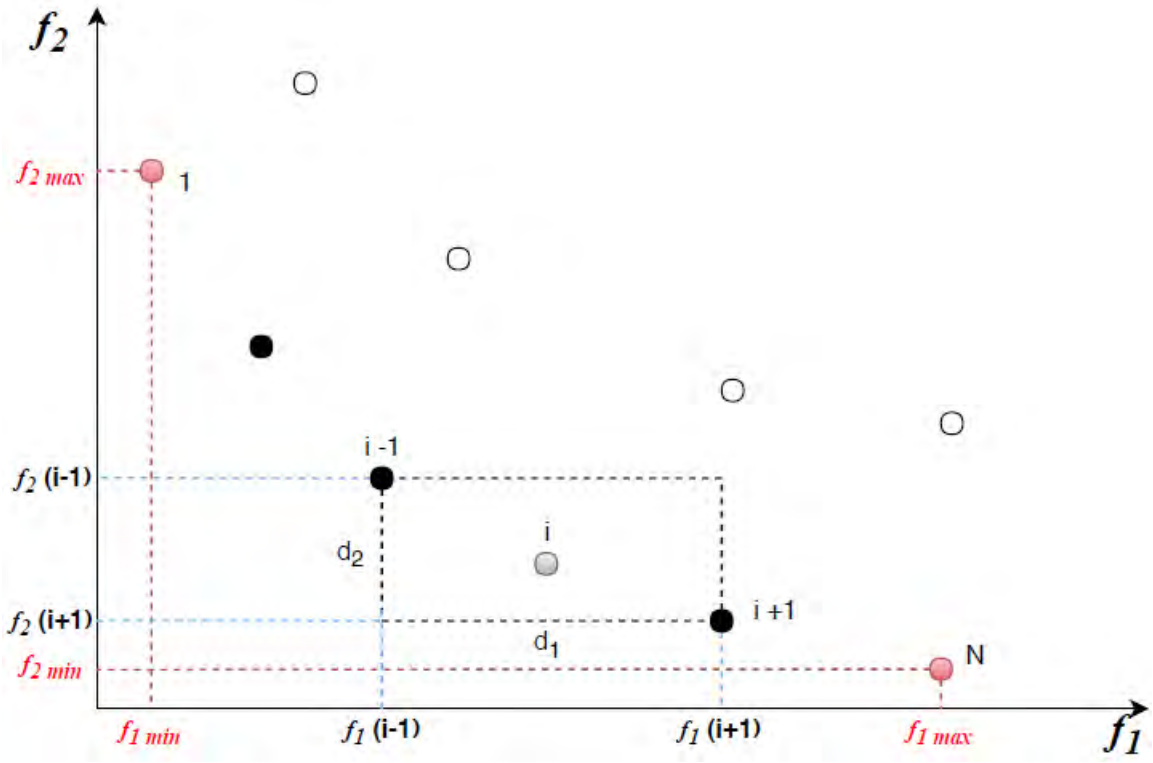


Figura 7. Distancia entre un individuo y sus vecinos más cercanos

La distancia d_k, que representa el largo de los lados de la cuadricula formada, se define formalmente en la siguiente ecuación:

$$d_k = |f_k(i+1) - f_k(i-1)| \tag{2}$$

Luego, a las soluciones ubicadas en los extremos del Frente se les asigna un valor *Crowding* infinito. A las soluciones ubicadas entre los máximos se les asigna un valor *Crowding* según la ecuación:

$$Crowding\ Distance_i = \sum_{k=1}^{k} \frac{{d_k}^i}{{f_k}^{max} - {f_k}^{min}} \tag{3}$$

Por último, el individuo p_i que tenga el menor valor *Crowding*, será elegido como padre. Esto permite evitar caer en óptimos locales.

3.1.2 Cruce de Individuos

Dos individuos previamente seleccionados por torneo binario tendrán un heredero a partir de la simulación de un cruce de especies. Dado un individuo $\boldsymbol{f_i(x)}$, el

algoritmo NSGA-II, toma de forma análoga las variables de decisión $\boldsymbol{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n})$ como los genes del individuo. La idea es que se dé una herencia de dos hijos con genes y_1 y y_2, de tal forma que cada hijo tenga más información genética de un padre que del otro. Lo anterior se logra con las siguientes ecuaciones:

$$y_1 = \alpha .* x_a + (1 - \alpha) .* x_b \quad (4)$$

$$y_2 = \alpha .* x_b + (1 - \alpha) .* \mathrm{x}_a \quad (5)$$

El factor $\boldsymbol{\alpha}$, es un arreglo que contiene $\boldsymbol{n}$ valores generados de forma aleatoria en el rango [0,1]. Los vectores $\boldsymbol{x_a}$ y $\boldsymbol{x_b}$ son los vectores de decisión correspondientes a los padres seleccionados. El producto punto permite a $\boldsymbol{\alpha}$ asignar diferentes valores de probabilidad a cada gen de los vectores $\boldsymbol{x_a}$ y $\boldsymbol{x_b}$. El valor $(\boldsymbol{1} - \boldsymbol{\alpha})$ es un valor de probabilidad complementario a $\boldsymbol{\alpha}$. Luego, para lograr una selección natural justa la probabilidad de que un gen o variable de decisión pase a la siguiente generación depende de un valor aleatorio.

3.1.3 Mutación

Para mantener la diversidad, el NSGA-II selecciona por torneo binario a un individuo de P_t para ser mutado. Para generar una mutación en el individuo, partiendo de $\boldsymbol{x_i} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n})$, se toma de forma aleatoria un gen x_n y se le asigna un nuevo valor dentro del rango de las variables de decisión definido en el problema. En la Figura 8 las variables A, B, C, D y E definen de una forma general los valores guardados en las variables de decisión $\boldsymbol{x_n}$. En este ejemplo la variable $\boldsymbol{x_3}$ se escoge de forma aleatoria y se cambia el valor C por A.

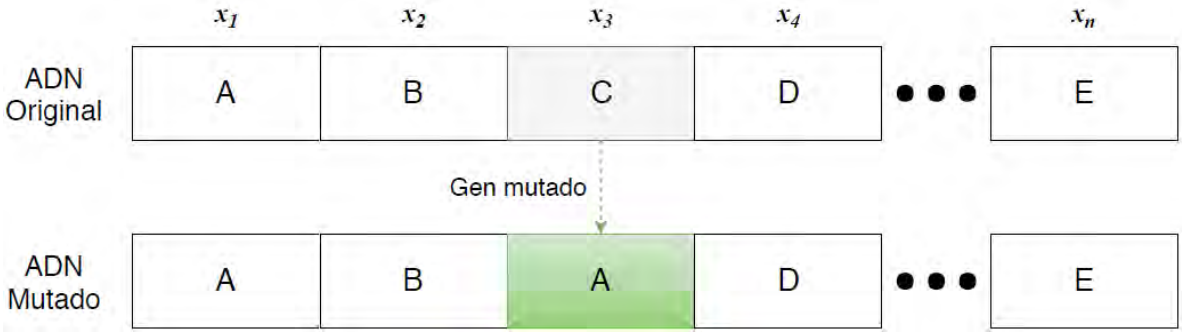


Figura 8. Mutación del Genoma

3.2 PROCESO DE BUSQUEDA DEL MOPSO

El seudocódigo del MOPSO se puede observar en el ALGORITMO V. A continuación, se describe como el algoritmo realiza su proceso de búsqueda.

ALGORITMO V

Algoritmo MOPSO

Fase de inicialización:
Para i=1 hasta N **hacer**
    Aplicar a *posición*$_i$ un valor aleatorio en el rango [Xmín, Xmáx].
    Asignar ceros a *velocidad*$_i$.
    Evaluar *fitness*$_i$.
    Mejor posición$_i$ igual a *posición*$_i$.
    Mejor fitness$_i$ igual a *fitness*$_i$.
Fin Para
Evaluar dominancia.
Guardar en un *repositorio* los individuos no dominados y asignarles índice de cuadrilla.
Fase de Búsqueda:
Para it=1 hasta el máximo de Iteraciones **Hacer**
    Para i=0 hasta N **Hacer**
        Escoger del *repositorio* al *mejor global.*
        Calcular *velocidad*$_i$.
        Calcular *posición*$_i$.
        Mantener individuos dentro del espacio de búsqueda y evaluar *fitness*$_i$.
        Aplicar factor de mutación.
        Si *posición*$_i$ domina a la mejor posición en su memoria **entonces**
        Pbest$_i$ = *posición*$_i$.
    Fin Para
    Evaluar dominancia.
    Actualizar *repositorio* y evaluar dominancia.
    Eliminar del *repositorio* los individuos dominados.
    Decrementar el factor *w.*
Fin Para

El MOPSO realiza la búsqueda de soluciones optimas imitando el comportamiento de una bandada en busca de alimento. Un conjunto de individuos se mueve a través del espacio de búsqueda utilizando una inteligencia colectiva de tal manera que el cambio de posición de cada individuo depende de la mejor posición en la memoria del individuo, la mejor posición encontrada por el colectivo y un peso de inercia. La posición de cada individuo se obtiene de las siguientes ecuaciones.

$$v_{id}(t+1) = w * v_{id}(t) + c_1 * r_1 * \left(pbest_{id} - x_{id}(t)\right)$$

$$+c_2 * r_2 * (gbest_d - x_{id}(t)) \qquad (6)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (7)$$

Donde $\boldsymbol{v_{id}}$ es el valor de velocidad del individuo i en la dimensión d en el tiempo t, c_1 es el valor de aprendizaje cognitivo, c_2 es el factor de aprendizaje global, r_1 y r_2 son valores aleatorios uniformemente distribuidos en el rango [0,1], $\boldsymbol{x_{id}}$ es la posición del individuo i en la dimensión d en el tiempo t, $\boldsymbol{pbest_{id}}$ es el mejor valor de posición en la dimensión d encontrada por el individuo i, y $\boldsymbol{gbest_{id}}$ es el valor de posición en la dimensión d del individuo de la población con la mejor posición. El valor w es importante para la convergencia del algoritmo. Se sugiere que c_1 y c_2 tomen valores en el rango [1.5, 2] y w en el rango [0.1, 0.5] [20]. El cambio de posición puede verse en la Figura 9.

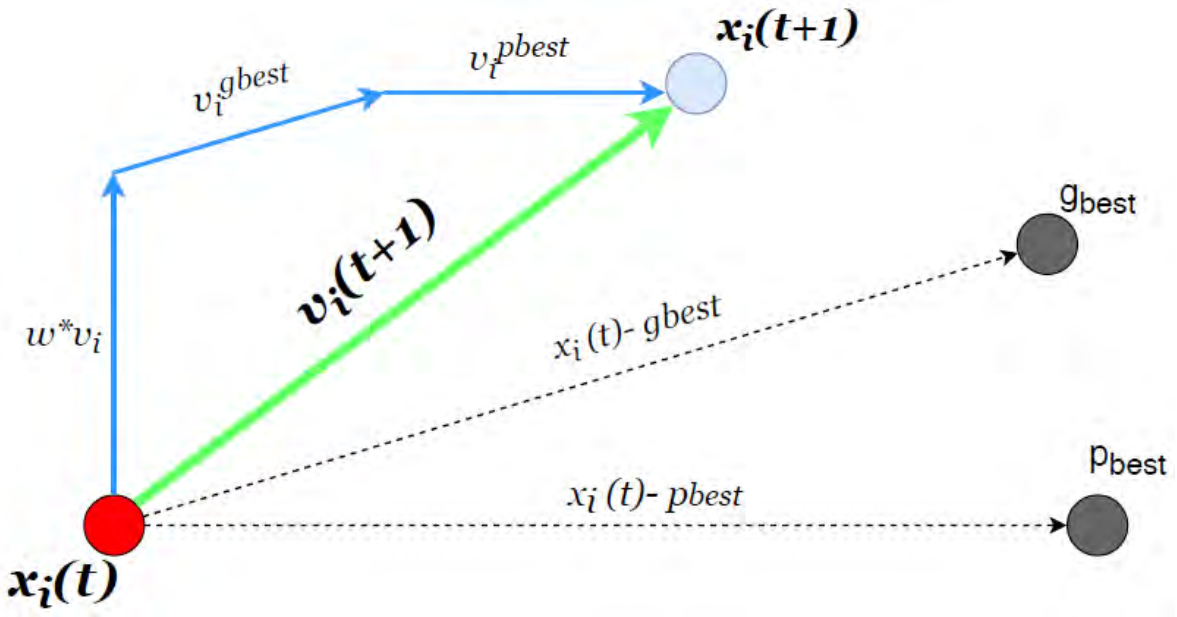


Figura 9. Cambio de posición de un individuo

Los vectores de decisión en este caso representan la posición de los individuos. En el primer *ciclo para* del seudocódigo se observa que el algoritmo inicia formando una población inicial de tamaño N y asigna a cada individuo valores de posición de forma aleatoria respetando el rango de las variables de decisión definido en el problema de optimización. Inicialmente toda la población tendrá un valor de velocidad cero debido a que en este punto no empieza el vuelo. Con los valores de posición y velocidad se calcula los *valores fitness* para toda la población. Todos los individuos guardan en $\boldsymbol{pbest_{id}}$ el valor de posición actual. En las líneas 9 y 10 del ALGORITMO V se extraen los miembros no dominados de la población y se los guarda en un *repositorio*. Luego, se escoge dentro de los individuos no dominados

a el líder que tiene la mejor posición de toda la población. Esta elección se hace simulando una rueda de fortuna (Roulete Wheel Selection) asignando valores de probabilidad según la densidad poblacional en las áreas en las que se ubican los individuos del *repositorio*. Dicha densidad poblacional se calcula con un mapa coordenado a partir de una cuadricula (Grid). Por último, el proceso se repite para cada iteración y se actualiza la posición de los individuos utilizando las ecuaciones (6) y (7). A continuación, se profundiza como se crea la cuadricula y como se hace la selección por ruleta.

3.2.1 Grid o Cuadricula

El algoritmo necesita un sistema coordenado para determinar qué tan pobladas están ciertas áreas del espacio objetivo por individuos del *repositorio*. Se define f_{1min} y f_{2min} como los *valores fitness* más pequeños de individuos no dominados. Se define a f_{1max} y f_{2max} como los *valores fitness* más grandes de los individuos del *repositorio.* Partiendo de los *valores fitness* anteriores, los vértices de la Cuadricula que será el perímetro del mapa coordenado se definen como:

$$l_{sup} = (f_{1\,max} + 0{,}1 * [f_{1\,max} - f_{1\,min}], f_{2\,max} + 0{,}1 * [f_{2\,max} - f_{2\,min}]) \quad (8)$$

$$l_{inf} = (f_{1\,min} + 0{,}1 * [f_{1\,max} - f_{1\,min}],\ f_{2\,min} + 0{,}1 * [f_{2\,max} - f_{2\,min}]) \quad (9)$$

Los vértices de la Cuadricula son l_{sup} y l_{inf}. Luego, la Cuadricula se divide en vecindarios con el fin de obtener un buen mapeo. El numero de vecindarios puede definirse según considere. En este trabajo se dividió la Cuadricula en 7 filas y 7 columnas para formar 49 vecindarios de igual tamaño dentro de los vértices ya definidos. El número de vecindarios parece suficiente para Frentes de Pareto definidos entre 0 y 1. En la Figura 10 se puede ver el mapa coordenado que se creó a partir de las ecuaciones (8) y (9). Al número de vecindario se le conoce como *Grid Índex o Índice de Cuadricula.*

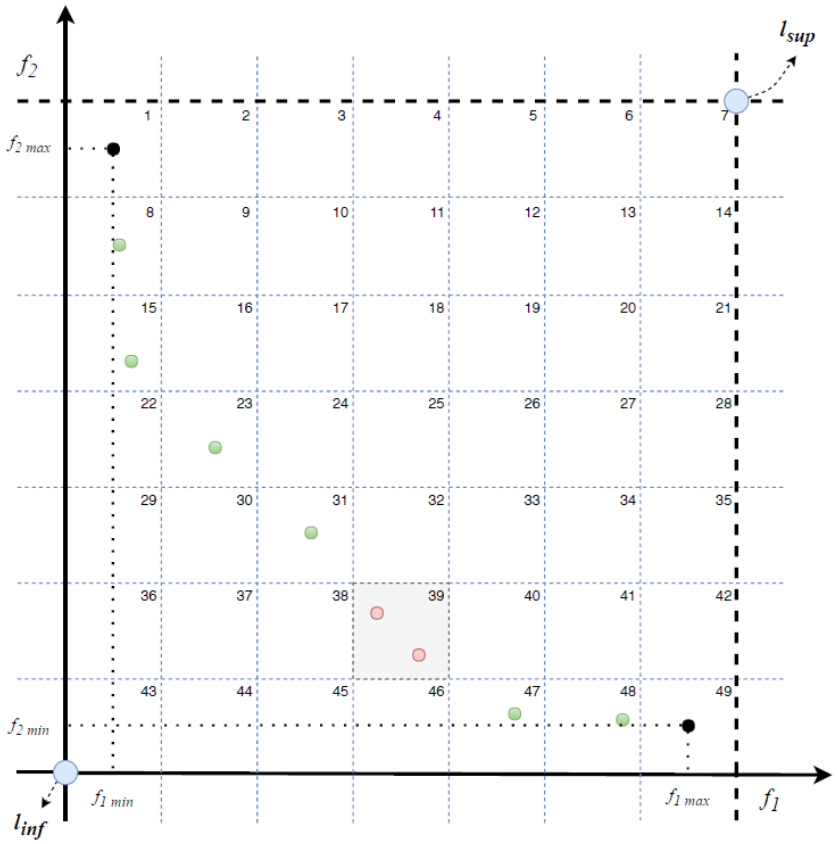


Figura 10. Grid o cuadricula como mapa de coordenadas

3.2.2 Roulette Wheel Selection

El líder o mejor global se obtiene de los individuos del repositorio, haciendo una selección aleatoria de tal manera que los individuos ubicados en vecindarios con mayor densidad poblacional tengan menos posibilidades de ser seleccionados. Se crea una rueda de la fortuna $pr_i = p_i / \sum_{j=1}^{a} p_j$, donde $p_i = e^{-\beta a}$, β es un factor de decremento y *a* es el número de *Grid Índex* o vecindarios definidos por el *repositorio.* En la Figura 10, por ejemplo, los individuos con *Índice de Cuadricula* = 39 tendrán menor valor de probabilidad. En la Figura 11, vemos un ejemplo de una ruleta con las probabilidades asignadas a cada individuo i.

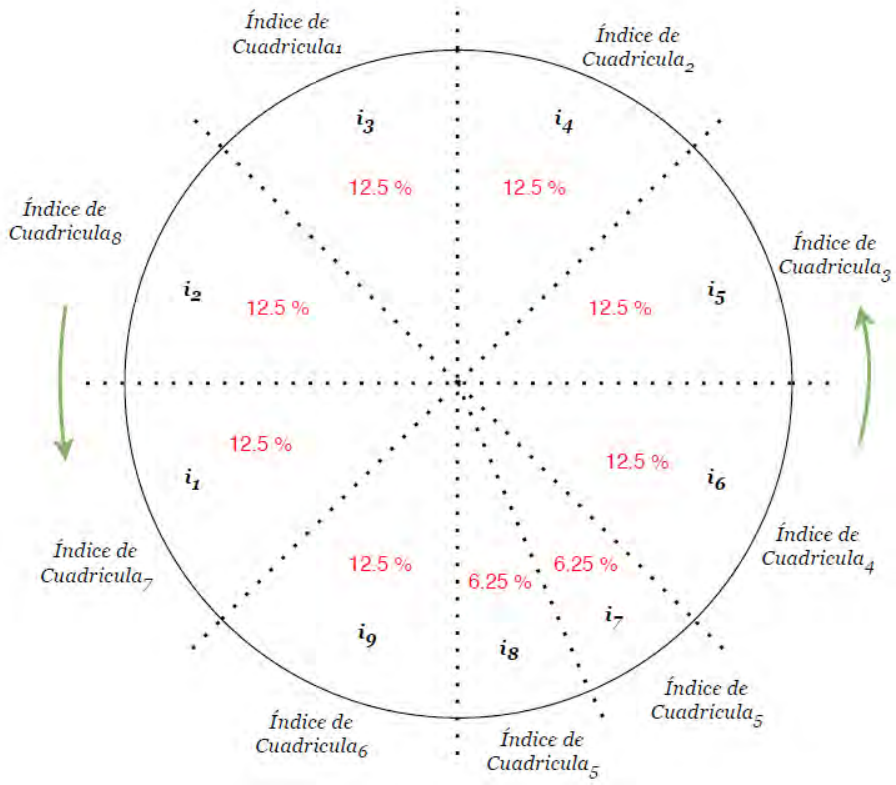


Figura 11. Roulette Wheel Selection

3.3 METODOLOGÍA DE COMPARACIÓN

En este punto se conoce cómo funcionan los dos algoritmos de optimización multiobjetivo MOPSO Y NSGA-II. Ahora, es necesario establecer pautas que permitan entender cómo se desempeñan los dos métodos frente ciertas funciones objetivo. Para evaluar su rendimiento se utilizan cuatro métricas que miden la convergencia al frente óptimo de Pareto.

3.3.1 Razón de Error (E)

Sea Y_{known} el conjunto de valores objetivo encontrados tras alcanzar el máximo de iteraciones definido y Y_{true} [23] el conjunto de soluciones dentro del Frente óptimo de Pareto, la razón de error se define como la proporción de los Y_{known} que se encuentran también dentro del Y_{true}. Entre más cercana sea la razón a 1, habrá

menos correspondencia entre el Frente de Pareto obtenido y el real. De esta manera un valor E=0 es ideal. Matemáticamente se representa como:

$$E \triangleq \frac{\sum_{i=1}^{N} e_i}{N} \tag{10}$$

$$e_i = \begin{cases} 0, & si\ un\ vector\ de\ Y_{known}\ esta\ en\ Y_{true} \\ 1, & en\ el\ caso\ contrario \end{cases} \tag{11}$$

3.3.2 Distancia Generacional (DG)

Esta métrica es un valor que representa qué tan lejos está cada individuo de Y_{known} del individuo más cercano de Y_{true} [21]. Matemáticamente se define como:

$$DG = \sqrt{\frac{\sum_{i=1}^{N} {d_i}^2}{N}} \tag{12}$$

donde d_i es la distancia euclidiana entre cada vector objetivo que pertenece a Y_{known} y su miembro correspondiente más cercano en el frente Pareto óptimo real Y_{true}. Entre más grande sea el valor de DG, más alejado estará el Y_{known} y el Y_{true}. De esta manera, un valor $DG = 0$ es ideal.

3.3.3 Espaciamiento (S)

Verifica la dispersión de los elementos del conjunto de Pareto *X* encontrado por el algoritmo. Conociendo los individuos en los extremos del conjunto, esta medida propone usar la varianza de la distancia entre vectores vecinos del conjunto *X* actual.

$$S \triangleq \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\bar{d} - d_i)^2} \tag{13}$$

Para dos funciones objetivo, $d_i = \min_j \left(\left| {f_1}^i(x) - {f_1}^j(x) \right| + \left| {f_2}^i(x) - {f_2}^j(x) \right| \right)$ es la distancia euclidiana entre soluciones consecutivas de Y_{known} $i, j = 1, 2, \ldots, n$. Donde n es la cantidad de individuos en. Un valor de S igual a cero significa que los individuos de Y_{known} están igualmente distribuidos, 1 es el caso opuesto [21].

3.3.4 Tiempo (T)

Se utiliza esta medida para determinar cuánto se demora el algoritmo en encontrar el conjunto de Pareto. La cuenta del tiempo se inicia desde el momento en el que se corre el algoritmo y termina cuando se detiene por completo. El valor de tiempo encontrado se da en segundos.

3.4 MARCO EXPERIMENTAL

Para poner a prueba el desempeño de los algoritmos de optimización, se utilizan las funciones de prueba propuestas por Zitzler, Deb y Thiele en [24]. Las funciones ZDT1, ZDT2, ZDT3, ZDT4 y ZDT6 permiten analizar el comportamiento de los algoritmos al optimizar 5 diferentes Frentes de Pareto. La idea de este trabajo de grado es dar al usuario diferentes formas de Frentes óptimos para que este pueda relacionar la optimización de estas funciones de prueba con un problema de optimización real. Para que los algoritmos corran es necesario definir los parámetros de evaluación necesarios para guiar la búsqueda de soluciones óptimas. La modificación interactiva de dichos parámetros ayudara a los usuarios a variar la calidad de los resultados.

3.4.1 Funciones de Prueba

3.4.1.1 Función ZDT1

Esta función tiene un frente de Pareto convexo y continuo. Se plantea minimizar las funciones f_1 *y* f_2 que se describen a continuación:

$$f_1(x) = x_1, \tag{14}$$

$$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \ ,$$

$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}},$$

$$f_2 = g(x) * h(f_1(x), g(x)). \tag{15}$$

Con $n = 30$ como el número de variables de decisión y x_i en el rango [0, 1]. En la Figura 12 se ve la forma del Frente de Pareto óptimos dado para $g(x) = 1$.

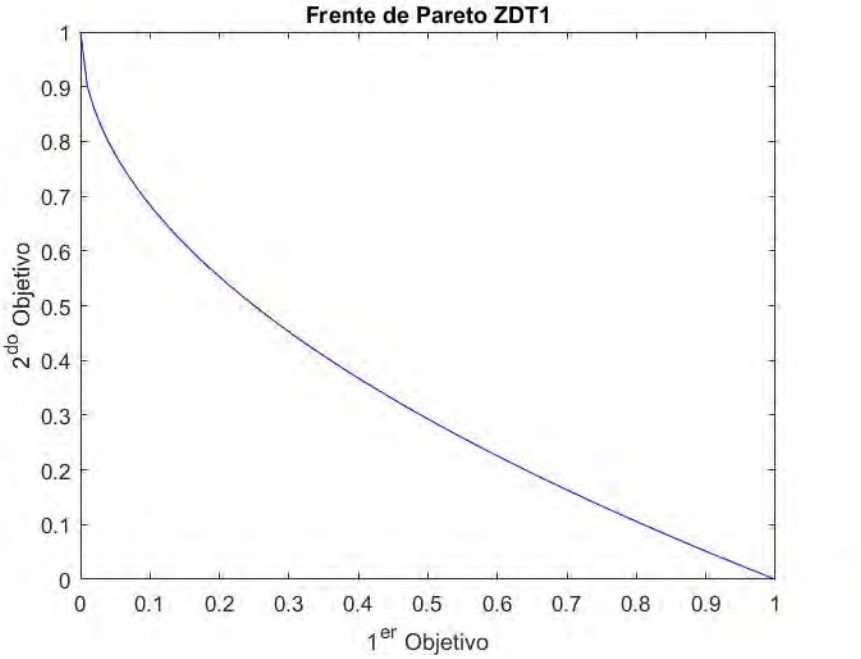


Figura 12. Frente de Pareto ZDT1

3.4.1.2 Función ZDT2

Esta función tiene un frente de Pareto no convexo y continuo. Se plantea minimizar las funciones f_1 *y* f_2 que se describen a continuación:

$$f_1(x) = x_1, \tag{16}$$

$$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \ ,$$

$$h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^2,$$

$$f_2 = g(x) * h(f_1(x), g(x)). \tag{17}$$

Con $n = 30$ como el número de variables de decisión y x_i en el rango [0, 1]. En la Figura 13 se ve la forma del Frente de Pareto óptimos dado para $g(x) = 1$.

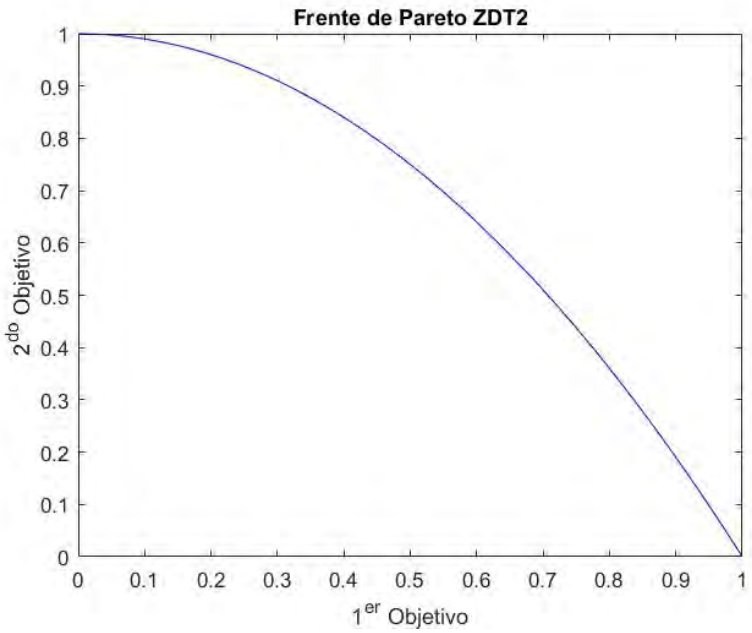


Figura 13. Frente de Pareto ZDT2

3.4.1.3 Función ZDT3

Esta función tiene un frente de Pareto discontinuo segmentado en cinco partes. Se plantea minimizar las funciones f_1 *y* f_2 que se describen a continuación:

$$f_1(x) = x_1 , \tag{18}$$

$$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i ,$$

$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g}\sin(10\pi f_1),$$

$$f_2(x) = g(x) * h(f_1(x), g(x)). \tag{19}$$

Con $n = 30$ como el número de variables de decisión y x_i en el rango [0, 1]. En la Figura 14 se ve la forma del Frente de Pareto óptimos dado para $g(x) = 1$. La función seno en h causa discontinuidad en el Frente de Pareto óptimo.

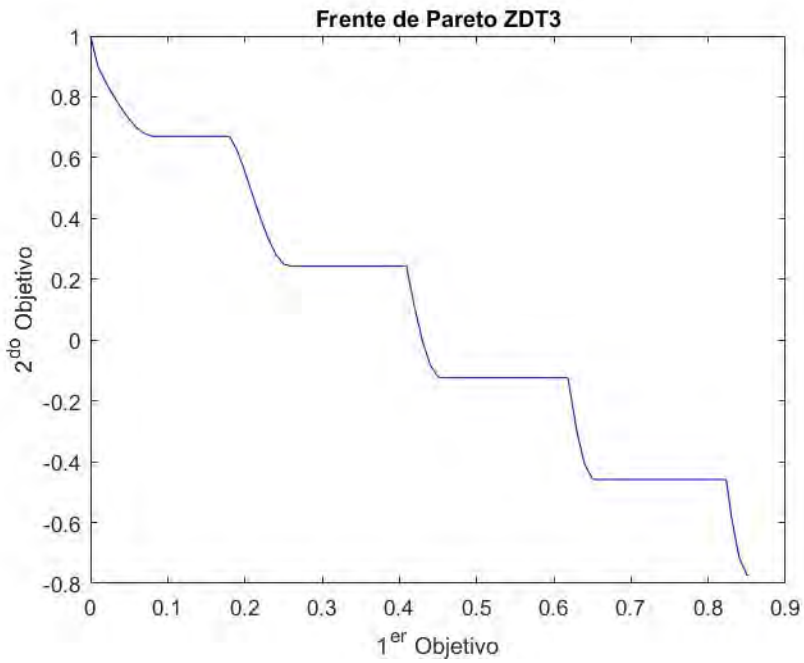


Figura 14. Frente de Pareto ZDT3

3.4.1.4 Función ZDT4

Esta es una función multimodal que tiene varios frentes de Pareto convexo y continuo. Se plantea minimizar las funciones f_1 *y* f_2 que se describen a continuación:

$$f_1(x) = x_1, \tag{20}$$

$$g(x) = 1 + 10(n-1) + \sum_{i=2}^{n}({x_i}^2 - 10\cos(4\pi x_i)),$$

$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}},$$

$$f_2(x) = g(x) * h(f_1(x), g(x)). \tag{21}$$

Con $n = 10$ como el número de variables de decisión, x_1 en el rango [0,1] y x_i en el rango [-5, 5] para $i = 2, \ldots, n$. El Frente de Pareto optimo global se da para $g(x) = 1$ y el mejor frente de Pareto optimo local se da para $g(x) = 1.25$. En la Figura 15 se observa la forma del Frente óptimo global. No todos los conjuntos de Pareto óptimos locales se distinguen en el espacio objetivo.

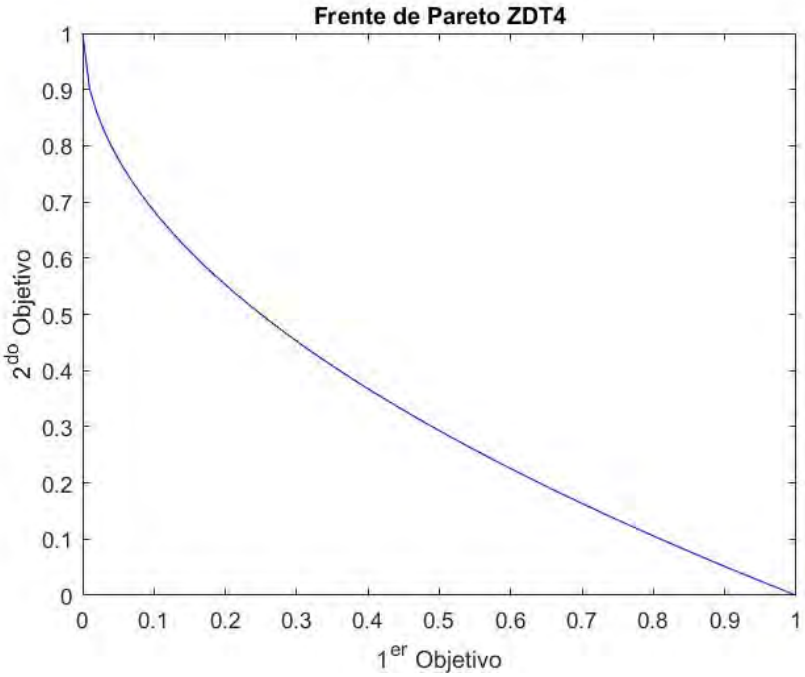


Figura 15. Frente de Pareto global ZDT4

3.4.1.5 Función ZDT6

Esta función tiene un frente de Pareto no convexo y continuo. Se plantea minimizar las funciones f_1 *y* f_2 que se describen a continuación:

$$f_1(x) = 1 - \exp(-4x_1) * \sin^6(6\pi x_1), \tag{22}$$

$$g(x) = 1 + 9\left[\frac{\sum_{i=2}^{n} x_i}{9}\right]^{0.125},$$

$$h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^2,$$

$$f_2(x) = g(x) * h(f_1(x), g(x)). \tag{23}$$

Con $n = 10$ como el número de variables de decisión y x_i en el rango [0, 1]. En la Figura 16 se ve la forma del Frente de Pareto óptimos dado para $g(x) = 1$.

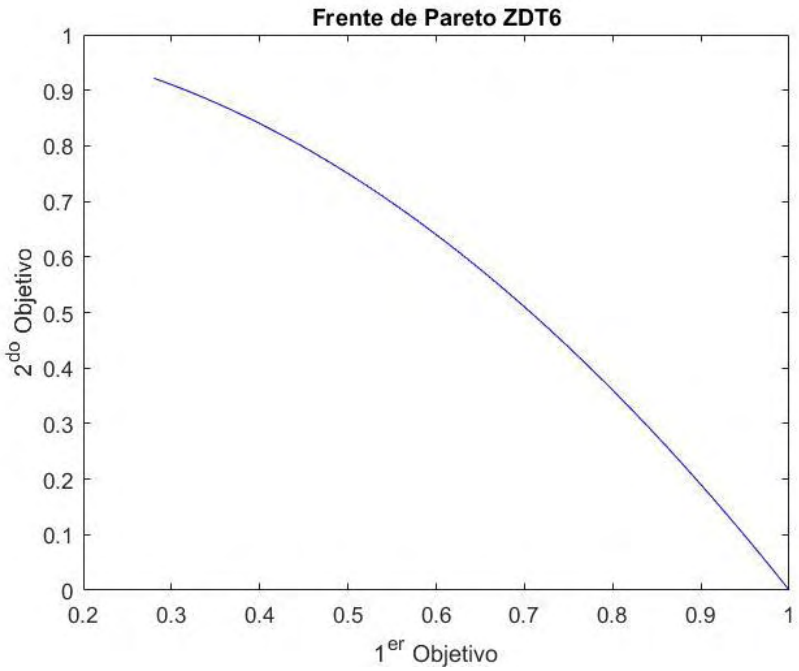


Figura 16. Frente de Pareto ZDT6

3.4.2 Parámetros de Evaluación

3.4.2.1 Parámetros NSGA-II

Como el algoritmo se basa en una población para la búsqueda de soluciones, se debe definir el tamaño N de dicha población. Es necesario un parámetro de parada que detenga la búsqueda, en este caso, un máximo de iteraciones $MaxIt$. Para crear la población Q_t primero se debe definir el número de *Padres* que serán usados para generar un grupo de *descendientes* de tamaño $nc = round\left(pc * \frac{N}{2}\right) * 2$, donde pc es la taza de cruce. El número de mutantes se define como $nm = round(pm_1 * N)$**,** donde pm_1 es la taza de mutación. El tamaño de Q_t será $nc + nm.$ En la Tabla I se muestran los parámetros utilizados para evaluar las 5 funciones de prueba antes definidas. En la literatura se recomienda que los parámetros pc y pm_1 se complementen de tal forma que su suma sea iguala 1. El tamaño de la población puede definirse de manera empírica observando el comportamiento del optimizador al aumentar o disminuir el numero de individuos. Los valores de la Tabla I se definieron por prueba y error utilizando la interfaz gráfica aquí desarrollada. Se deja un máximo de iteraciones como 500 porque para un numero mayor se obtienen valores muy similares y aumenta considerablemente el tiempo de ejecución.

TABLA I. Parámetros de Evaluación NSGA-II

PARÁMETROS	***ZDT1***	***ZDT2***	***ZDT3***	***ZDT4***	***ZDT6***
N	100	100	100	100	100
$MaxIt$	500	500	500	500	500
pc	0.67	0.63	0.63	0.67	0.67
pm_1	0.33	0.33	0.33	0.33	0.33

3.4.2.2 Parámetros MOPSO

Al igual que con el NSGA-II, se debe definir el tamaño N de los individuos que emprenderán vuelo en busca de soluciones óptimas y un parámetro de parada $MaxIt$. Como para el procedimiento de búsqueda el cambio de posición de los individuos es fundamental, se deben definir los parámetros $w, c1, c2$ de la ecuación (6) teniendo en cuenta las recomendaciones que se dieron en la sección 3.2.

Para generar diversidad, el algoritmo simula una turbulencia en el vuelo utilizando un operador de mutación. Cada Iteración del algoritmo, se asigna a todos los individuos una probabilidad de mutar (pm_2). Esta probabilidad debe tender a cero con el paso de las iteraciones, y se define bajo la ecuación:

$$pm_2 = \left(\frac{1-(it-1)}{MaxIt-1}\right)^{1/mu} \quad (24)$$

Donde it es la iteración actual y mu es la taza de mutación. En la Tabla II se muestran los parámetros utilizados para evaluar las 5 funciones de prueba antes definidas. En la Sección 3.2 se mencionan rangos recomendados para asignar valores a $w, c1, c2$ y pm_2. Los valores definidos en la TABLA II parten de un proceso de prueba y error utilizando la interfaz gráfica y las recomendaciones de los rangos.

TABLA II. Parámetros de Evaluación MOPSO

Parámetros	***ZDT1***	***ZDT2***	***ZDT3***	***ZDT4***	***ZDT6***
N_pop	100	100	100	100	100
Max_It	100	100	100	100	200
W	0.5	0.5	0.5	0.5	0.5
c_1	1.5	1.5	1.5	1.5	1.5
c_2	2	2	2.5	2	2.5
pm_2	0.1	0.1	0.5	0.1	0.5

3.4.3 Interactividad de la Interfaz Propuesta

La interactividad entre una herramienta computacional y el usuario es de gran importancia para lograr un mejor entendimiento de los resultados encontrados por los algoritmos de optimización. Por lo anterior, se plantea el desarrollo de una interfaz intuitiva que permita a su usuario modificar los parámetros de evaluación que guían los procedimientos de búsqueda de los algoritmos. Los cambios realizados en los parámetros modificarán la calidad del conjunto de Pareto que se encuentre al final de la ejecución del algoritmo utilizado. El usuario obtendrá información numérica y gráfica que le permitirá concluir sobre la calidad de los resultados encontrados y, por lo tanto, del desempeño del algoritmo.

4 RESULTADOS

En esta sección se describe la interfaz interactiva desarrollada para la comparación de los métodos de optimización multiobjetivo NSGA-II y MOPSO. Además, se discuten los resultados gráficos y numéricos obtenidos al usar la interfaz ingresando los parámetros de evaluación definidos en las Tablas I y II.

4.1 Interfaz Comparativa Interactiva

La interfaz se crea, partiendo de la necesidad de escoger un método de optimización teniendo en cuenta que su desempeño depende de la dificultad de las funciones objetivo y de los parámetros de evaluación que se utilicen. Esta, permite comparar los algoritmos de optimización NSGA-II y MOPSO, que representan dos ramas de la computación evolutiva conocidas como Algoritmos Genéticos e Inteligencia de Enjambres respectivamente. La interfaz fue implementada en el software Matlab por la facilidad que este da para la aplicación de algoritmos. Se utilizaron archivos de Matlab provenientes de [25] para el NSGA-II y de [26] para el MOPSO. Un usuario no experto en optimización multi-criterio podrá utilizar la interfaz de manera intuitiva y podrá obtener resultados numéricos de las medidas de desempeño mencionadas en la Sección 3. Podrá también ver gráficamente el conjunto de Pareto encontrado por cada algoritmo. La apariencia de la interfaz se puede observar en la Figura 17. Cuenta con un menú que despliega las Funciones de Prueba disponibles y un menú para escoger el algoritmo de optimización.

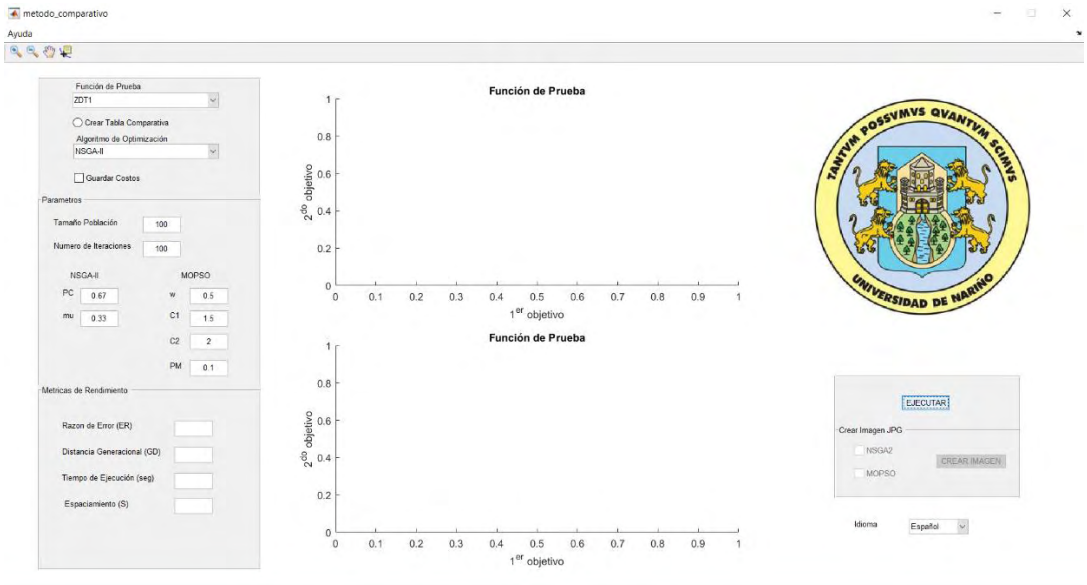


Figura 17. Interfaz desarrollada

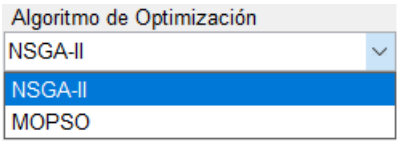


Figura 18. Menú selección algoritmo de optimización

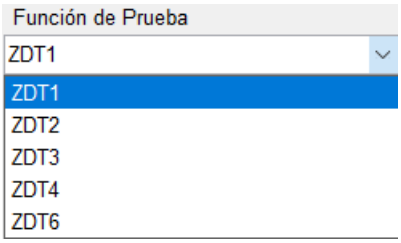


Figura 19. Menú selección función de prueba

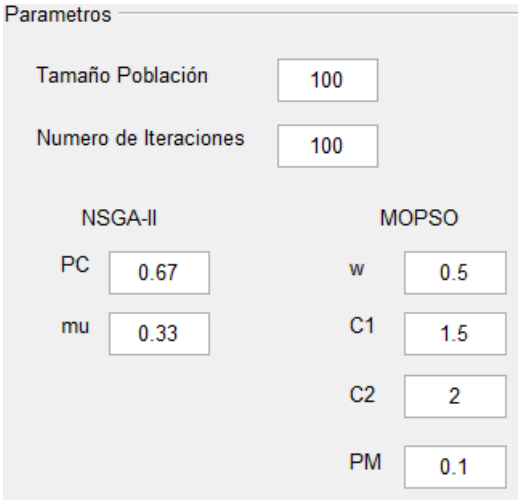


Figura 20. Parámetros de Evaluación predeterminados

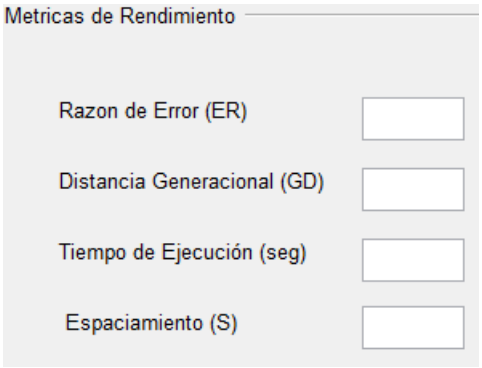


Figura 21. Medidas de Desempeño

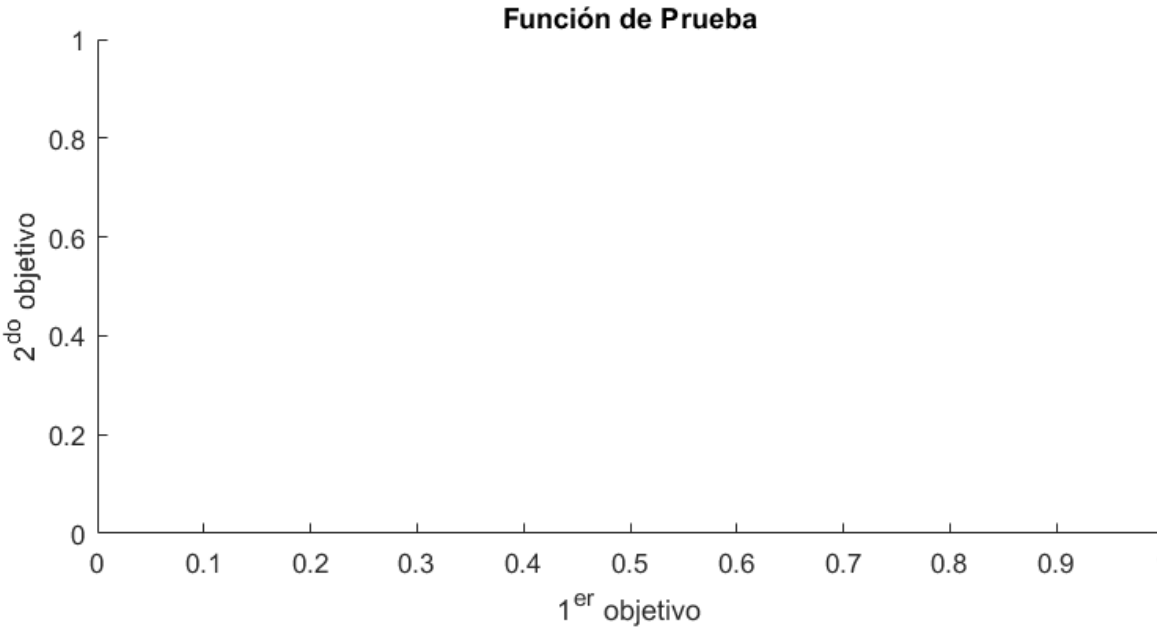


Figura 22. Plano espacio objetivo

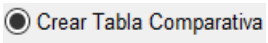


Figura 23. botón para crear tabla comparativa

Figura 24. Check para guardar los costos

Figura 25. Boton para ejecutar algoritmos y boton para guardar mapeo

La interfaz carga automáticamente un conjunto de parámetros de evaluación predeterminados para que un usuario sin conocimiento pueda ejecutar la interfaz y observar su funcionamiento. Los resultados de las métricas de rendimiento ubicados en la parte inferior izquierda de la interfaz se actualizan cada vez que esta se corre. Los conjuntos de Pareto encontrados por los algoritmos en cada iteración se grafican en los ejes disponibles en la interfaz.

Con cada ejecución, el algoritmo seleccionado encuentra un conjunto de Pareto diferente. Por esto, para que un usuario logre una buena conclusión sobre el desempeño de los algoritmos, este debe analizar los resultados obtenidos en varias ejecuciones. La interfaz da facilidades al usuario con la función “Crear Tabla Comparativa” que permite crear una tabla comparativa de forma automática en un archivo .xls, el cual contiene los resultados de las medidas de desempeño de los dos algoritmos aplicados a la misma función de prueba luego de 10 ejecuciones continuas. Si el usuario lo desea, puede guardar los conjuntos de Pareto encontrados en formato Excel seleccionando la función “Guardar Costos”. El usuario puede guardar el mapeo de los conjuntos de Pareto como un archivo JPEG usando el botón “Crear Imagen”. Para dar más facilidades, la interfaz cuenta con un menú que permite seleccionar el idioma de esta entre inglés y español.

4.2 Generación de Tablas Comparativas

Para evaluar el desempeño de los algoritmos de optimización, se ejecuta la interfaz comparativa utilizando la función “Crear Tabla Comparativa”, en cada una de las funciones de prueba antes definidas. Previo a iniciar el proceso de comparación, se realizaron varias ejecuciones de la interfaz para determinar los parámetros de

evaluación que presenten buenos resultados frente a cada función de prueba. En esta sección se muestran los resultados numéricos y gráficos encontrados para todas las funciones.

4.2.1 Resultados ZDT1

Como se puede ver en las Tablas I y II, el NSGA-II utiliza un numero de iteraciones mayor que el MOPSO debido a que los algoritmos evolutivos requieren más iteraciones para lograr buenos resultados, por lo tanto, de forma empírica se asignó un valor máximo de 500 iteraciones para el primer algoritmo. Un usuario de la interfaz podrá notar que los resultados encontrados por el NSGA-II con un máximo de iteraciones igual a 100, distan mucho del Frente de Pareto.

En la Tabla III, se evidencia un mejor desempeño por parte del MOPSO al optimizar un problema con Frente de Pareto convexo y continuo. Las métricas E y DG, muestran una muy buena aproximación al Frente real. También se evidencia que el algoritmo de inteligencia colectiva es mucho más rápido que su oponente. Cabe notar que el NSGA-II tiene una mejor dispersión que el MOPSO según la métrica S.

TABLA III. Resultados ZDT1

EJECUCIÓN	E_ NSGA2	E_ MOPSO	DG_ NSGA2	DG_ MOPSO	Tiempo_ NSGA2	Tiempo_ MOPSO	S_ NSGA2	S_ MOPSO
No.1	1,000	0,000	0,012	0,000	318,006	44,153	0,007	0,035
No.2	0,990	0,000	0,013	0,001	384,542	42,192	0,006	0,023
No.3	0,970	0,000	0,016	0,001	466,853	40,592	0,009	0,023
No.4	1,000	0,000	0,013	0,001	517,460	40,701	0,009	0,018
No.5	1,000	0,000	0,021	0,001	574,547	41,070	0,016	0,022
No.6	1,000	0,000	0,013	0,000	645,206	42,438	0,007	0,020
No.7	1,000	0,000	0,018	0,001	712,545	41,705	0,013	0,021
No.8	0,990	0,000	0,013	0,000	777,596	40,710	0,007	0,021
No.9	0,970	0,000	0,013	0,001	893,973	42,343	0,007	0,018
No.10	1,000	0,000	0,015	0,001	1196,094	39,570	0,008	0,019
PROMEDIO	***0,992***	***0,000***	***0,015***	***0,001***	***648,682***	***41,547***	***0,009***	***0,022***

En las Figuras 26 y 27, se puede observar el Frente de Pareto del problema ZDT1 como la línea celeste continua y la distribución de las soluciones y la cercanía de estas al Frente.

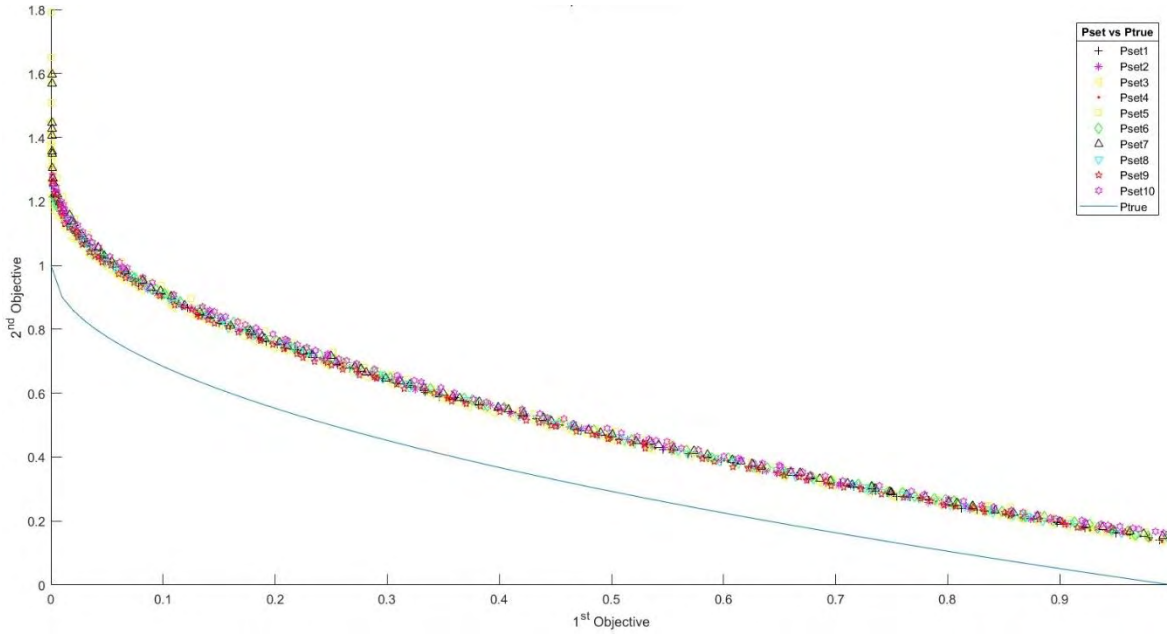


Figura 26. Soluciones encontradas por el NSGA-II luego de optimizar el problema ZDT1 en 10 ejecuciones vs Frente de Pareto convexo continuo

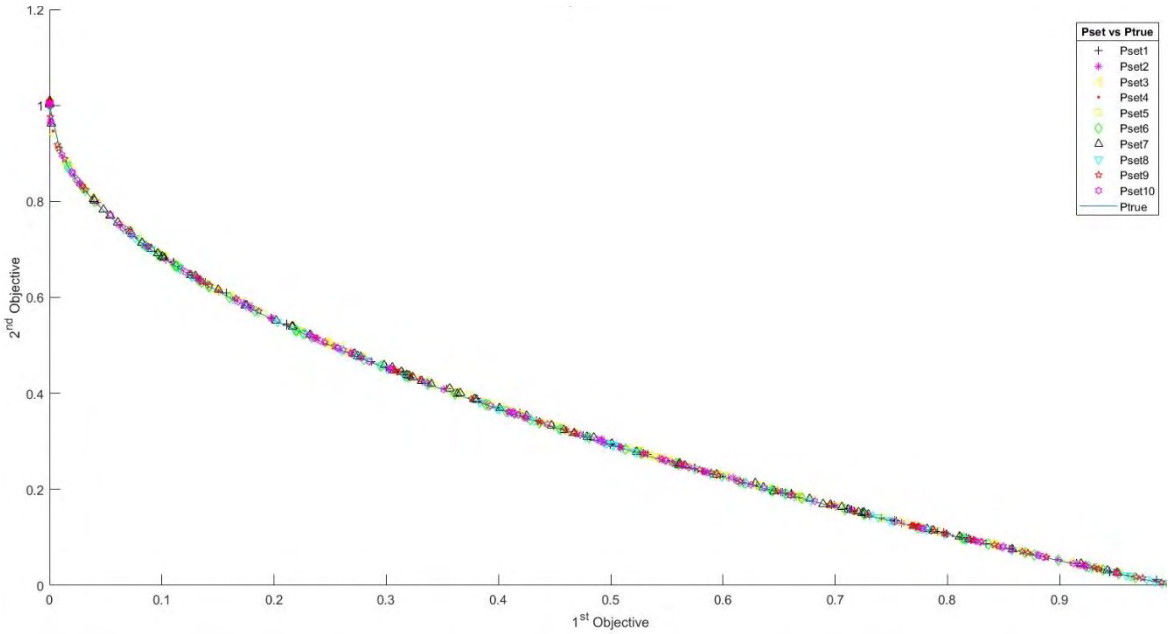


Figura 27. Soluciones encontradas por el MOPSO luego de optimizar el problema ZDT1 en 10 ejecuciones vs Frente de Pareto convexo continuo

4.2.2 Resultados ZDT2

El algoritmo MOPSO presenta un buen desempeño al optimizar el problema con un Frente de Pareto no convexo y continuo porque las medidas de desempeño logran su valor ideal como se puede observar en la Tabla IV. El valor 0 en las medidas de desempeño E y DG, indica que todas las soluciones encontradas en cada uno de los 10 procesos de optimización están contenidas en el Frente de Pareto del problema. El tiempo de ejecución del NSGA-II es muy alto y no logra superar la calidad de los resultados de su oponente, excepto por una mejor distribución de las soluciones según S.

TABLA IV. Resultados ZDT2

EJECUCIÓN	$E\_$ NSGA2	$E\_$ MOPSO	$DG\_$ NSGA2	$DG\_$ MOPSO	T_ NSGA2	T_ MOPSO	$S\_$ NSGA2	$S\_$ MOPSO
No.1	1,000	0,000	0,023	0,000	525,946	32,727	0,009	0,021
No.2	1,000	0,000	0,017	0,000	672,616	167,137	0,007	0,017
No.3	1,000	0,000	0,024	0,000	652,280	31,345	0,009	0,024
No.4	1,000	0,000	0,017	0,000	713,446	39,182	0,006	0,018
No.5	1,000	0,000	0,017	0,000	787,819	36,270	0,006	0,023
No.6	1,000	0,000	0,017	0,000	861,836	31,502	0,007	0,018
No.7	1,000	0,000	0,025	0,000	928,134	33,498	0,010	0,018
No.8	1,000	0,000	0,018	0,000	1000,959	30,929	0,005	0,019
No.9	1,000	0,000	0,025	0,000	1074,137	29,779	0,008	0,018
No.10	1,000	0,000	0,025	0,000	1144,182	31,050	0,009	0,019
PROMEDIO	***1,000***	***0,000***	***0,021***	***0,000***	***836,135***	***46,342***	***0,008***	***0,020***

En las Figuras 28 y 29, se puede observar el Frente de Pareto del problema ZDT2 como la línea celeste continua. También, se ve como están distribuidas las soluciones y que tan cercanas están al Frente.

El algoritmo NSGA-II se estanca en óptimos locales y no logra alcanzar el Frente óptimo de Pareto. Plantear unas ecuaciones matemáticas más robustas en los operadores de cruce y mutación podría ayudar al algoritmo a escapar de los óptimos locales.

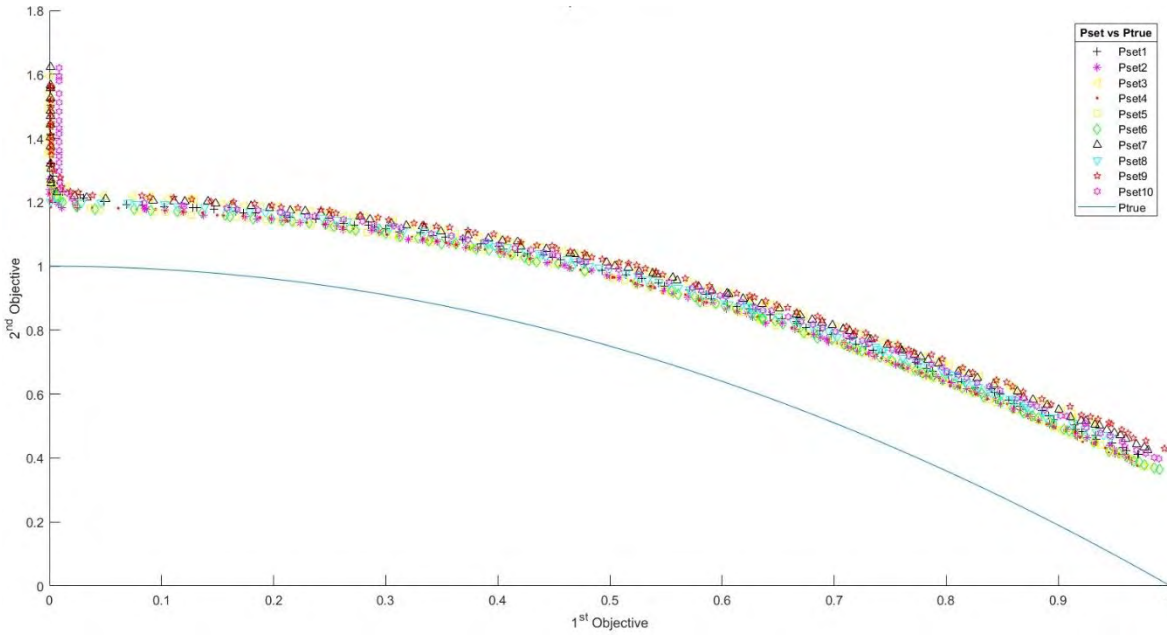


Figura 28. Soluciones encontradas por el NSGA-II luego de optimizar el problema ZDT2 en 10 ejecuciones vs Frente de Pareto no convexo continuo

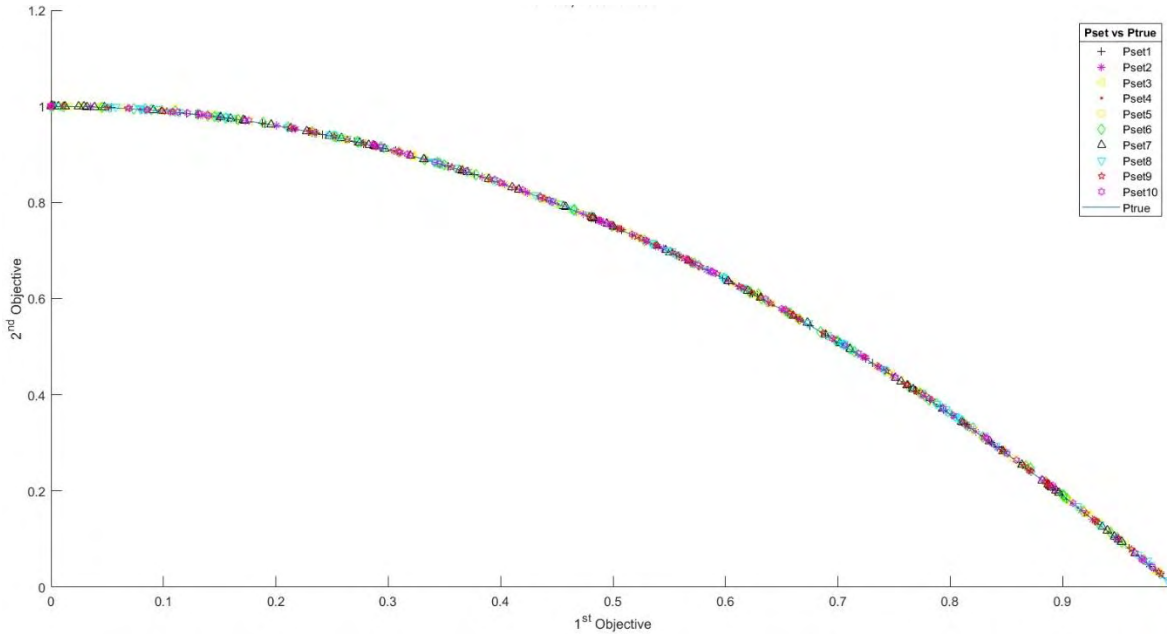


Figura 29. Soluciones encontradas por el MOPSO luego de optimizar el problema ZDT2 en 10 ejecuciones vs Frente de Pareto no convexo continuo

4.2.3 Resultados ZDT3

Al optimizar el problema ZDT3 con frente de Pareto discontinuo fragmentado en 5 partes, el MOPSO muestra una dificultad para encontrar el Frente de Pareto. Según las medidas E y DG se puede ver que, aunque en las ejecuciones 4,6 y 8 se obtiene una buena aproximación al Frente real, en el resto de las ejecuciones se obtienen soluciones distantes al Frente con una mala distribución.

Por otro lado, aunque las soluciones encontradas por el NSGA-II no logran alcanzar el Frente Optimo, estas tienden a ser constantes en calidad y tienen una mejor distribución que las encontradas por el MOPSO, pero requiere de un tiempo de ejecución mucho más largo. Para un análisis más detallado, ver la Tabla V.

TABLA V. Resultados ZDT3

EJECUCIÓN	$E\_$ NSGA2	$E\_$ MOPSO	$DG\_$ NSGA2	$DG\_$ MOPSO	T_ NSGA2	T_ MOPSO	$S\_$ NSGA2	$S\_$ MOPSO
No.1	0,950	0,910	0,026	0,018	1265,539	27,480	0,053	0,074
No.2	1,000	0,260	0,033	0,004	1333,715	29,983	0,077	0,085
No.3	0,890	1,000	0,020	0,113	1404,536	22,400	0,028	0,242
No.4	0,930	0,090	0,024	0,003	1480,041	28,327	0,028	0,039
No.5	0,890	0,870	0,019	0,025	1549,932	23,706	0,031	0,214
No.6	0,920	0,090	0,021	0,003	1613,282	32,330	0,029	0,032
No.7	0,930	0,880	0,020	0,037	1687,176	23,737	0,029	0,228
No.8	0,960	0,080	0,024	0,003	1774,601	31,559	0,055	0,035
No.9	0,980	0,370	0,023	0,006	1887,102	27,226	0,029	0,059
No.10	0,930	0,430	0,024	0,006	1958,618	30,228	0,042	0,084
PROMEDIO	***0,938***	***0,498***	***0,023***	***0,022***	***1595,454***	***27,698***	***0,040***	***0,109***

En las Figuras 30 y 31, se puede observar el Frente de Pareto del problema ZDT3 como la línea celeste continua, también la distribución de las soluciones y la cercanía de estas al Frente.

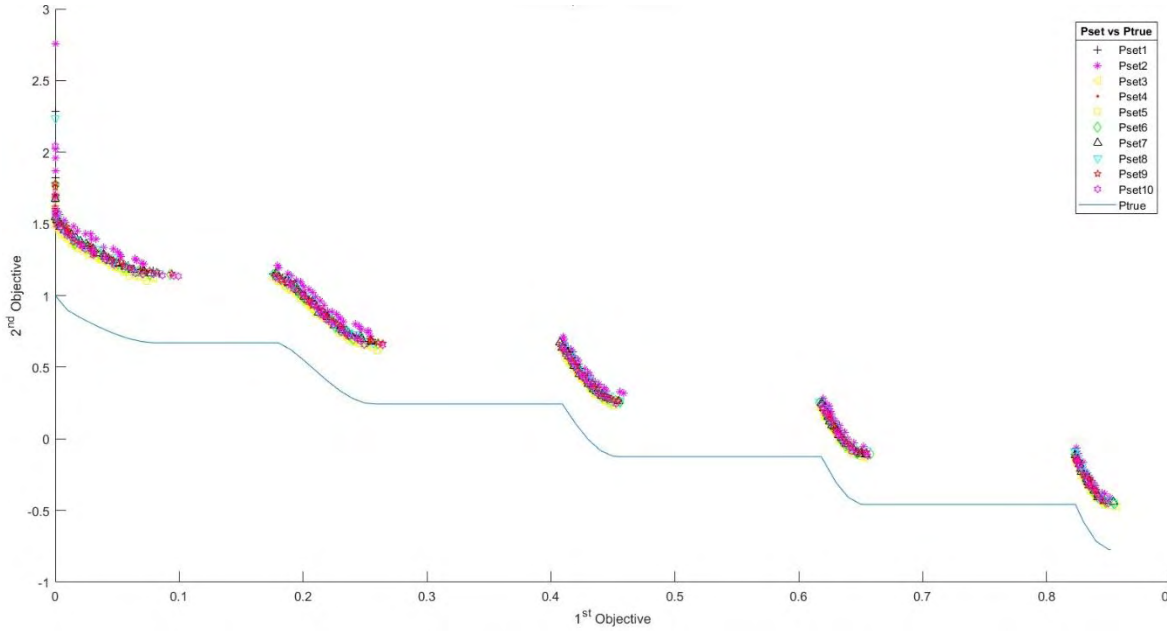


Figura 30. Soluciones encontradas por el NSGA-II luego de optimizar el problema ZDT3 en 10 ejecuciones vs Frente de Pareto discontinuo

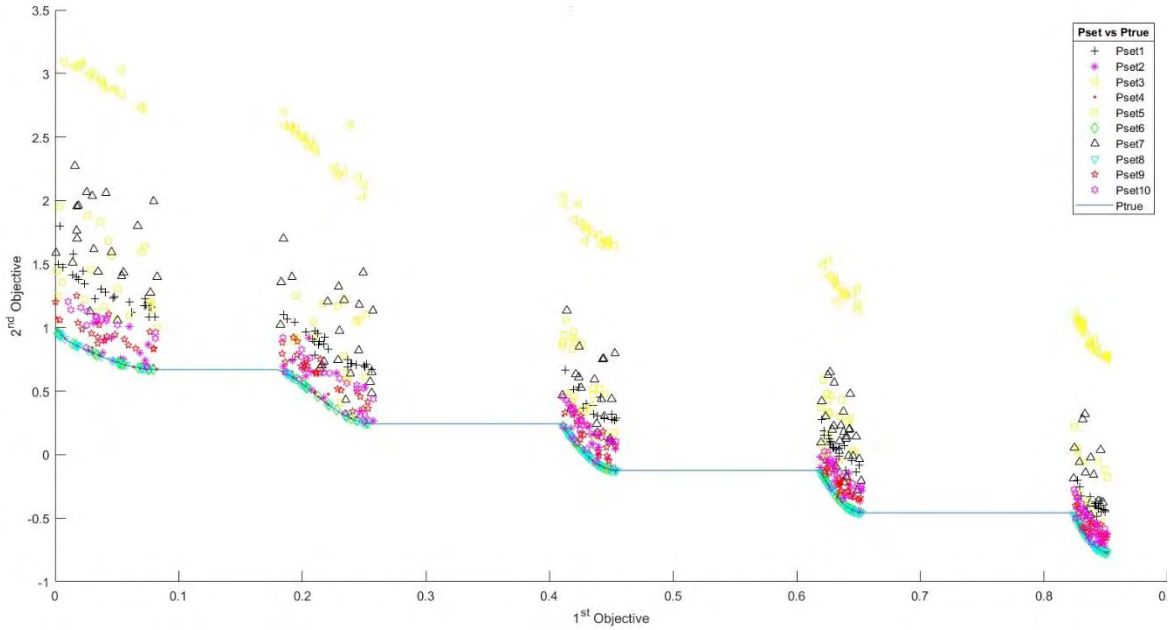


Figura 31. Soluciones encontradas por el MOPSO luego de optimizar el problema ZDT3 en 10 ejecuciones vs Frente de Pareto discontinuo

4.2.4 Resultados ZDT4

Al optimizar una función multimodal con varios Frentes de Pareto convexos y continuos, el algoritmo NSGA-II muestra un mejor desempeño. Aunque las soluciones encontradas por este no logran el Frente Pareto optimo, si logran buenas aproximaciones bien distribuidas.

Por otro lado, el MOPSO se pierde en el espacio de búsqueda y no logra optimizar el problema. Esto se debe a que los lideres no logran escapar de óptimos locales y no encuentran el Frente óptimo. Los resultados se pueden entender mejor en la Tabla VI.

TABLA VI. Resultados ZDT4

EJECUCIÓN	$E\_$ NSGA2	$E\_$ MOPSO	$DG\_$ NSGA2	$DG\_$ MOPSO	T_ NSGA2	T_ MOPSO	$S\_$ NSGA2	$S\_$ MOPSO
No.1	0,950	1,000	0,009	14,351	2354,344	50,643	0,008	25,177
No.2	0,570	1,000	0,105	14,871	2427,632	22,548	0,196	24,473
No.3	0,950	1,000	0,032	12,007	2726,507	24,206	0,031	22,601
No.4	0,700	1,000	0,008	11,917	2692,371	23,271	0,007	27,139
No.5	0,060	1,000	0,075	13,760	2686,197	22,537	0,312	27,502
No.6	1,000	1,000	0,071	14,515	2813,111	19,681	0,150	24,671
No.7	0,940	1,000	0,046	13,449	2878,507	19,849	0,073	22,431
No.8	0,190	1,000	0,071	12,985	2515,402	19,697	0,122	25,421
No.9	1,000	1,000	0,076	15,112	2566,152	20,092	0,257	28,002
No.10	1,000	1,000	0,065	11,162	2627,491	19,243	0,177	17,736
PROMEDIO	***0,736***	***1,000***	***0,056***	***13,413***	***2628,771***	***24,177***	***0,133***	***24,515***

En las Figuras 32 y 33, se puede observar el Frente de Pareto del problema ZDT4 para $g(x) = 1$ como la línea celeste continua. También, se ve como están distribuidas las soluciones y que tan cercanas están al Frente.

El MOPSO puede obtener mejores resultados si se hace que la escogencia de un líder sea mas selectiva. El NSGA-II muestra una buena capacidad para encapar de los óptimos locales, pero puede llegar a obtener mejores resultados si se mejoran las ecuaciones de cruce y mutación.

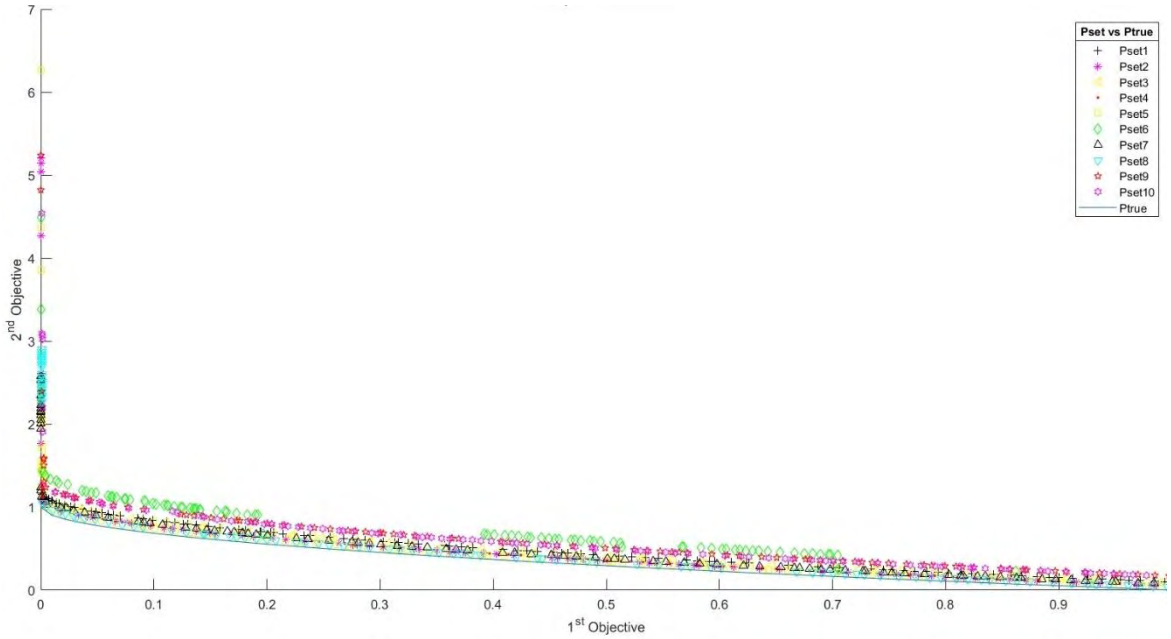


Figura 32. Soluciones encontradas por el NSGA-II luego de optimizar el problema ZDT4 en 10 ejecuciones vs Frente de Pareto convexo y continuo para g(x)=1

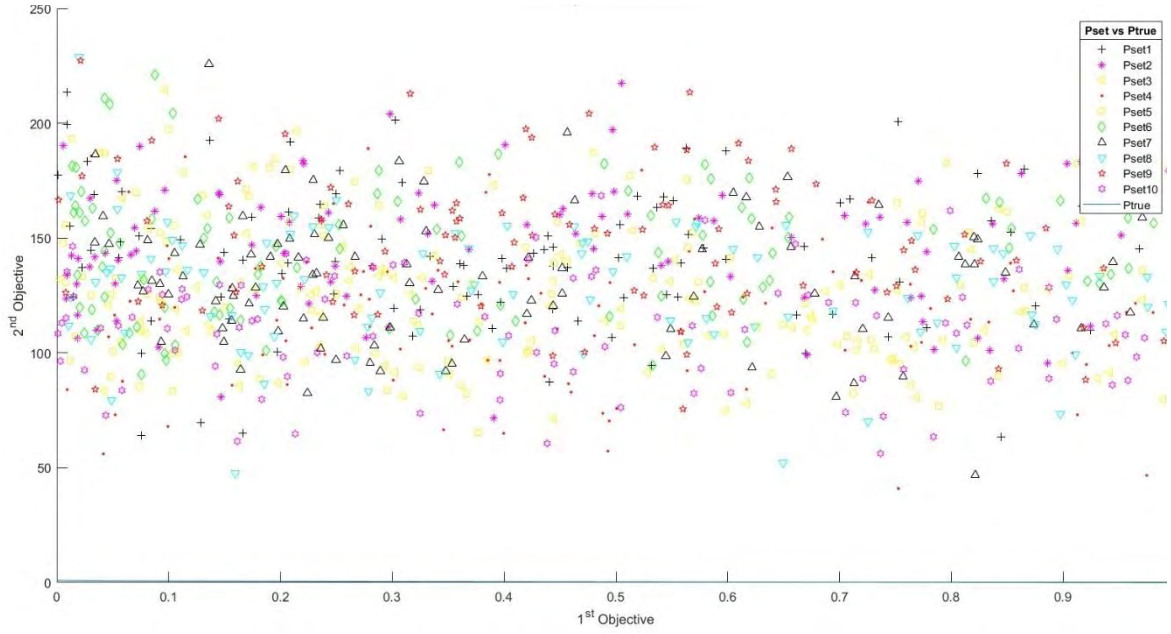


Figura 33. Soluciones encontradas por el MOPSO luego de optimizar el problema ZDT4 en 10 ejecuciones vs Frente de Pareto convexo y continuo para g(x)=1

4.2.5 Función ZDT6

Al optimizar el problema ZDT6 con frente de Pareto no convexo y continuo que tiene una no uniformidad en el espacio de búsqueda, el algoritmo MOPSO encuentra en 9 de 10 ejecuciones un conjunto de soluciones dentro del Frente de Pareto.

Por otro lado, el NSGA-II no logra buenas aproximaciones al Frente de Pareto y requiere un tiempo considerablemente alto para alcanzar resultados. La única ventaja sobre su oponente es que tiene una mejor distribución en los conjuntos de soluciones encontrados. Los resultados pueden observarse en la Tabla VII.

TABLA VII. Resultados ZDT6

EJECUCIÓN	$E\_$ NSGA2	$E\_$ MOPSO	$DG\_$ NSGA2	$DG\_$ MOPSO	T_ NSGA2	T_ MOPSO	$S\_$ NSGA2	$S\_$ MOPSO
No.1	1,000	0,000	0,200	0,000	515,210	63,310	0,037	0,025
No.2	1,000	0,000	0,205	0,000	581,132	70,026	0,047	0,022
No.3	1,000	0,000	0,182	0,000	637,318	63,332	0,017	0,020
No.4	1,000	0,000	0,169	0,000	729,697	68,171	0,023	0,020
No.5	1,000	0,000	0,195	0,000	826,544	62,126	0,071	0,021
No.6	1,000	0,000	0,185	0,000	1046,471	56,822	0,032	0,027
No.7	1,000	0,000	0,176	0,000	936,206	72,980	0,013	0,023
No.8	1,000	0,590	0,216	0,020	1088,454	43,768	0,072	0,220
No.9	*1,000*	*0,000*	*0,194*	*0,000*	*1057,488*	*63,993*	*0,034*	*0,021*
No.10	1,000	0,000	0,178	0,000	1135,490	65,859	0,022	0,028
PROMEDIO	***1,000***	***0,059***	***0,190***	***0,002***	***855,401***	***63,039***	***0,037***	***0,043***

En las Figuras 34 y 35, se puede observar el Frente de Pareto del problema ZDT6 como la línea celeste continua. También, se ve como están distribuidas las soluciones y que tan cercanas están al Frente.

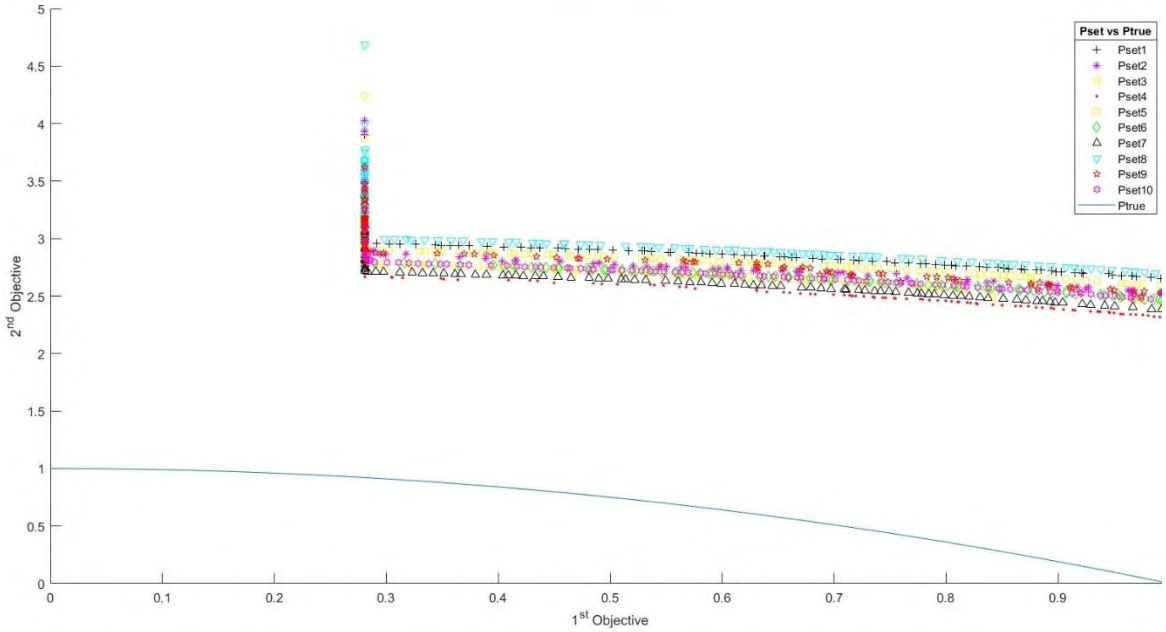


Figura 34. Soluciones encontradas por el NSGA-II luego de optimizar el problema ZDT6 en 10 ejecuciones vs Frente de Pareto no convexo y continuo

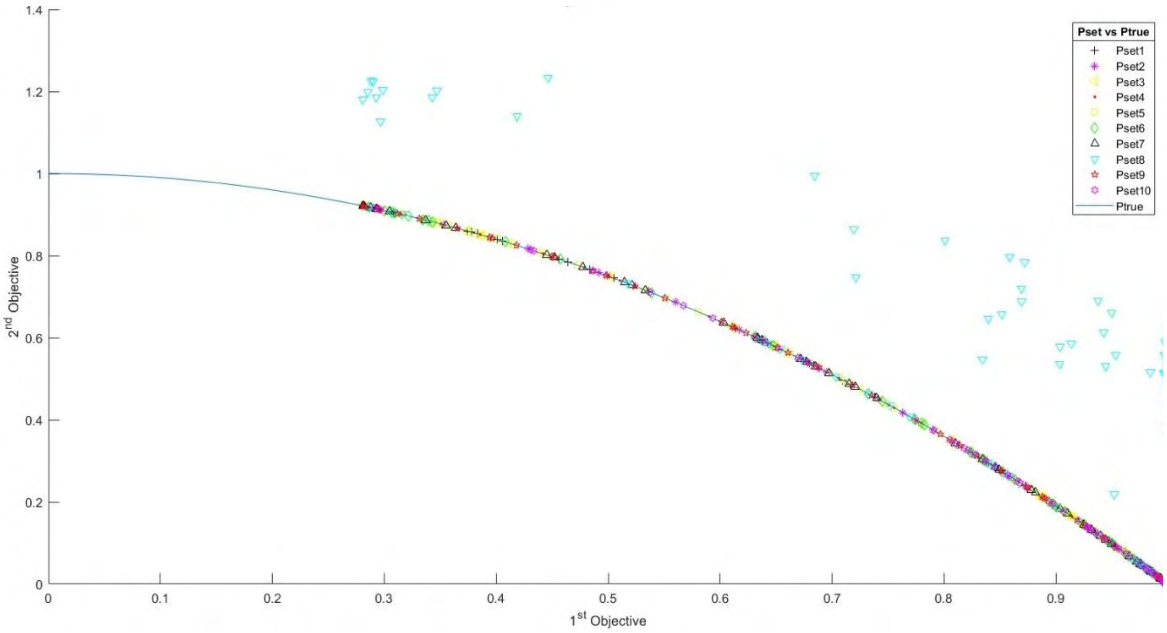


Figura 35. Soluciones encontradas por el MOPSO luego de optimizar el problema ZDT6 en 10 ejecuciones vs Frente de Pareto no convexo y continuo

5 CONCLUSIONES

Los resultados arrojados por un algoritmo de optimización pueden alcanzar diferentes niveles de calidad, dependiendo de la variación de los parámetros de evaluación. Por lo tanto, los estudios comparativos de métodos de optimización multi-criterio disponibles en la literatura ofrecen al lector un análisis limitado del desempeño de los algoritmos, al basar sus experimentos en parámetros de evaluación fijos.

El desarrollo de la interfaz comparativa interactiva ofrece la posibilidad de realizar fácilmente un proceso de optimización de forma intuitiva. La interactividad de la interfaz permite al usuario escoger el algoritmo de optimización y la función de prueba que desea optimizar según el frente de Pareto de interés, y establecer una comunicación hombre-máquina, de tal forma que, a través de varias entradas definidas como parámetros de evaluación, que pueden ser modificados a voluntad, se obtenga una respuesta gráfica dinámica y una respuesta numérica.

Utilizando un mapeo de las funciones objetivo, se puede observar el conjunto de soluciones encontradas en el espacio objetivo al termino de cada iteración, permitiendo observar el procedimiento de búsqueda de forma dinámica. El usuario puede medir de forma exacta el desempeño de los algoritmos al evaluar los resultados numéricos de las medidas de desempeño, que recaen sobre el conjunto final de soluciones encontradas. Por lo expuesto anteriormente, un usuario no necesariamente experto, tendrá un mayor entendimiento del proceso de optimización y será de mayor facilidad la escogencia del método adecuado de acuerdo con sus necesidades.

Utilizar la función “Crear Tabla Comparativa” de la interfaz, permite generar un análisis comparativo efectivo, al estar sustentado por el promedio de resultados encontrados tras varias ejecuciones de los algoritmos a comparar, aplicados a una misma función de prueba. La utilización de dicha opción frente a las funciones de prueba que muestra la interfaz permitió un análisis preliminar del comportamiento de los algoritmos NSGA-II y MOPSO. Las funciones de prueba ZDT1 y ZDT2 tienen Frentes de Pareto convexo y no convexo, respectivamente y dejan ver, en los gráficos y en los resultados numéricos de las tablas comparativas, que la mejor aproximación al frente Pareto real la da el algoritmo MOPSO. En la prueba ZDT6 también de frente de Pareto Convexo, pero con un espacio de búsqueda no uniforme [22], el MOPSO presenta una buena aproximación al frente optimo, pero puede caer en óptimos locales, mientras que el NSGA-II no alcanza al frente optimo, pero muestra uniformidad a lo largo de las ejecuciones realizadas sin caer en óptimos locales. El método de búsqueda evolutivo del NSGA-II muestra mejor desempeño que el MOPSO en las pruebas ZDT3 con frente optimo discontinuo y

ZDT4 con espacio de búsqueda multimodal. En el ZDT3, EL MOPSO puede lograr el frente optimo, pero la calidad de los resultados varía mucho cada vez que se ejecuta el algoritmo. Por otro lado, el NSGA-II no alcanza el frente optimo, pero logra una buena aproximación al mismo y muestra resultados uniformes tras varias ejecuciones. El método de búsqueda basado en inteligencia colectiva del MOPSO, tiene dificultades para ubicar el frente optimo en el espacio de búsqueda multimodal, debido a que la población no logra identificar un buen líder y predominan las mejores posiciones personales. Luego de aprovechar las funciones de la interfaz, un usuario no experto en optimización podrá tener información necesaria para definir cuál de los dos algoritmos se acomoda mejor a sus necesidades.

Como trabajo futuro, se plantea evaluar nuevas funciones de prueba que permitan ver el comportamiento de algoritmos de optimización al evaluar tres o más funciones objetivo simultáneamente. También, aumentar los métodos de optimización en la interfaz, para darle mayor variedad al usuario.

Teniendo en cuenta que el código base de la interfaz comparativa fue creado en MatLab, las modificaciones que se le hagan al mismo pueden generar errores en la ejecución. Pueden evidenciarse cambios o mejoras en los resultados de los algoritmos, si se hacen modificaciones en los métodos de cruce y mutación del NSGAII [23] y los métodos de selección de líder y mutación del MOPSO [24].

BIBLIOGRAFÍA

[1] K. Deb, A. Pratap y S. Agarwal, «A fast and elitist multiobjective genetic algorithm: NSGA-II,» *IEEE Transactions on Evolutionary Computation,* vol. 6, nº 7256659, pp. 182 - 197, 2002.

[2] C. A. Coello Coello y M. S. Lechuga, «MOPSO: a proposal for multiple objective particle swarm optimization,» *Proceedings of the 2002 Congress on Evolutionary Computation,* vol. 2, nº 7328831, pp. 1051- 1056 , 2002.

[3] D. H. Peluffo-Ordóñez, «A multi-class extension for multi-labeler support vector,» de *ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence*, Bruges (Belgium), 2014.

[4] A. Arciniegas, D. Imbajoa , I. Gustin, B. Mauricio y D. H. Peluffo Ordóñez, «Multi-Labeller classification Method based on Mixture of Classifiers and Genetic Algorithm Optimization».

[5] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, 2001.

[6] D. A. Van Veldhuizen y G. B. Lamont, «Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art,» *Evolutionary Computation,* vol. 8, pp. 125-147, 2000.

[7] B. Melián, J. A. Moreno Pérez y J. M. Moreno Vega, «Metaheurísticas: Una visión global,» *Inteligencia Artificial,* vol. 7, nº 019, 2003.

[8] L. V. Santana Quintero y C. A. Coello Coello, «Una introducción a la computación evolutiva y alguna de sus aplicaciones en Economía y Finanzas,» *Revista de Métodos Cuantitativos para la Economia y la Empresa,* vol. 2, pp. 3-26, 2006.

[9] J. J. Meza Álvarez, J. M. Cueva Lovelle y H. E. Espita Cuchango, «Revisión Sobre Algoritmos de Optimización Multi-Objetivo, Genéticos y basados en Enjambres de Partículas,» *Redes de Ingenieria,* vol. 6, pp. 54-76, 2015.

[10] M. A. Muñoz, J. A. López y E. F. Caicedo, «Inteligencia de enjambres: sociedades para la solución de problemas (una revisión),» *Ingeniería e Inestigación,* vol. 28, pp. 119-130, 2008.

[11] W. Y. Kwong, P. Y. Zhang, D. Romero y M. Morgenroth, «Multi-Objective Wind Farm Layout Optimization Considering Energy Generation and Noise

Propagation With NSGA-II,» *Journal of Mechanical Desing,* vol. 136, nº 091010, 2014.

[12] S. Kannan, S. Baskar y J. D. McCalley, «Application of NSGA-II Algorithm to Generation Expansion Planning,» *IEEE Transactions on Power Systems,* vol. 24, nº 10428863, pp. 454-461, 2009.

[13] S. Honda, T. Lgarashi y Y. Narita, «Multi-objective optimization of curvilinear fiber shapes for laminated composite plates by using NSGA-II,» *Composites Part B: Engineering,* vol. 45, pp. 1071-1078, 2013.

[14] S. Carlucci, G. Cattarin, F. Causone y L. Pagliano, «Multi-objective optimization of a nearly zero-energy building based on thermal and visual discomfort minimization using a non-dominated sorting genetic algorithm (NSGA-II),» *Energy and Buildings,* vol. 104, pp. 378-394, 2015.

[15] C. E. Robles-Rodriguez, C. Bideaux y S. E. Guillouet, «Multi-objective particle swarm optimization (MOPSO) of lipid accumulation in Fed-batch cultures,» *2016 24th Mediterranean Conference on Control and Automation (MED),* pp. 979-984, 2016.

[16] H. Borhanazad, S. Mekhilef, V. G. Ganapathy, M. Modiri-Delshad y A. Mirtaheri, «Optimization of micro-grid system using MOPSO,» *Renewable Energy,* vol. 71, pp. 295-306, 2014.

[17] M. A. Abido, «Multiobjective particle swarm optimization for environmental/economic dispatch problem,» *Electric Power Systems Research,* vol. 79, pp. 1105-1113, 2009.

[18] J. Marro, «Los estorninos de San Lorenzo, o cómo mejorar la eficacia del grupo,» *Revista Española de Física,* vol. 25, pp. 62-64, 2011.

[19] E. Zitzler, Evolutionary algorithms for multiobjective optimization: Methods and applications., 1999.

[20] J. Knowles y D. Corne, «The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation,» *Proceedings of the 1999 Congress on Evolutionary Computation,* vol. 1, pp. 98-105, 1999.

[21] J. Kennedy y R. Eberhart, «Particle swarm optimization,» *IEEE International Conference on Neural Networks,* vol. 4, pp. 1942-1948, 1995.

[22] K. E. Parsopoulos y M. N. Vrahatis, «Recent approaches to global optimization problems through particle swarm optimization,» *Natural computing,* vol. 1, pp. 235-306, 2002.

[23] D. A. van Veldhuizen y G. B. Lamont, «Multiobjective evolutionary algorithm test suites,» *Proceedings of the 1999 ACM symposium on Applied computing,* pp. 351-357, 1999.

[24] E. Zitzler, K. Deb y L. Thiele, «Comparison of Multiobjective Evolutionary Algorithms: Empirical Results,» *Evolutionary Computation,* vol. 8, pp. 173-195, 2000.

[25] M. Heris, «kalami.ir.,» Abril 2010. [En línea]. Available: http://www.kalami.ir..

[26] M. Heris, «Multi-Objective Particle Swarm Optimization (MOPSO),» 2015. [En línea]. Available: http://www.yarpiz.com..

[27] D. H. Peluffo Ordóñez, S. Murillo Rendón , J. D. Arias Londoño y G. Castellanos Domíngez, «A multi-class extension for multi-labeler support vector,» de *ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence*, Bruges (Belgium), 2014.

ANEXOS

Esta sección ha sido destinada a los resultados tangibles logrados con el trabajo realizado en esta tesis.

ANEXO 1. ARTÍCULO: International Conference on Information Systems and Computer Science (INSISCOS)

Este anexo contiene el artículo seleccionado para sustentación en modalidad ponente en el evento INSISCOS desarrollado en Quito del 24 al 26 de noviembre de 2016.

Estudio comparativo de NSGA-II y PSO como métodos de optimización multiobjetivo en problemas con frente optimo de Pareto convexo

Comparative study of NSGA-II and PSO as multiobjetive optimization methods in problems with a convex Pareto optimal front

David Francisco Dorado Sevilla
Facultad de ingeniería
Universidad de Nariño
Pasto, Colombia
david.dorado.sevilla@gmail.com

Fredy Alexander Guasmayán Guasmayán
Facultad de Ingeniería
Universidad Mariana
Pasto, Colombia
fguasmayan@umariana.edu.co

María Janeth Bravo Montenegro
Facultad de ingeniería
Universidad Mariana
Pasto, Colombia
mabravo@umariana.edu.co

Diego Hernan Peluffo Ordoñez
Facultad de Ingeniería y ciencias aplicadas
Universidad Técnica del Norte
Ibarra, Ecuador
dhpeluffo@utn.edu.ec

***Abstract*—in this study, two representative algorithms from evolutionary computation are applied to a simple multiobjective test problem with a convex optimal Pareto set in order to identify which one is more efficient finding from the searching space a set of optimal solutions. We describe how these methods inspired by nature works and how they try to find the Pareto set. The results are tested with the metrics generational distance an error rate to determine the efficiency.**

Keywords—NSGA-II; PSO; multi objective optimization; Pareto set.

I. RESUMEN

En este estudio, dos algoritmos representativos de la computación evolutiva se aplican a un problema de prueba multiobjetivo simple con un conjunto óptimo de Pareto convexo con el fin de identificar cuál de los dos es más eficiente encontrando desde el espacio de búsqueda un conjunto óptimo de soluciones. Se describe cómo funcionan estos métodos inspirados en comportamientos de la naturaleza y como estos intentan encontrar un conjunto Pareto óptimo. Los resultados se analizan utilizando las métricas de rendimiento Distancia Generacional y Razón de Error.

II. INTRODUCCIÓN

Cuando se habla de optimización, se entiende el logro de determinado objetivo haciendo uso de los recursos disponibles de forma eficiente. Hoy en día, la escases de recursos obliga a las personas a enfrentar los problemas buscando que su solución sea la mejor posible. El cerebro humano tiene la capacidad de resolver problemas triviales (Mono-Objetivo) de forma directa, pero al ir aumentando su complejidad también aumenta la dificultad para resolverlos, a mayor número de objetivos mayor complejidad. Para cada problema existe un gran espacio de búsqueda en donde se encuentra un conjunto de soluciones óptimas y existe un gran número de métodos que permiten llegar o tratar de acercarse a este conjunto. El desempeño de los métodos puede variar comparando uno respecto a otro y la complejidad de su aplicación puede ser de alto nivel. Para personas que pretenden realizar una optimización multiobjetivo de problemas suaves, puede ser difícil escoger un método eficiente y que tenga poca complejidad. Los algoritmos basados en mecanismos de la naturaleza son los más utilizados debido a su simplicidad.

Este artículo pretende comparar dos algoritmos comunes en la literatura, que permiten optimizar problemas multiobjetivo de forma rápida y sencilla. NSGA-II [4] y PSO [12] son representantes de las ramas de la computación evolutiva y son puestos a prueba en un problema con frente óptimo de Pareto convexo conocido como ZDT1 [10].

La sección 2 Describe el proceso de optimización y la forma que tiene cada algoritmo para realizar la búsqueda del conjunto que contenga las mejores soluciones posibles, además de las métricas de rendimiento que son utilizadas para determinar la eficiencia del optimizador. La sección 3 expone el problema a optimizar y los parámetros necesarios para la ejecución del algoritmo. En la sección 4 se evidencian los resultados obtenidos y en 6 se realizan las observaciones que surgieron a partir de los resultados.

III. OPTIMIZACIÓN MULTIOBJETIVO

En optimización multiobjetivo se busca encontrar un conjunto de soluciones que mejor representen las funciones objetivo, teniendo en cuenta las restricciones del problema. Un problema de optimización multiobjetivo puede definirse formalmente como (Van Veldhuizen, 1999) [1]:

$$\begin{aligned} &\text{Optimizar} && y = F(x) = \big(f_1(x), f_2(x), \dots, f_k(x)\big) \\ &\text{Sujeto a} && g(x) = \big(g_1(x), \dots, g_m(x)\big) \leq 0 \\ &\text{Donde} && x = (x_1, \dots, x_n) \in X \subseteq R^n \\ & && y = (y_1, \dots, y_n) \in Y \subseteq R^n \end{aligned} \quad (1)$$

En general se tiene un conjunto de *n* variables de decisión, un conjunto de *k* funciones objetivo y un conjunto de *m* restricciones, donde x es el vector de decisión en un espacio de decisión *X* e y es el vector objetivo en un espacio objetivo *Y*.

El problema de optimización consiste en encontrar un conjunto de soluciones óptimas de tal forma que no se pueda mejorar una sin deteriorar otra, es decir un conjunto Pareto óptimo. Dependiendo de las características del problema, optimizar puede significar maximizar o minimizar *F* (x). Por ejemplo, si se busca que los tiempos de fabricación de un objeto reduzcan hablaríamos de minimizar, o si se busca que un vehículo alcance la mayor velocidad posible se hablaría de maximizar, todo dependiendo de las necesidades del usuario.

Previo a tomar la decisión de que soluciones son las mejores y cuales deben descartarse se debe conocer el término de dominancia. Este concepto sirve de ayuda para el proceso de clasificación de aquellas soluciones que sean factibles teniendo en cuenta los objetivos y restricciones del problema [2]. En un contexto de minimización se expresa de la siguiente manera:

$$\begin{aligned} &u \succ v \ (u \text{ domina a } v) \text{si y solo si } f(u) < f(v) \\ &v \succ u \ (v \text{ domina a } u) \text{si y solo si } f(v) < f(u) \\ &u \sim v \ (u \text{ y } v \text{ no son comparables}) \text{ si y solo si} \\ &f(u) \preceq f(v) \wedge f(v) \preceq f(u) \end{aligned} \quad (2)$$

Teniendo en cuenta que el vector x ∈ *X*, se dice que x es no dominado respecto a un conjunto *V*⊆*X* si y solo si $x \succ v$ o $x \sim v, \forall v \in V$. De lo anterior se dice que si x es no dominado por ningún elemento de *V* entonces es una solución Pareto óptima. De lo anterior puede definirse un conjunto Pareto óptimo como:

$$X_{true} = \{x \in X \mid x \text{ es no dominado respecto a } X\} \quad (3)$$

Donde el frente optimo está formado por el conjunto de vectores objetivo $Y_{true} = F(X_{true})$

A. Metaheurísticas

Las técnicas de optimización se dividen en técnicas exactas (determinísticas) y en técnicas estocásticas (probabilísticas). Las metaheurísticas, que son una familia de algoritmos estocásticos que guiados por un conocimiento experto recorren en forma iterativa el espacio de búsqueda del problema, son el principal interés en este artículo.

Aquí se aborda la computación evolutiva que incluye aquellos algoritmos inspirados en mecanismos propios de la Naturaleza. Sus ramas principales son los algoritmos genéticos y la inteligencia de enjambres. Para nuestro estudio comparativo utilizaremos un representante de cada una de las ramas mencionada anteriormente, el algoritmo NSGA-II para la primera y el PSO para la segunda.

***1) NSGA-II* (*Nondominated Sorting Genetic Algorithm-II*): El NSGA-II [4] es la versión mejorada de [5] que elimina los problemas de alta complejidad computacional y la necesidad de especificar un parámetro de intercambio. Esta nueva versión, comienza creando una población de padres P_0 de manera aleatoria. Luego se realiza un ordenamiento no dominado en donde se determinan los frentes de Pareto. A cada función se le asigna un rango igual a su nivel de no dominancia (siendo 1 el mejor). Aquí se asume que el rango debe disminuir. Para crear la población de descendientes Q_0 de tamaño N, se utilizan la selección por torneo, cruzamiento y mutación.**

Luego de la primera generación el procedimiento es diferente, pues para lograr el elitismo se compara la población actual con las mejores soluciones no dominadas encontradas anteriormente. Así se tiene lo siguiente:

- Se forma una población combinada $R_t = P_t \cup Q_t$. luego esta se clasifica de acuerdo a la no dominancia y se encuentran los frentes de Pareto.
- Se crea una población P_{t+1}, a partir de los frentes encontrados anteriormente.
- Para escoger exactamente una población de N miembros se clasifican las soluciones del último frente

usando el comparador de apilamiento que, al comparar dos soluciones, escoge la de menor rango y si son de rangos iguales se escoge la de la región menos poblada, esto se hace en orden descendiente. El rango y la distancia de apilamiento se calculan durante el proceso de formación de P_{t+1}.

- Luego de P_{t+1}, se usa la selección cruce y mutación para crear la nueva Q_{t+1}. Las dos son de tamaño N.

La diversidad entre soluciones no-dominadas se introduce usando el procedimiento comparación de apilamiento, que se usa en la selección por torneo y durante la fase de reducción de la población. El pseudo código del NSGA-II se muestra en Algoritmo 2.1 [4].

Algoritmo 2.1 NSGA-II

```
1:  Inicializar una población:
2:      Generar aleatoriamente una población.
3:      Evaluar la aptitud.
4:      Asignar un nivel basado en la dominancia de Pareto
        – "ordenar".
5:      Generar una población siguiente:
6:          Selección mediante un torneo binario.
7:          Recombinación y mutación.
8:  Para i=1 hasta el número de generaciones hacer
9:      Para la población padre e hijo hacer
10:         Asignar un nivel basado en la dominancia de
            Pareto y ordenar.
11:         Generar el conjunto de frentes no dominados.
12:         Sumar soluciones a la siguiente generación,
        empezando por la primera jerárquica y
            utilizar el factor de agrupamiento (crowding)
            en cada frente.
13:     Fin Para
14:     Seleccionar los puntos en el frente más bajo y que
        estén fuera de la distancia del factor de
        agrupamiento.
15:     Crear la siguiente generación:
16:       Seleccionar mediante un torneo binario.
17:           Recombinación y mutación.
18: Fin Para
```

2) PSO (Particle Swarm Optimization):

La inteligencia colectiva es un paradigma que agrupa técnicas de inteligencia artificial basadas en el estudio del comportamiento colectivo observado en la naturaleza. PSO involucra un conjunto de agentes o partículas conocido como bandada (swarm) que recorre el espacio de soluciones tratando de localizar regiones prometedoras. El recorrido se hace siguiendo una trayectoria definida por la velocidad de la partícula y la memoria del mejor valor encontrado por la misma y el mejor valor encontrado por una partícula de la población. La atracción o fuerza que dirige a dichas regiones se conoce como presión social. Las partículas se interpretan como posibles soluciones del problema de optimización y son representadas como puntos *n*-dimensionales en el espacio de búsqueda.

Así, cada partícula está formada por cinco componentes:

- $\vec{x}$ describe la ubicación de la partícula en el espacio de soluciones.
- Un valor objetivo, se obtiene del cálculo de la función objetivo.
- $\vec{v}$ representa la velocidad de la partícula.
- $pbest$ es el mejor valor objetivo (fitness) encontrado por la partícula hasta el momento.
- $gbest$ es el mejor valor objetivo (fitness) encontrado por una partícula del swarm.

En [6], en el proceso de búsqueda las partículas ajustan su posición de acuerdo a las siguientes ecuaciones:

$$v_{id} = w * v_{id} + c_1 * r_1 * (pbest_{id} - x_{id}) + \quad (4.1)$$

$$c_2 * r_2 * (gbest_d - x_{id})$$

$$x_{id} = x_{id} + v_{id} \quad (4.2)$$

Donde v_{id} es el valor de velocidad de la partícula i en la dimensión d, c_1 es el valor de aprendizaje cognitivo, c_2 es el factor de aprendizaje global, r_1 y r_2 son valores aleatorios uniformemente distribuidos en el rango [0,1], x_{id} es la posición de la partícula i en la dimensión d, $pbest_{id}$ es el valor en la dimensión d de la partícula con el mejor valor objetivo encontrado por la partícula i, y $gbest_d$ es el valor en la dimensión d del individuo del swarm que encontró el mejor valor objetivo. El valor w es importante para la convergencia del algoritmo. En algunos estudios [7] [8], se ha demostrado que, para algunos problemas suaves, el decremento del valor w en el tiempo permite combinar la exploración de una búsqueda global con la exploración de una búsqueda local. En cada iteración del algoritmo, la dirección que tomara la partícula es modificada considerando los valores *pbest* y *gbest.* El pseudo código del PSO se muestra en Algoritmo 2.2 [11].

Algoritmo 2.2 PSO básico

```
1:  Fase de Inicialización del cumulo:
2:  Para cada partícula i, i ∈ [1, N] hacer
3:      Para cada dimensión d, d ∈ [1, D] hacer
4:          Aplicar a xid un valor aleatorio en el rango
            [Xmin, Xmax]
5:          Aplicar a pbestid el valor de xid
6:      Fin Para
7:      Calcular el fitness xi
```

8: Aplicar a valor fitness *pbesti* el valor fitness xi
9: Aplicar a *gbest* el valor de xi si el fitness xi es mejor que valor fitness *gbest*
10: **Fin Para**
11: **Fase de Busqueda:**
12: **Mientras** no se alcance la condición de parada **hacer**
13: **Para** cada partícula *i*, *i* ∈ [1, N] **hacer**
14: **Para** cada dimensión *d*, *d* ∈ [1, D] **hacer**
15: Calcular *vid*
16: Calcular *xid*
17: **Fin Para**
18: Calcular fitness *xi*
19: **Fin Para**
20: Actualizar *gbest* con xi si valor fitness xi es mejor qué valor fitness gbest
21: Actualizar $pbest_i$ y valor fitness *pbesti* si fitness xi es mejor que $pbest_i$
22: **Fin Mientras**
23: Retornar Resultados

B. Métricas de Rendimiento

Para evaluar el rendimiento de los dos algoritmos multiobjetivo se utilizan dos métricas que miden la convergencia al frente óptimo de Pareto.

1) Razon de Error (E):

Determina la proporción de los vectores objetivo en el conjunto de soluciones encontrado por el optimizador (Y_{known}), que se encuentran en Y_{true} [9]. Entre más cercana sea la razón a 1, habrá menos correspondencia entre el frente de Pareto obtenido y el real. De esta manera un valor E=0 es ideal. Matemáticamente se representa como:

$$E \triangleq \frac{\sum_{i=1}^{N} e_i}{N} \quad (5)$$

$$e_i = \begin{cases} 0 \; si \; un \; vector \; de \; Y_{known} \; esta \; en \; Y_{true} \\ 1 \qquad\qquad\qquad en \; el \; caso \; contrario \end{cases}$$

2) Distancia Generacional (G):

Esta métrica es un valor que representa que tan lejos esta Y_{known} del Y_{true} [9].Matemáticamente se define como:

$$G = \sqrt{\frac{\sum_{i=1}^{N} {d_i}^2}{N}} \quad (6)$$

Donde d_i es la distancia euclidiana (en *X*) entre cada vector objetivo que pertenece a Y_{known} y su miembro correspondiente más cercano en el frente Pareto optimo real Y_{true}. Entre más grande sea el valor de G, más alejado estará el Y_{known} y el Y_{true}. De esta manera, un valor G=0 es ideal.

IV. MARCO EXPERIMENTAL

Para lograr determinar el desempeño de los optimizadores antes mencionados, se utiliza un problema de prueba conocido en la literatura como ZDT1 [10] que tiene un frente óptimo de Pareto convexo. Este problema se describe a continuación:

$$f_1(x_1) = x_1 \quad (7)$$

$$g(x_2, \dots, x_m) = 1 + 9 * \sum_{i=1}^{m} x_i/(m-1)$$

$$h(f_1, g) = 1 - \sqrt{f_1/g}$$

$$f_2(x) = g(x_2, \dots, x_m) * h(f_1, g)$$

Donde m=30 es el número de variables de decisión y $x_i \in [0,1]$. El frente Pareto optimo real se da para *g=1*.

Los dos optimizadores aquí propuestos se compilan y ejecutan en MatLab [12] 10 veces, con el fin de obtener 10 diferentes conjuntos de soluciones óptimas para cada uno y así lograr evaluar la cercanía de los mismos al conjunto óptimo de Pareto real. Se escoge [12] como compilador debido a la facilidad en la programación. En las Tablas 1 y 2, se indican los parámetros utilizados para los dos algoritmos.

Parámetros utilizados en NSGA-II	
Tamaño de población N	100
m	30
Numero de iteraciones	200
Rango, variables decisión	[0 1]
Taza de cruce	0.8
Taza de mutación	0.033
Numero de mutantes	20

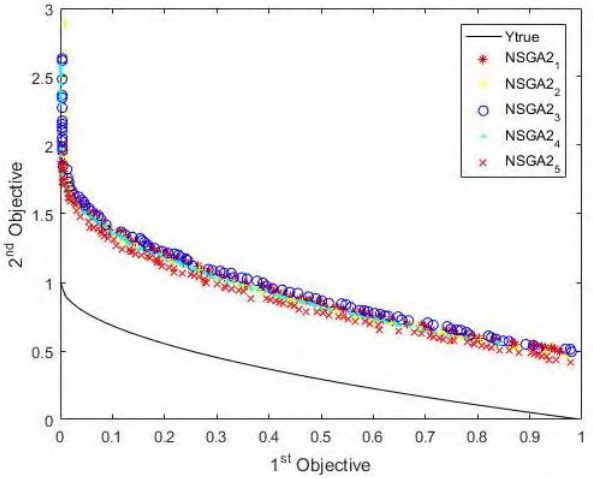
Tabla 1. Parámetros para ejecución del algoritmo NSGA-II

Parámetros utilizados en PSO	
Tamaño de población N	100
variables de decisión	30
Numero de iteraciones	200
Rango, variables decisión	[0 1]
w	0.5
Wdamp	0.99
taza de mutación	0.01
C1	1
C2	2

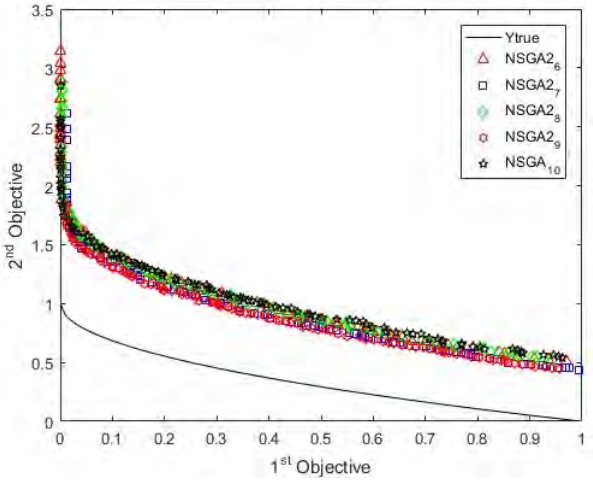
Tabla 2. Parámetros utilizados para ejecución del algoritmo PSO.

V. RESULTADOS

Las Gráficas 1 y 2, muestran los conjuntos de solución encontrados por el NSGA-II en las 10 ejecuciones.

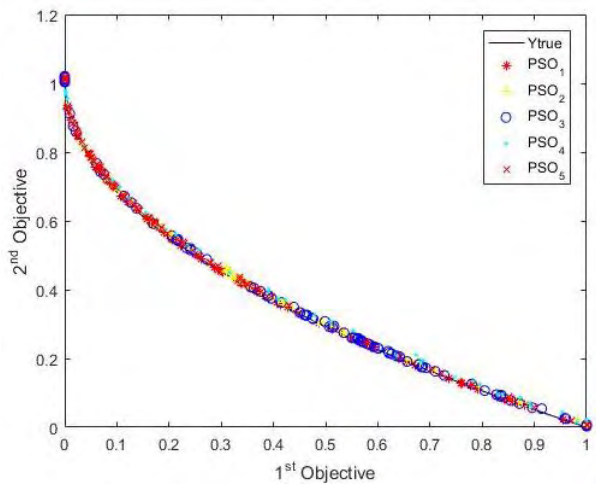


Grafica 1. Conjuntos Y_{known} del NSGA-II vs. Y_{true}.

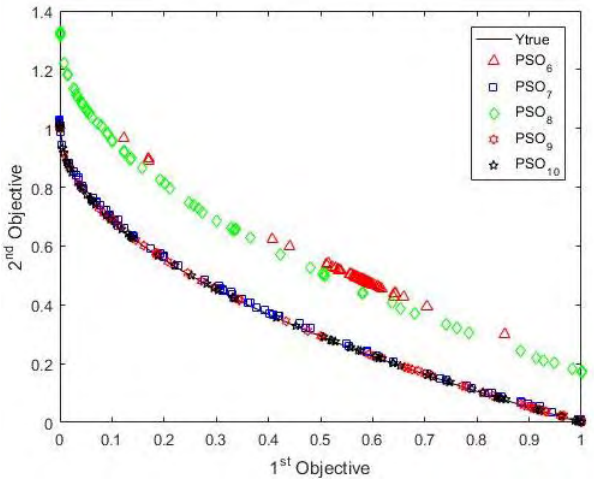


Grafica 2. Conjuntos Y_{known} del NSGA-II vs. Y_{true}.

Las Gráficas 3 y 4 muestran los conjuntos de solución encontrados por el PSO en sus 10 ejecuciones.



Grafica 3. Conjuntos Y_{known} del PSo vs. Y_{true}



Grafica 4. Conjuntos Y_{known} del PSO vs. Y_{true}

Una vez obtenidos los resultados, se procede a utilizar las métricas de rendimiento antes mencionadas en los 20 conjuntos de soluciones encontrados luego de optimizar el problema ZDT1. La Tabla 3, muestra los resultados obtenidos por la métrica E y la Tabla 4, los resultados de la métrica G.

Para determinar si un vector de Y_{known} pertenece al conjunto Pareto optimo real, se tuvo un margen de error de ± 0.005 unidades.

Set	NSGA-II	PSO
1	1	0.19
2	1	0.84
3	1	0.65
4	1	0.93
5	1	0.82
6	1	1
7	1	0.89
8	1	1

9	1	0.53
10	1	0.21
promedio	**1**	**0.7060**

Tabla 3. Resultados métrica E sobre ZDT1.

Set	NSGA-II	PSO
1	0,0470	0,0007
2	0,0608	0,0009
3	0,0630	0,0008
4	0,0614	0,0015
5	0,0401	0,0010
6	0,0810	0,0200
7	0,0529	0,0012
8	0,0721	0,0224
9	0,0397	0,0007
10	0,0665	0,0007
promedio	**0,0585**	**0,0050**

Tabla 4. Resultados métrica G sobre ZDT1.

VI. OBSERVACIONES

Los conjuntos de soluciones obtenidos por el algoritmo NSGA-II que se muestran en las Gráficas 1 y 2, siguen la forma del conjunto Pareto optimo real pero desplazados aproximadamente una unidad hacia arriba en el espacio de soluciones factibles. Las Gráficas 3 y 4, contienen los conjuntos de soluciones obtenidos por el algoritmo PSO e indican mayor cercanía al conjunto Pareto óptimo real a excepción de los conjuntos PSO_6 y PSO_8.

La métrica E expone que el PSO encuentra un mayor número de vectores objetivo contenidos en el conjunto Pareto optimo real que el NSGA-II que no logra ubicar ninguno de estos vectores en el Y_{true}.

La cercanía de los conjuntos de soluciones obtenidos por PSO es mucho mayor que la de los conjuntos encontrados por el algoritmo genético, como clara mente lo indican los resultados de la métrica G.

De lo anterior, se puede concluir que para el caso de un problema de optimización multiobjetivo convexo como el ZDT1, el algoritmo PSO tiene mayor efectividad que el NSGA-II a la hora de buscar en el espacio de soluciones factibles un conjunto de soluciones que tenga vectores objetivo muy cercanos o contenidos en el conjunto Pareto optimo real.

REFERENCES

[1] Van Veldhuizen, D. a. (1999). *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations*. *IRE Transactions on Education*.

[2] Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, *3*(4), 257–271.

[3] Crichigno, J., & Baran, B. (2004). Multiobjective multicast routing algorithm for traffic engineering. *Proceedings. 13th International Conference on Computer Communications and Networks (IEEE Cat. No.04EX969)*, *0*(C), 301–DEB, Kalyanmoy, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 2002, vol. 6, no 2, p. 182-197.

[4] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

[5] Srinivas, N., & Deb, K. (1994). Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, *2*(3), 221–248.

[6] Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, 69–73.

[7] Venter, G., & Sobieszczanski-Sobieski, J. (2003). Particle Swarm Optimization. *AIAA Journal*, *41*, 1583–1589.

[8] Zheng, Y., Ma, L., Zhang, L., & Qian, J. (2003). Empirical study of particle swarm optimizer with an increasing inertia weight. *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, *2*, 221–226.

[9] Duarte Flores, S., & Barán, B. (2001). Optimización multiobjetivo de redes empleando algoritmos evolutivos paralelos, (Proyecto de tesis).

[10] Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, *8*(2), 173–195.

[11] Cagnina, L. C. (2010). Optimización mono y multiobjetivo a través de una heurística de inteligencia colectiva, (tesis de doctorado), 54–72.

[12] Inc., T. M. (2016). MATLAB (R2016a). *The MathWorks Inc.*

ANEXO 2. PONENCIA: International Conference on Information Systems and Computer Science (INSISCOS)

Figura 36. Certificado de participación del evento INCISCOS2016 en Quito, Ecuador

ANEXO 3. PAGINA WEB: Interactive Comparator of Multiobjective Optimization algorithms

Uno de los aportes de este trabajo de grado, es la creación de un sitio web que contiene la información necesaria para entender el comparador interactivo aquí desarrollado. En esta se encuentra una breve introducción sobre el trabajo realizado, los documentos de la investigación, los archivos de Matlab de la Interfaz y un video tutorial para que un usuario no experto pueda utilizar con facilidad la interfaz realizada. Puede visitar la página web en el siguiente Link:

https://sites.google.com/site/degreethesisdiegopeluffo/interactive-comparator

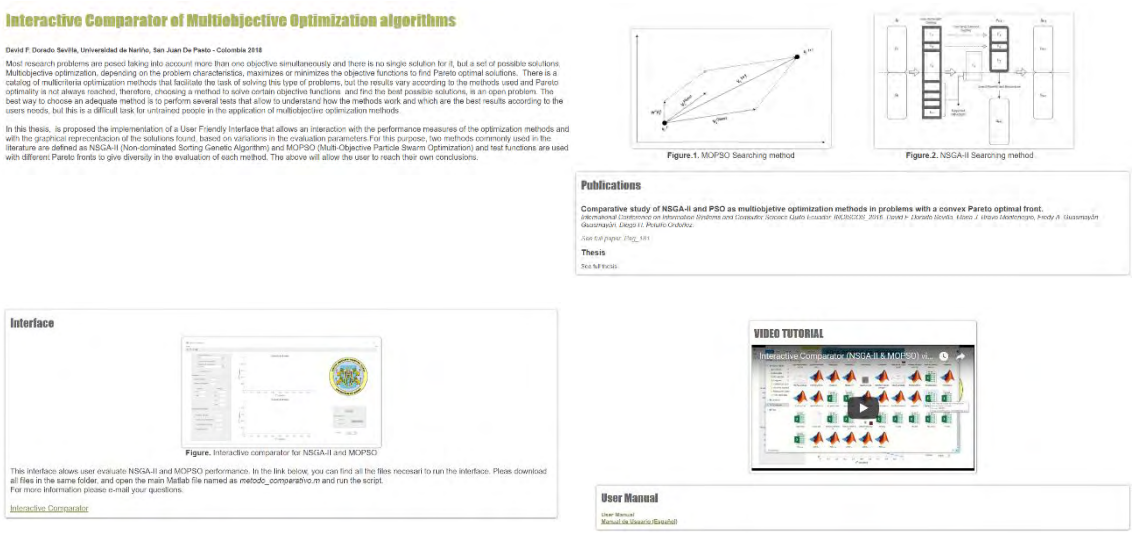


Figura 37. Página Web desarrollada para ayudar al usuario de la interfaz a entender su funcionamiento

ANEXO 4. MANUAL DE USUARIO: Comparador Interactivo de los Algoritmos de Optimización Multi-Criterio NSGAII y MOPSO

Se realizo un manual del comparador interactivo, para que los usuarios puedan entender con mayor facilidad su funcionamiento. Se describen las tareas que realiza cada una de las funciones que este tiene y los pasos que se deben seguir para optimizar los problemas de prueba disponibles. Se hizo una versión del manual en español y otra en inglés.

Comparador Interactivo de los Algoritmos de Optimización Multicriterio NSGA-II y MOPSO

Manual de Usuario

Departamento de Ingeniería Electrónica,
Universidad de Nariño

1. Introducción

Este documento es una guía rápida para entender y utilizar la Interfaz Gráfica (Figura.1.). El comparador interactivo permite a su usuario evaluar el desempeño de los algoritmos NSGA-II y MOPSO frente a5 funciones de prueba ZDT [], utilizando 4 medidas de desempeño y una representación gráfica del frente de Pareto encontrado por cada optimizador.

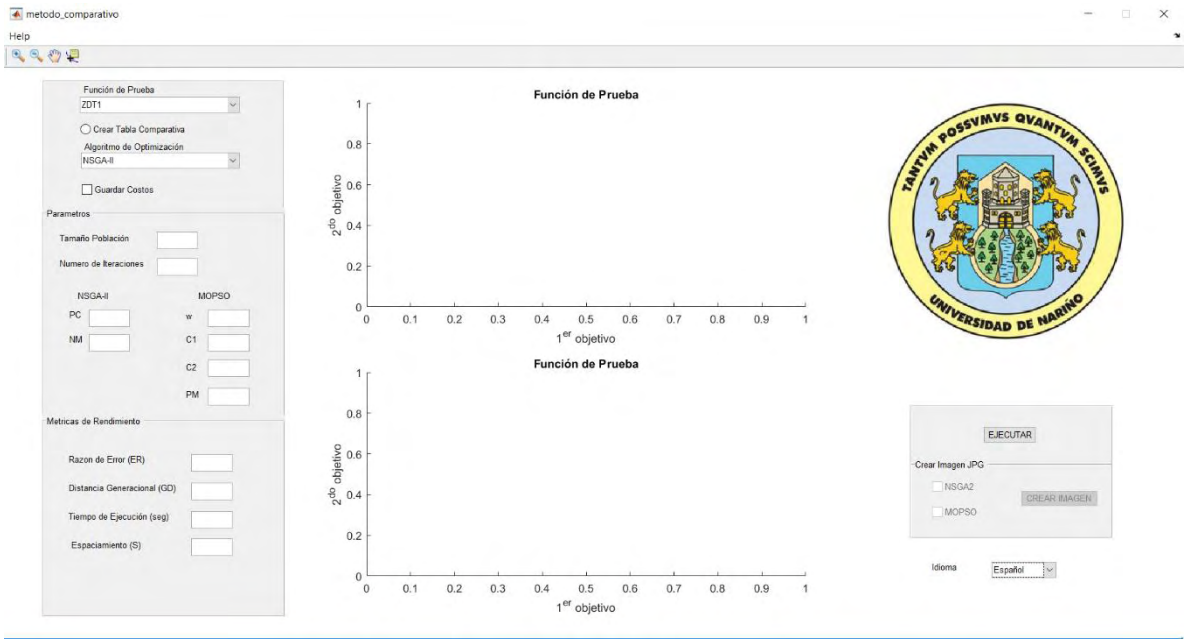


Figura.1. Comparador interactivo de los algoritmos de optimización Multicriterio NSAGA-II y MOPSO.

2. Ejecutar Comparador

- Expandir el menú “Función de Prueba” y elegir la función que desea evaluar.

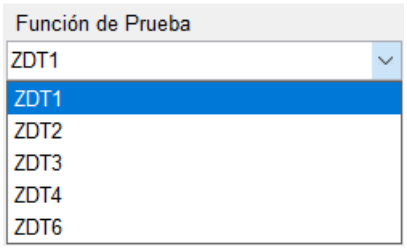


Figura.2. Función de Prueba.

- Expandir el menú “Algoritmo de Optimización” y elija el método para evaluar la función de prueba seleccionada en el paso anterior.

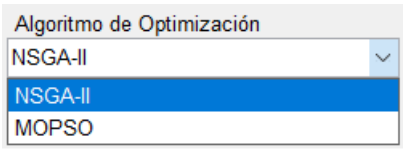


Figura.3. Algoritmo de Optimización.

- Inicialmente la Interfaz carga parámetros de ejecución predeterminados. Si lo desea puede modificar los valores de estos parámetros para sus propias evaluaciones.

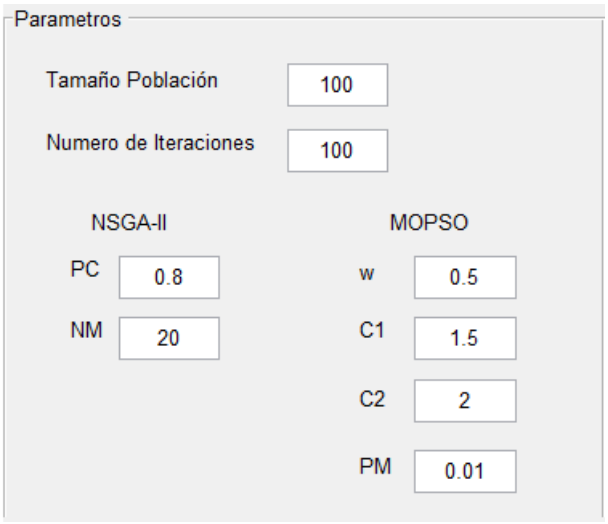


Figura.4. Parámetros.

- Finalmente presione clic en el botón “EJECUTAR”. El algoritmo seleccionado comenzara con la búsqueda del conjunto de soluciones Pareto óptimas.

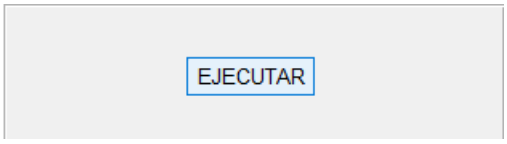


Figura.5. botón Ejecutar.

- Al finalizar la ejecución, se muestran los resultados de las medidas de desempeño en el panel “Métricas de Rendimiento” (Figura.6.) y la grafica del conjunto de soluciones Pareto optimas encontrado. El plano superior corresponde al NSGA-II y el plano inferior corresponde al MOPSO (figura.7.).

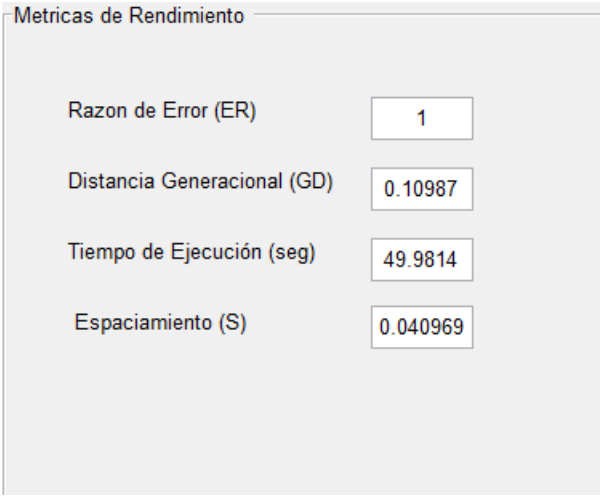


Figura.6. Métricas de Rendimiento.

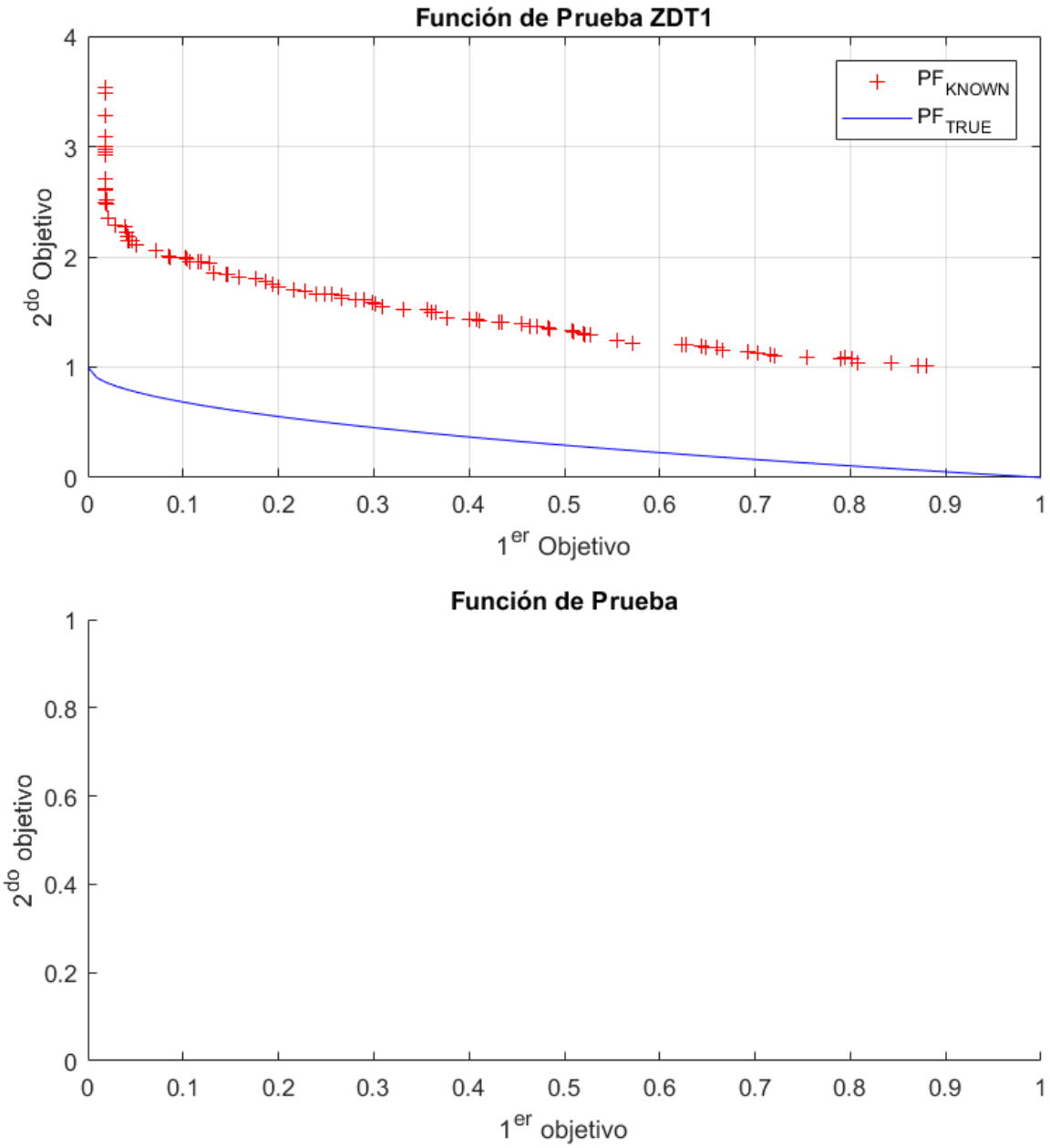


Figura.7. Plano Superior NSGA-II, plano inferior MOPSO.

3. Crear Tabla Comparativa

La Interfaz incluye una función que le permite crear una tabla comparativa de forma automática. La tabla contiene los resultados de las Métricas de Rendimiento encontradas en 10 ejecuciones diferentes de cada Algoritmo. Para activar esta función, debe seguir estos pasos:

- Elegir Función de Prueba.
- Presione clic en el botón circular “Crear Tabla Comparativa”.
- Ingresar los parámetros de evaluación deseados. En este punto, debe verse la interfaz como la Figura.8.

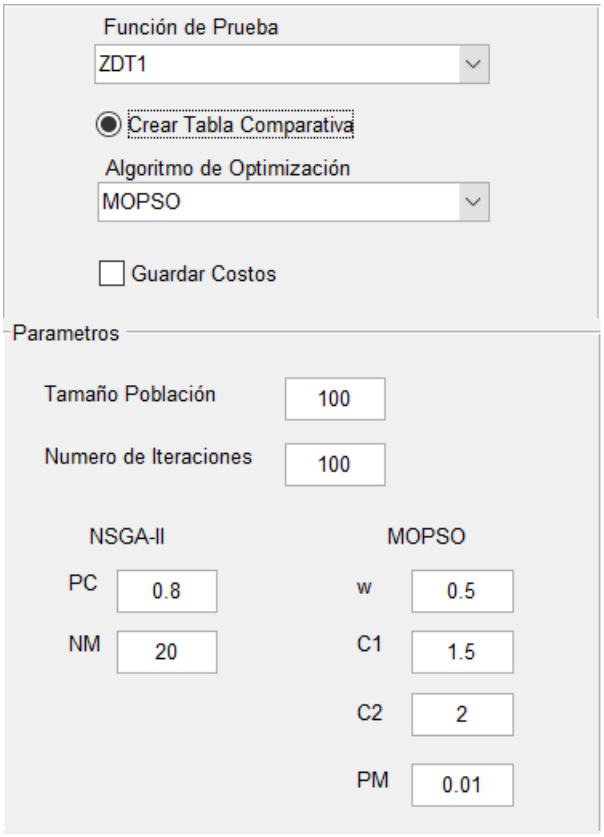


Figure.8. Crear Tabla Comparativa.

- Presione clic en el botón “EJECUTAR”. El algoritmo escogido corre 10 veces y guarda los resultados de las métricas de rendimiento.
- Ahora, debe seleccionar el otro algoritmo para continuar creando la tabla comparativa. Ingrese los parámetros deseados y presione el botón “EJECUTAR”. Si presiona “EJECUAR” sin antes haber cambiado de algoritmo de optimización, recibirá un mensaje de advertencia (Figure9.). Cuando el segundo algoritmo termine sus 10 ejecuciones, se crean dos archivos de Excel y se abre una ventana con mensaje (Figura.10.).
- El archivo Tabla_comp.xlsx contiene la tabla comparativa (Figura.11.) y el archivo Tabla_parametros.xlsx contiene los parámetros de evaluación utilizados por los dos algoritmos (Figura.12.).

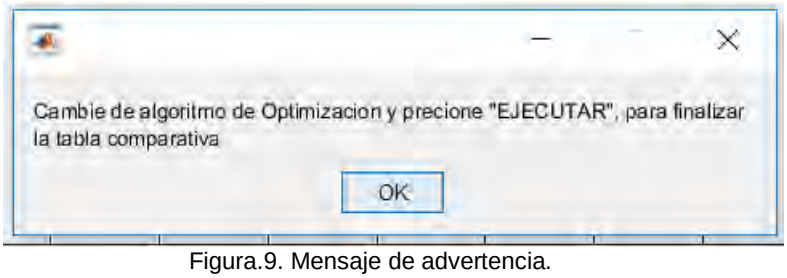


Figura.9. Mensaje de advertencia.

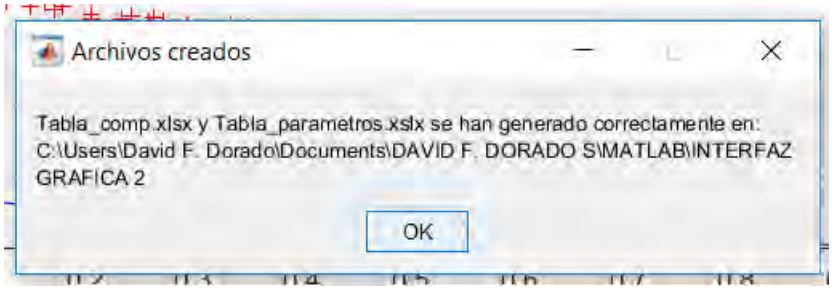


Figura.10. Dirección de los archivos creados.

	A	B	C	D	E	F	G	H	I
					Tablacomp				
	Ejecucion	E_NSGA2	E_MOPSO	DG_NSGA2	DG_MOPSO	Tiempo_NSGA2	Tiempo_MOPSO	S_NSGA2	S_MOPSO
	Text	Number	Number	Number	Number	Number	Number	Number	Number
1	Ejecucion	E_NSGA2	E_MOPSO	DG_NSGA2	DG_MOPSO	Tiempo_NSGA2	Tiempo_MOPSO	S_NSGA2	S_MOPSO
2	No.1	1	0.9700	0.2499	0.0289	6.8309	2.0917	0.1052	0.1424
3	No.2	1	1	0.2475	0.0588	7.1002	2.1449	0.1455	0.1926
4	No.3	1	1	0.2556	0.0628	6.9319	2.0478	0.1115	0.2319
5	No.4	1	1	0.2653	0.0646	7.0570	2.0419	0.0981	0.2341
6	No.5	1	0.9800	0.2726	0.0428	7.0491	2.1342	0.0819	0.2139
7	No.6	1	1	0.2793	0.0391	6.8719	2.0590	0.0890	0.1744
8	No.7	1	0.9900	0.2608	0.0474	6.8014	2.0580	0.0870	0.2378
9	No.8	1	1	0.2425	0.0747	6.9447	2.1376	0.0762	0.2346
10	No.9	1	1	0.2703	0.0436	6.9475	2.0552	0.0984	0.1824
11	No.10	1	0.9600	0.2379	0.0279	7.0410	2.0227	0.0990	0.1201
12	PROMEDIO	1	0.9900	0.2582	0.0491	6.9576	2.0793	0.0992	0.1964

Figura.11. Tabla comparativa.

	A	B	C
	Tablaparametros		
	Paramet...	NSGA_II	MOPSO
	Text	Number	Number
1	Parametr...	NSGA_II	MOPSO
2	N_pop	100	100
3	max_It	10	10
4	pc	0.8000	
5	nm	20	
6	w		0.5000
7	c1		1.5000
8	c2		2
9	pm		0.0100

Figura.12. Parámetros.

NOTA: Los archivos de Excel se sobrescriben cada vez que se crea una tabla comparativa, por lo tanto, si desea guardar los datos de la tabla creada copie el archivo y cambie el nombre antes de crear otra tabla.

4. **Guardar Costos**

Si necesita el conjunto de soluciones encontrado por el algoritmo seleccionado, presione clic sobre el cuadro “Guardar Costos” (figura.13.) para crear un archivo Excel que contiene los costos de la población. El cuadro debe seleccionarse antes de presionar el botón “EJECUTAR”.

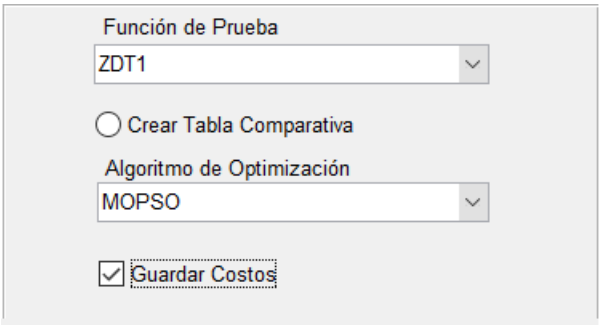


Figura.13. Guardar Costos.

5. **Crear Imagen**

Para crear un archivo JPG de un plano, debe seleccionar el cuadro del algoritmo que desea guardar como imagen y presionar el botón “CREAR IMAGEN” (Figura.14.).

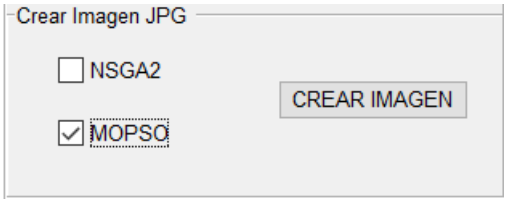


Figure.14. Crear Imagen.

Interactive Comparator of Multi-objective Optimization algorithms NSGA-II and MOPSO

User Manual

Electronic Engineering Department, Nariño University

6. Introduction

This document is a quick guide to understand and use the graphic interface (Figure.1.). The interactive comparator allows its user to evaluate the performance of the NSGA-II and MOPSO algorithms against five test functions. for this purpose, the interface has four performance metrics and a graphic representation of the found Pareto front.

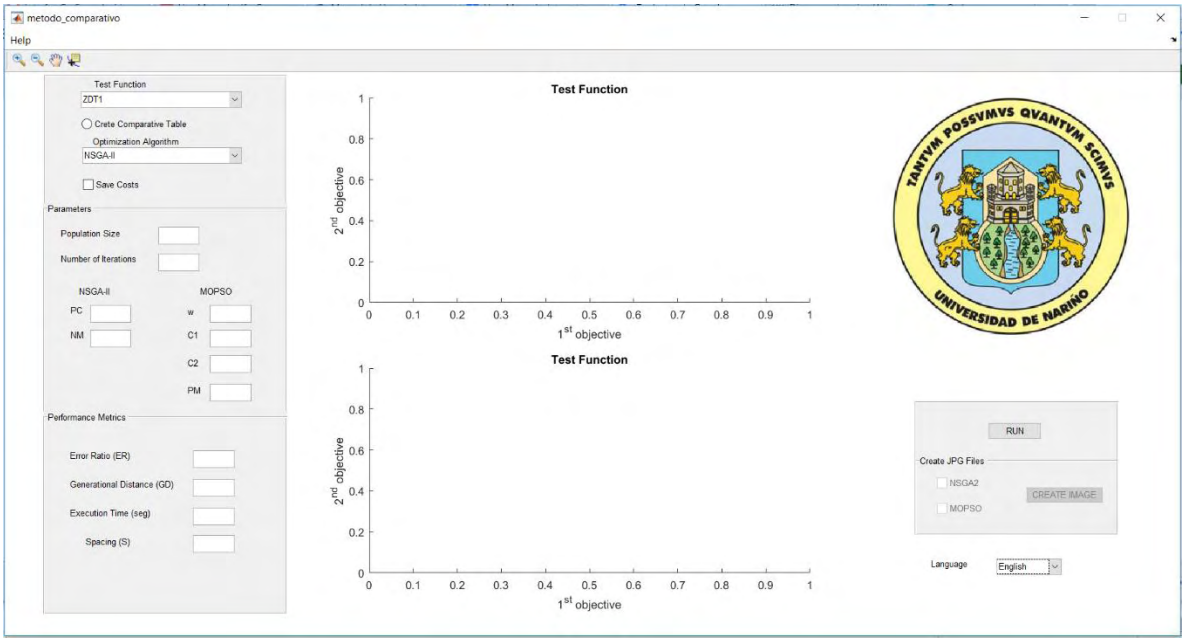


Figure.1. Interactive comparator of multi-objective optimization algorithms NSGA-II and MOPSO.

7. Run Comparator

- Expand the "test functions" popup menu and select the function you want.

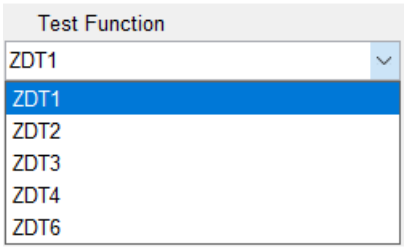


Figure.2. Test Function.

- Expand the “Optimization Algorithm” popup menu and select one method.

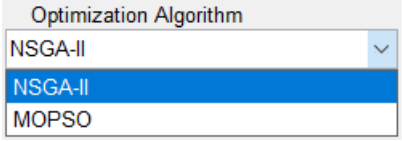


Figure.3. Optimization Algorithm.

- initially predetermined execution parameters are loaded. if you wish you can modify these values (Figure.4.).

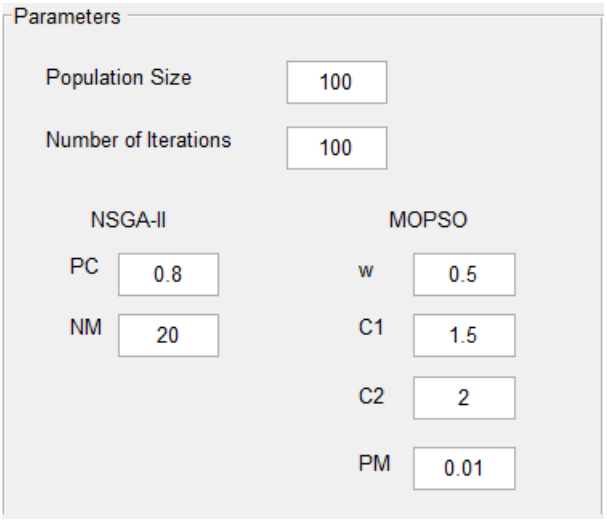


Figure.4. Parameters.

- Finally click the "RUN" button. The chosen algorithm begins searching the set of Pareto optimal solutions.

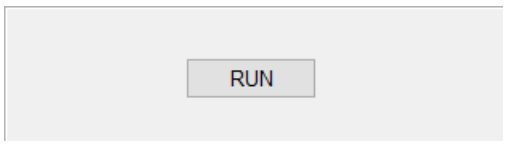


Figure.5. RUN button.

- At the end of the execution, the values of the performance metrics are printed (Figure.6.) and the found set of solutions and the Pareto optimal front are plotted. the upper axis is for NSGA-II and the lower axis is for MOPSO (Figure.7.)

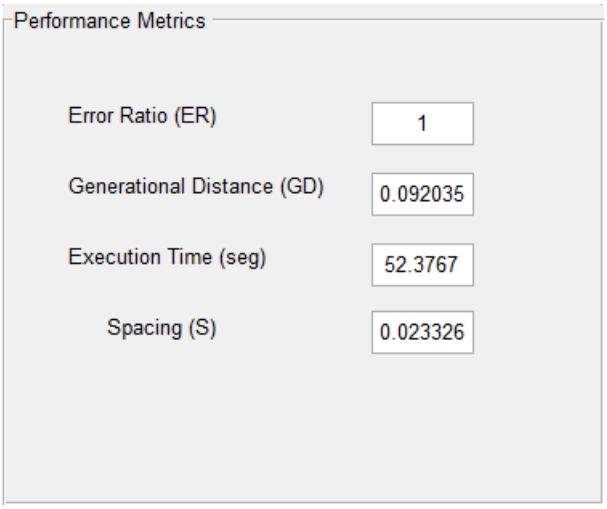


Figure.6. Performance Metrics.

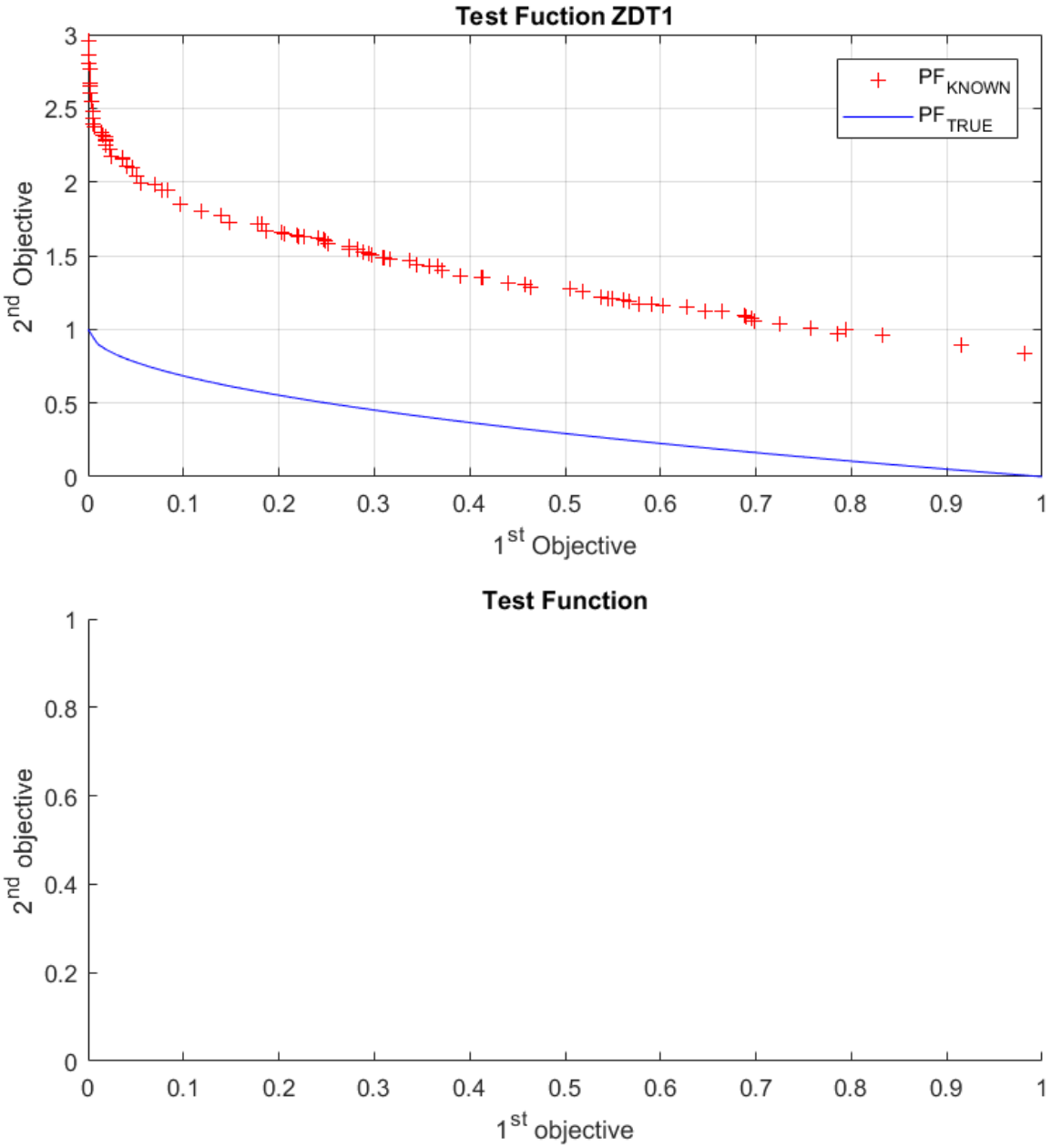


Figure.7. Upper axis NSGA-II, lower axis MOPSO.

8. **Create Comparative table**

The interface includes a function that allows the user to create a comparative table automatically. The table contains the results of the performance metrics found in ten different runs of each optimization algorithm. To activate this function, you must perform the following steps:

- Select test function.
- Click “Create Comparative Table” radio button.
- Enter the evaluation parameters to create the table. So far, the interface should look like figure.8.

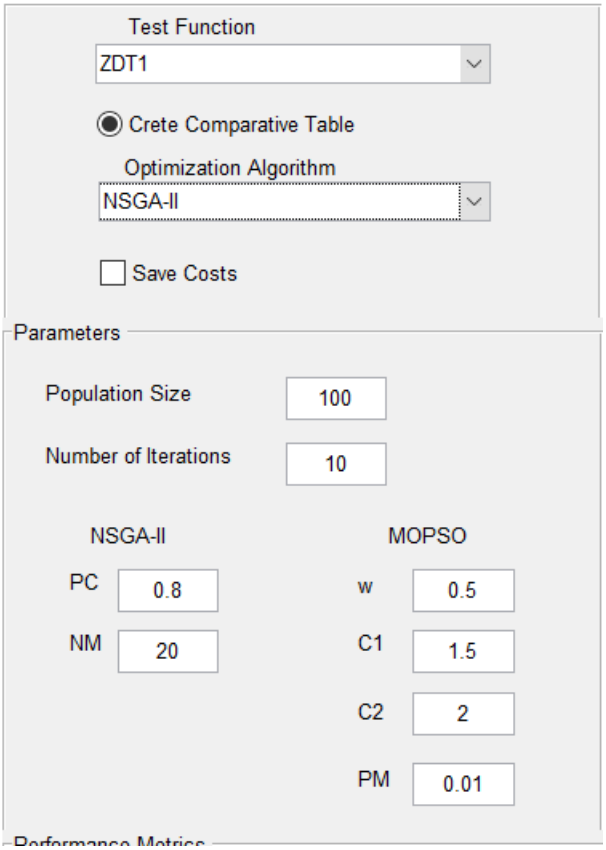


Figure.8. Create comparative table.

- Click “RUN” button. The chosen algorithm runs ten different times and keep all performance metrics results.
- You must select the other algorithm to continue creating comparative table. Enter the parameters you want to and click “RUN” button. If you click “RUN” button without changing the optimization algorithm, you will get an warning message (Figure.9.). When the second algorithm finish running ten times, two excel files are created and a message box is shown (figure.10.).
- Tabla_comp.xlsx contain the comparative table (Figure.11.) and Tabla_parametros.xlsx contains the evaluation parameter of the optimization algorithms (Figure.12.).

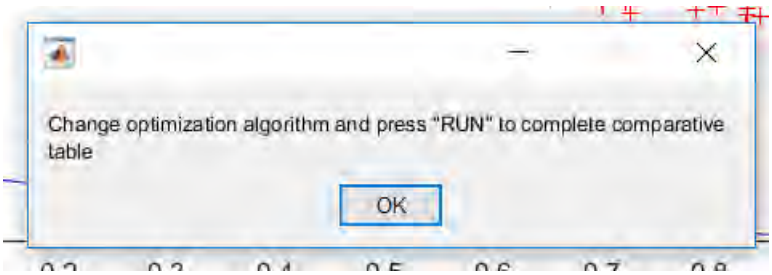


Figure.9. Warning message.

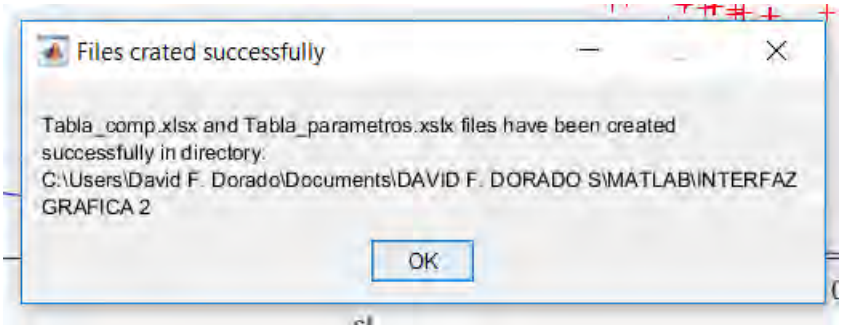


Figure.10. Created files names and directory.

	A	B	C	D	E	F	G	H	I
					Tablacomp				
	Ejecucion	E_NSGA2	E_MOPSO	DG_NSGA2	DG_MOPSO	Tiempo_NSGA2	Tiempo_MOPSO	S_NSGA2	S_MOPSO
	Text	Number	Number	Number	Number	Number	Number	Number	Number
1	Ejecucion	E_NSGA2	E_MOPSO	DG_NSGA2	DG_MOPSO	Tiempo_NSGA2	Tiempo_MOPSO	S_NSGA2	S_MOPSO
2	No.1	1	0.9700	0.2499	0.0289	6.8309	2.0917	0.1052	0.1424
3	No.2	1	1	0.2475	0.0588	7.1002	2.1449	0.1455	0.1926
4	No.3	1	1	0.2556	0.0628	6.9319	2.0478	0.1115	0.2319
5	No.4	1	1	0.2653	0.0646	7.0570	2.0419	0.0981	0.2341
6	No.5	1	0.9800	0.2726	0.0428	7.0491	2.1342	0.0819	0.2139
7	No.6	1	1	0.2793	0.0391	6.8719	2.0590	0.0890	0.1744
8	No.7	1	0.9900	0.2608	0.0474	6.8014	2.0580	0.0870	0.2378
9	No.8	1	1	0.2425	0.0747	6.9447	2.1376	0.0762	0.2346
10	No.9	1	1	0.2703	0.0436	6.9475	2.0552	0.0984	0.1824
11	No.10	1	0.9600	0.2379	0.0279	7.0410	2.0227	0.0990	0.1201
12	PROMEDIO	1	0.9900	0.2582	0.0491	6.9576	2.0793	0.0992	0.1964

Figure.11. Comparative table.

	A	B	C
	Tablaparametros		
	Paramet...	NSGA_II	MOPSO
	Text	Number	Number
1	Parametr...	NSGA_II	MOPSO
2	N_pop	100	100
3	max_It	10	10
4	pc	0.8000	
5	nm	20	
6	w		0.5000
7	c1		1.5000
8	c2		2
9	pm		0.0100

Figure.12. Parameters

NOTE: the excel files are overwritten every time you create a comparative table, so if you need these data you must copy these files and paste them with other name before you create other table.

9. **Save Costs**

If you need the set of solutions found by the selected algorithm you can check “Save Costs” (Figure.13.), to create a excel file that contains the population costs. Checkbox must be selected before pressing “RUN” button.

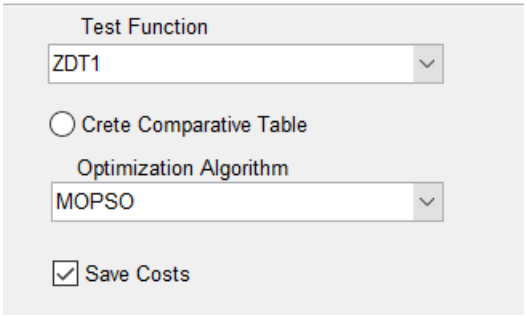


Figure.13. Save Costs.

10. **Create Image**

To create a JPG file of the axes you must check the algorithm plot that you want and press “CRATE IMAGE” button (Figure.14.)

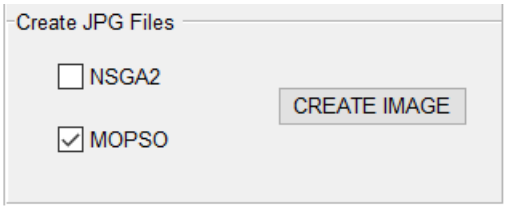


Figure.14. Create image.