Una Propuesta de Hibridación de Evolución Diferencial y Conjuntos Borrosos para Optimización Multi-Objetivo

Luis V. Santana Quintero
Carlos A. Coello Coello
CINVESTAV-IPN
Departamento de Computación
México D.F., MEXICO
lvspenny@hotmail.com
ccoello@cs.cinvestav.mx

Alfredo G. Hernández-Díaz
Universidad Pablo de Olavide
Depto. de Métodos Cuantitativos
Sevilla, ESPAÑA
agarher@upo.es

Julián Molina, Rafael Caballero
Universidad de Málaga
Depto. de Economía Aplicada (Matemáticas)
Málaga, ESPAÑA
julian.molina@uma.es
rafael.caballero@uma.es

Resumen— **Este trabajo presenta un algoritmo multi-objetivo basado en Evolución Diferencial y la teoría de conjuntos borrosos. Un rasgo distintivo de esta propuesta es el uso de la dominancia $pa\epsilon$, que lidia con varias de las limitaciones de la dominancia ϵ. En el híbrido propuesto, la evolución diferencial actúa como el motor de búsqueda, explotando así las buenas propiedades de convergencia de esta heurística. Posteriormente, los conjuntos borrosos se utilizan como un método de búsqueda local, que logra mejorar la distribución de las soluciones no dominadas obtenidas por la evolución diferencial. El resultado es un algoritmo que puede resolver problemas multi-objetivo con y sin restricciones, con un número reducido de evaluaciones (3,000 para los problemas sin restricciones y 10,000 para los problemas con restricciones). La propuesta es validada usando diversos problemas de prueba (con y sin restricciones), y comparando resultados con respecto a los producidos por el NSGA-II, usando métricas estándar tomadas de la literatura especializada.**

Palabras clave— **optimización multi-objetivo, evolución diferencial, conjuntos borrosos**

I. Introducción

La mayoría de los problemas del mundo real involucran la optimización simultánea de dos o más objetivos, los cuales normalmente se encuentran en conflicto entre sí. Este tipo de problemas se denominan "multi-objetivo", y su naturaleza suele dar pie no a una, sino a un conjunto de soluciones, a las cuales se denomina "no dominadas".

Debido a sus diversas ventajas, el uso de algoritmos evolutivos (y otro tipo de metaheurísticas) se ha vuelto muy popular en optimización multi-objetivo [1]. De entre los diversos algoritmos evolutivos que se han utilizado para optimización multi-objetivo, uno de los más recientes es la evolución diferencial [2]. Sin embargo, hasta ahora, se ha puesto poco énfasis en explotar las altas tasas de convergencia que caracterizan a la evolución diferencial cuando se ha utilizado en optimización mono-objetivo [3]. Esta alta tasa de convergencia, sin embargo, tiene un costo: de lograrse en problemas multi-objetivo, se presenta una muy alta pérdida de diversidad, lo cual se refleja con frentes de Pareto muy pobremente distribuidos. Sin embargo, si se acopla la evolución diferencial con un buen método de búsqueda local, es posible diseñar un algoritmo multi-objetivo que requiera un número muy bajo de evaluaciones de la función de aptitud. En nuestro caso, utilizamos conjuntos borrosos para realizar la búsqueda local. El resultado es un algoritmo capaz de resolver problemas sin restricciones en no más de 3,000 evaluaciones de la función de aptitud [4] (aún cuando dichos problemas lleguen a tener 30 variables) y problemas con restricciones en no más de 10,000 evaluaciones.

El resto del artículo está organizado de la forma siguiente. La Sección II proporciona una descripción general de la evolución diferencial. En la Sección III se describe brevemente la teoría en la que se basan los conjuntos borrosos. En la Sección IV se describe brevemente la dominancia Pareto adaptativa ϵ que se adoptó para el algoritmo propuesto en este artículo. El trabajo previo relacionado se revisa en la Sección V. Nuestra propuesta se detalla en la Sección VI. El diseño experimental que realizamos para validar nuestro esquema híbrido se presenta en la Sección VII y la Sección VIII describe los resultados obtenidos. Finalmente, en la Sección IX se presentan nuestras conclusiones y algunas de las posibles rutas de trabajo futuro.

II. Evolución Diferencial

La Evolución Diferencial (ED) [2] es una heurística relativamente reciente diseñada para resolver problemas de optimización en espacios continuos, en donde las variables se representan mediante números reales. La población inicial se genera de forma aleatoria y se seleccionan tres individuos como padres. Uno de ellos es el *padre principal* y éste se perturba con el vector de los otros dos padres. Si el valor resultante es mejor (cuando tiene un menor valor al evaluar la función objetivo) que el padre elegido, entonces lo reemplaza. De otra forma, se retiene al padre principal. Para cada vector $\overrightarrow{x_{i,G}}; i = 0, 1, 2, \ldots, N-1.$,

donde: N es el tamaño de la población; un vector de prueba $\overrightarrow{v}$ se genera de acuerdo a:

$$\overrightarrow{v} = \overrightarrow{x_{r1,G}} + F \cdot (\overrightarrow{x_{r2,G}} - \overrightarrow{x_{r3,G}})$$

con $r_1, r_2, r_3 \in [0, N-1]$, enteros y diferentes, y $F > 0$. Los enteros r_1, r_2 y r_3 son elegidos aleatoriamente en el intervalo [0, N-1] y son diferentes entre sí. F es un factor real y constante que controla la amplificación en la variación diferencial $(\overrightarrow{x_{r2,G}} - \overrightarrow{x_{r3,G}})$. G se refiere a la generación actual. Para decidir si el nuevo vector $\overrightarrow{v}$ se vuelve miembro de la población en la generación $G+1$, se compara con el vector $\overrightarrow{x_{i,G}}$, y si tiene un menor valor al evaluarla en la función objetivo, entonces $\overrightarrow{v}$ se hace $\overrightarrow{x_{i,G+1}}$; de otra forma, el valor de $\overrightarrow{x_{i,G}}$ es retenido.

III. Teoría de Conjuntos Borrosos

La Teoría de Conjuntos Borrosos es una nueva estrategia matemática para lidiar con problemas en los que se tiene conocimiento impreciso. El problema del conocimiento impreciso ha sido tratado durante mucho tiempo por filósofos, lógicos y matemáticos. Recientemente, ha sido un tema importante para las ciencias computacionales, particularmente en el área de la inteligencia artificial. Aunque existen muchos métodos para tratar este problema, el más utilizado ha sido la teoría de los conjuntos difusos propuesta por Lofti Zadeh [5]. La teoría de conjuntos borrosos fue propuesta por Pawlak [6], y ha sido utilizada en muchas aplicaciones, volviéndose un pilar de fundamental importancia para la inteligencia artificial y las ciencias cognitivas, especialmente en las áreas de aprendizaje de máquina, análisis de decisiones, descubrimiento de conocimiento en bases de datos, sistemas expertos y reconocimiento de patrones. La idea básica de la teoría de conjuntos borrosos así como algunas aplicaciones interesantes de la misma se han documentado en libros [7], revistas especializadas [8], y en internet (ver `www.roughsets.org`).

Supongamos que se tiene un conjunto de objetos llamado universo U y una relación de equivalencia $R \subseteq U \times U$, que representa la información disponible de los elementos en U (en este caso, R es una relación de equivalencia simple basada en una malla sobre el conjunto factible; esto es, una simple división del espacio factible en hiper-cuadrados). Si X es un subconjunto de U, entonces se quiere caracterizar este conjunto X con respecto a R. La forma en que la teoría de conjuntos borrosos trabaja ante esta imprecisión es analizando la frontera de la región del conjunto X. Si la frontera de este conjunto está vacía significa que ese conjunto es llamado *crisp*; de otra forma, el conjunto es llamado *borroso* (o inexacto). En la Figura 1 se ilustra la noción de aproximación que se usa en los conjuntos borrosos.

De acuerdo a lo anterior, cada elemento en U se clasifica con respecto a X. Se llama "aproximación superior" a los elementos de R (hiper-cuadrados o átomos) que contienen algún punto de X (y posiblemente alguno del conjunto no-X) y se llama "aproximación inferior" a los elementos de R que solo contienen puntos de X. La región *frontera* es la restante de estas dos. Cuanto más grande es la región frontera, entonces peor es el conocimiento que se tiene del conjunto X. Análogamente, cuanto más pequeña sea esta región, más precisa es la información que tenemos del conjunto X pero mayor número de elementos de R se tienen que tomar y más complejo se torna este problema. Lo que se desea hacer con los conjuntos borrosos es aproximarse al conjunto de óptimos de Pareto de un problema multi-objetivo. Para este propósito, se pretende diseñar una malla y decidir cuántos y cuáles elementos de R se tendrán que tomar del conjunto de óptimos de Pareto (llamados átomos) para delimitar la región frontera. Entonces el problema es:

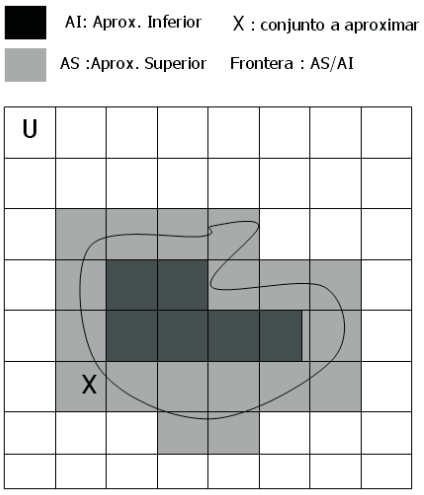

Fig. 1. Aproximación de los conjuntos borrosos

- Si la malla es más precisa, el costo computacional se vuelve muy grande como para manejarlo eficientemente.
- Si la malla es menos precisa, obtenemos menos conocimiento del conjunto de óptimos de Pareto.

Una vez que tengamos la malla construida, es relativamente fácil generar los átomos eficientes dentro de ella, dado que la construcción de estos átomos se hace en el espacio de las variables de decisión.

IV. Dominancia Pareto adaptativa-ϵ

Uno de los conceptos que ha despertado mayor interés en la comunidad de optimización multi-objetivo en los últimos años es, sin duda, el uso de formas relajadas de dominancia de Pareto que permitan el control de la convergencia de los algoritmos evolutivos multi-objetivo. De entre estas formas relajadas de dominancia, la dominancia-ϵ [9] es, sin duda, la más popular. La dominancia-ϵ ha sido utilizada principalmente como una estrategia para archivar las soluciones no dominadas, en la que se regula la aproximación del frente de Pareto dividiéndolo en hiper-rectángulos. Esto permite acelerar la conver-

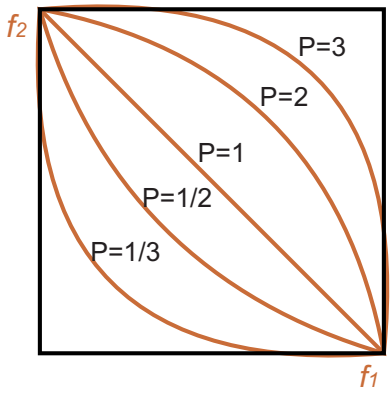

Fig. 2. Gráficas de las curvas P

gencia o mejorar la calidad de la aproximación, dependiendo del valor de ϵ que se adopte. Sin embargo, la dominancia- ϵ tiene algunas limitaciones, entre las que destacan:

- Se pierden un número importante de soluciones no dominadas si el tomador de decisiones no conoce de antemano las características geométricas del verdadero frente de Pareto (como suele ser el caso).
- Los extremos del frente de Pareto normalmente se pierden.
- Regular de manera precisa la cantidad de soluciones no dominadas que se desea retener en la malla ϵ no es fácil de lograr en la práctica.

A fin de evitar estos problemas o limitaciones, el concepto de dominancia-*pa*ϵ fue propuesto en [10]. De forma breve, la principal diferencia con respecto a la dominancia-ϵ radica en el tipo de malla que se forma. En la dominancia-*pa*ϵ, la malla se adapta al tamaño y a la forma geométrica del frente de Pareto de cada problema. De tal forma, este esquema mantiene las mismas buenas propiedades de la dominancia-ϵ mejorando sus debilidades. Esto se hace seleccionando un valor de ϵ diferente en cada objetivo, y realizando un muestreo preliminar que permita estimar la forma geométrica del frente de Pareto. Para este propósito, a cada frente de Pareto se le asocia un tipo de curva de la familia siguiente, tal y como se ilustra en la Figura 2:

$$\{x^p + y^p = 1 : 0 \leq x, y \leq 1, 0 < p < \infty\}.$$

para problemas de optimización bi-objetivo, o

$$\{x^p + y^p + z^p = 1 : 0 \leq x, y, z \leq 1, 0 < p < \infty\}$$

para problemas tri-objetivo. Estas familias tienen la propiedad siguiente: para $p > 1$, la curva es cóncava y cuanto más grande el valor de p, más horizontal (y vertical) es el frente; y para $p < 1$, la curva es convexa y entre más pequeño es el valor de p más horizontal (y vertical) se vuelve el frente. Finalmente, para $p = 1$ obtenemos la linea $x + y = 1$. En este caso, la dominancia-*pa*ϵ se reduce exactamente a la dominancia-ϵ.

Por lo tanto, se necesita asociar un valor de p al frente de Pareto, y se requiere una aproximación a este frente, la cual denotaremos con F. Esta aproximación determinará el valor de p a la curva que mejor se le parezca. Evidentemente, el número de puntos no dominados incluidos en F es crítico para un buen desempeño de este mecanismo, porque si el valor de p no se asigna apropiadamente, la malla no trabajará adecuadamente y esto puede resultar contraproducente para el desempeño del algoritmo de optimización multi-objetivo.

Para calcular p, se necesita determinar el área (hipervolumen) debajo del frente que forman los puntos en F. Una vez que conocemos esta área, se estima el valor de $p \in (0, +\infty)$ por medio de una interpolación con respecto al área que cubren los puntos en el frente. Una vez que conocemos el valor p y tenemos el valor T (dado por el usuario), que es el número de soluciones que se quieren en el frente de Pareto, se calculan los tamaños de las cajas para cada objetivo $i \in \{1, 2, ..., m\}$, esto es, el vector $\epsilon^i = \left(\epsilon_1^i, \epsilon_2^i, ..., \epsilon_T^i\right)$, utilizando series geométricas.

En [10], se proporcionan más detalles sobre este esquema y las ventajas que presenta adoptarlo en vez de la dominancia-ϵ, razón por la cual lo usamos en el trabajo desarrollado en este artículo.

V. Trabajo Previo

Existen varias propuestas recientes en las que se extiende a la evolución diferencial de manera que pueda resolver problemas multi-objetivo. Las propuestas más representativas se describen brevemente a continuación:

- **PDE** [11]: Maneja sólo una población principal. La reproducción se realiza sólo entre las soluciones no dominadas, y un hijo se coloca en la población sólo si domina a su padre. Este esquema utiliza una medida de distancia para mantener la diversidad en la población.
- **PDEA** [12]: Combina la ED con los elementos básicos del NSGA-II, tales como el ordenamiento de las soluciones no dominadas y su selección en rangos.
- **GDE** [13]: GDE extiende el operador de selección del algoritmo de ED para manipular restricciones multi-objetivo. Los autores reportan que el desempeño del GDE es similar al del NSGA-II, pero argumentan que es más eficiente, pues reportan tiempos de ejecución menores a los del NSGA-II.
- **NSDE** [14]: Es una simple modificación al NSGA-II donde el cruce y la mutación para representación real se reemplazan por el operador mixto adoptado en la evolución diferencial.
- **DEMO** [15]: Combina las ventajas de la ED con la jerarquización de Pareto y la distancia de agrupamiento (*crowding*). En DEMO, las soluciones nuevas son insertadas inmediatamente en la población permitiendo que sean candidatas a ser seleccionadas como padres en la siguiente generación.

Ninguno de estos algoritmos multi-objetivo basados en ED reportados en la literatura especializada han reportado tener éxito con menos de 20,000 evaluaciones de la función de aptitud. De hecho, algunos

de estos esquemas reportan más de 50,000 evaluaciones en problemas de alta dimensionalidad tales como las funciones ZDT con 30 variables. En contraste, mostraremos que nuestro algoritmo puede generar el verdadero frente de Pareto o una aproximación muy buena de él con tan sólo 3,000 (cuando el problema no tiene restricciones) y 10,000 evaluaciones (cuando el problema tiene restricciones).

VI. Algoritmo Propuesto

Nuestra propuesta, llamada DEMORS (*Differential Evolution for Multiobjective Optimization with Rough Sets*), se divide en dos fases. Durante la fase I, se aplica la evolución diferencial como el motor de búsqueda, que intenta converger tan cerca como sea posible del verdadero frente de Pareto. Durante la fase II, se usan los conjuntos borrosos como un buscador local que intenta mejorar la dispersión de las soluciones no dominadas generadas en la fase previa. Cada una de estas dos fases se describe con mayor detalle a continuación:

A. Fase I

El pseudocódigo del algoritmo de ED que adoptamos para la fase I se muestra en el Algoritmo 1 (para mayores detalles sobre el algoritmo, ver [16]). Nuestro algoritmo utiliza tres poblaciones: la población principal (que se utiliza para seleccionar los padres), una población secundaria (externa) que se usa para retener a las soluciones no dominadas encontradas a lo largo del proceso y una tercera población que retiene a las soluciones dominadas que son eliminadas por la población secundaria.

Algoritmo 1 Algoritmo de la Fase I

```
1:  Inicializa la población P
2:  Evaluar la aptitud para cada solución en P
3:  for i = 0 to G do
4:      repeat
5:          Selecciona (aleatoriamente) tres padres
6:          Realiza el cruce con ED
7:          Realiza la mutación
8:          Evalúa la aptitud del hijo
9:          if Si el hijo es mejor que el padre then
10:             Lo reemplaza en la población
11:     until población esté completa
12:     Identifica soluciones no dominadas
13:     Agrega no dominados a la población secundaria
14:     Agrega los dominados a la población terciaria
15: end for
```

Primero, se generan aleatoriamente 25 individuos, y éstos se usan a su vez para generar 25 hijos. La fase I tiene dos mecanismos diferentes de selección que se activan basado en el número total de generaciones y un parámetro llamado $sel_2 \in [0, 1]$, que regula la presión de selección. Por ejemplo, si $sel_2 = 0.6$ y el número total de generaciones es $G_{max} = 200$, significa que durante las primeras 120 generaciones (60 % de G_{max}), se adopta una selección aleatoria, y durante las últimas 80 generaciones se adopta una selección elitista. En ambas selecciones (aleatoria y elitista), se selecciona un padre como referencia. Este padre es utilizado para compararse con el hijo generado por los tres padres. Este mecanismo garantiza que todos los padres de la población sean usados como padres de referencia por una sola ocasión. Ambos mecanismos de selección se describen a continuación:

1. **Selección Aleatoria:** Se seleccionan 3 padres de la población primaria de forma aleatoria.

2. **Selección Elitista:** Se seleccionan 3 padres de la población secundaria de forma aleatoria y además éstos deben cumplir con un factor de cercanía entre ellos $f_{cercania}$. Este factor está dado por:

$$f_{cercania} = \frac{\sqrt{\sum_{i=0}^{k} (X_{i,max} - X_{i,min})^2}}{2^k}$$

donde: k = número de funciones objetivo, $X_{i,max}$ = valor máximo de la i-ésima función objetivo del fichero externo, $X_{i,min}$ = valor mínimo de la i-ésima función objetivo del fichero externo.

La recombinación en la fase I se realiza conforme al siguiente procedimiento: Para cada padre $\overrightarrow{p_i}$; $i = 0, 1, 2, \ldots, P-1$ (P = población), el hijo $\overrightarrow{h}$ es generado utilizando:

$$\begin{cases} h_j = p_{r1,j} + F \cdot (p_{r2,j} - p_{r3,j}), & \text{si } x \in [0,1) < p_c; \\ h_j = p_{ref,j}, & \text{d.o.f.} \end{cases} \quad (1)$$

donde: $j = 0, 1, 2, \ldots, n-1$ (n = número de variables). p_c = probabilidad de cruce , $p_{r1}, p_{r2}, p_{r3} \in [0, P-1]$, son enteros y diferentes entre sí, y $F > 0$. Los enteros r_1, r_2 y r_3 son los índices de los padres seleccionados aleatoriamente en el intervalo $[0, N-1]$ y ref es el índice del padre de referencia. F es un factor real y constante que mantiene el control de la diferencia $p_{r2,j} - p_{r3,j}$.

Aunque la Evolución Diferencial no maneja el operador de mutación en ninguno de sus esquemas, en este trabajo se incluyó este operador debido a que en problemas con restricciones, el cruce por sí solo no lograba llegar a ciertas regiones del espacio de búsqueda. Dado que existe evidencia clara de que la mutación ayuda a la exploración de los algoritmos evolutivos, se eligió una mutación uniforme para nuestra propuesta.

Una vez que se ha generado un hijo, se realiza la evaluación de éste con respecto a las funciones objetivo. Después se compara con el padre de referencia, y se elige a uno de ellos. Es importante mencionar que el esquema que se maneja para los problemas

con restricciones requiere que éstas se normalicen. A continuación se darán a conocer las reglas de comparación entre el padre y el hijo para las funciones sin y con restricciones.

** si el padre es no factible y el hijo es no factible*, se elige al individuo que esté más cerca de la zona factible.

** si el padre es factible y el hijo es no factible*, se elige al hijo si y sólo si el hijo está a una distancia de 0.1 de la zona factible y además se hace un $flip$ (0.5). De lo contrario, se elige al padre.

** si el padre es no factible y el hijo es factible*, se elige al padre si y sólo si el padre está a una distancia de 0.1 de la zona factible y además se hace un $flip$ (0.5). De lo contrario, se elige al hijo.

Si ambos son factibles, entonces se comparan en cuanto a la dominancia de Pareto.

- *si el padre domina al hijo*, se elige al padre
- *si el hijo domina al padre*, se elige al hijo
- *si ambos son no dominados entre ellos*, se realiza un $flip$ (0.5) para determinar quién trasciende a la siguiente generación.

Como se indicó anteriormente, nuestra propuesta utiliza un fichero externo (también conocido como población secundaria). Para incluir una solución en el fichero, ésta es comparada con respecto a todos los puntos contenidos en el fichero utilizando la malla de dominancia-$pa\epsilon$ [10]. Cualquier solución que sea removida de la población secundaria es incluida en la población terciaria. La población terciaria almacena las soluciones dominadas que son necesarias por la fase II.

B. Fase II

Una vez que concluye la primera fase, la segunda comienza a partir del conjunto de no dominados generado por la fase I (llamado ES). Esta fase también requiere un conjunto de soluciones dominadas (DS), que está en la población terciaria.

Se eligen $NumEff$ del conjunto ES, y $NumDom$ del conjunto DS. Estos puntos serán utilizados para aproximar la región entre el frente de Pareto y el resto de las soluciones factibles en el espacio de las variables. Lo que se pretende hacer es intensificar la búsqueda en el área en la que residen las soluciones no dominadas, y evitar buscar más puntos en el área en la que se encuentran los puntos dominados. Para este propósito, almacenamos todos estos puntos en un conjunto llamado $Items$ y construimos los conjuntos borrosos de la siguiente manera:

1. **Inicialización:** Para cada variable i, se calcula y se ordenan (de menor a mayor) los valores que toma en el conjunto $Items$. Después, para cada variable i, se obtienen los valores $Rango_i$, los cuales, al juntarse, dan pie a una malla en el espacio de las variables.
2. **Calcular Átomos:** Se calcula $NumEff$ rectángulos con centro en los puntos $NumEff$ previamente seleccionados. Para contruir este rectángulo se le asocia un punto no dominado y se calculan los límites superiores e inferiores para cada variable i, (x_i^e representa la i-esima variable de la x^e solución contenida en el conjunto $Items$:
   - límite-inferior i: punto medio entre x_i^e y el valor previo en el conjunto $Rango_i$.
   - límite-superior i: punto medio entre x_i^e y el valor siguiente en el conjunto $Rango_i$.

   En ambos casos, si no existen valores previos o siguientes en el conjunto $Rango_i$, se considera el límite inferior o superior de la variable i. Este mecanismo permite explorar las regiones cercanas al frente de Pareto.

Algoritmo 2 Algoritmo de la Fase II

```
1:  ES ← soluciones nodominadas de la Fase I
2:  DS ← soluciones dominadas de la Fase I
3:  eval ← 0
4:  repeat
5:      Items ← NumEff ∪ NumDom points
6:      Inicialización
7:      Calcular Átomos
8:      for i ← 0, hijos do
9:          eval ← eval + 1
10:         ES ← se genera hijo
11:         Agrega hijo al conjunto ES
12: until maxeval < eval
```

3. **Generar Soluciones:** Dentro de cada átomo se generan aleatoriamente nuevos puntos. Cada uno de estos puntos se envía al conjunto ES sólo si es factible, a fin de verificar si se incluye o no como nuevo punto no dominado. Si cualquier punto en ES es dominado por este nuevo punto, entonces se envía al conjunto DS.

VII. Diseño Experimental

Para validar nuestro algoritmo, nuestros resultados se comparan con respecto a aquellos generados por el NSGA-II [17] [1], que es un algoritmo representativo del estado del arte en el área.

En la primera parte de nuestro algoritmo se requieren de tres parámetros: probabilidad de cruce (p_c), elitismo (Sel_2) y tamaño de la población primaria (P). En la segunda fase del algoritmo se usan tres parámetros más: número de puntos generados dentro de cada átomo $Offspring$, número de átomos por generación ($NumEff$) y número de puntos dominados considerados para los átomos ($NumDom$). Finalmente, se requiere también definir el número deseado de soluciones no dominadas necesarias para construir por vez primera la malla-$pa\epsilon$.

Nuestro esquema híbrido es validado utilizando 6 problemas diferentes de prueba, de los cuales, cuatro tienen restricciones: (1) y (2) son dos problemas del conjunto **ZDT** [18] que son de alta dimensionalidad con hasta 30 variables, (3) **Kita**, propuesta en

[1] La versión adoptada del NSGA-II fue liberada el 10 de Mayo de 2005 y está disponible en `http://www.iitk.ac.in/kangal/codes.shtml`.

1996 [19], (4) **Welded Beam**, que es un problema de ingeniería relacionado a una viga que necesita ser soldada [20], (5) **Osyczka**, que es particularmente difícil por la forma del frente de Pareto [21] y (6) **Tanaka**, que es discontinua en el espacio de las funciones objetivo [22]. En los casos para problemas sin restricciones los parámetros utilizados fueron: p_c =0.3, Sel_2 =0.1, $P = 25$, $G_{max} = 80$, $Offspring = 1$, $NumEff = 2$, $NumDom = 10$ y $Eval = 3,000$; y para los problemas con restricciones solo cambian los parámetros $G_{max} = 250$ y $Eval = 10,000$. El NSGA-II se utilizó con los siguientes parámetros: p_c = 0.9, p_m = 1/num_var (num_var = número de variables), $\eta_c = 15$, $\eta_m = 20$, tamaño de población = 100 y número máximo de generaciones = 30 (se realizó un total de 3,000 evaluaciones de la función de aptitud); para los problemas con restricciones se utilizaron los mismos parámetros, excepto por el número máximo de generaciones que cambió a 100 (y realizar 10,000 evaluaciones de la función de aptitud).

Para comparar cuantitativamente nuestros resultados, se adoptaron las siguientes medidas de desempeño: *Distancia Generacional Invertida (IGD)* propuesta por Veldhuizen [23] en la que se mide la distancia del verdadero frente con respecto al generado por el algoritmo; *Cobertura (SC)* propuesta por Zitzler et al. [18], en la que se comparan dos frentes generado por diversos algoritmos y se verifica dominancia entre los puntos de ambos frentes; *Dispersion (Spread)*, propuesta por Deb [24], la cual mide la distribución de las soluciones y la distancia con los extremos del verdadero frente.

VIII. Análisis de Resultados

En la Tabla I se muestra una recopilación de los resultados para cada problema de prueba. En cada caso, se realizaron 30 ejecuciones independientes de cada algoritmo para cada problema de prueba. Los resultados reportados en la Tabla I son el promedio del desempeño de cada métrica y su desviación estándar sobre las 30 ejecuciones. Los mejores valores en cada caso se muestran en **negritas**.

Se puede ver claramente en la Tabla I que nuestra propuesta (DEMORS) produce los mejores valores en la mayoría de los casos. En los problemas sin restricciones, que son **ZDT1** y **ZDT4** obtenemos los mejores resultados en las tres diferentes métricas. Para los problemas con restricciones se le gana al NSGA-II en la mayoría, aunque hubieron unas pocas excepciones. Con respecto a la métrica IGD se obtienen los mejores resultados excepto para **Osyczka**, la cual contiene 5 regiones de Pareto diferentes y resulta muy complicado cubrirlas todas. Aunque el NSGA-II no logra cubrir todas esas áreas de forma uniforme, logra acercarse más al verdadero frente que DEMORS. Con respecto a la métrica Cobertura (SC), el NSGA-II obtiene mejores resultados en la **Kita** y **Welded Beam**, lo cual refleja que el NSGA-II es capaz de aproximarse al verdadero frente en algunas regiones de Pareto de este problema, aunque en otras no logra converger. Nuestro algoritmo no logró acercarse tanto como el NSGA-II, pero sí logró abarcar el frente de Pareto completo (con respecto a anchura). En cuanto a la métrica de Spread, el NSGA-II obtuvo mejores resultados en **Tanaka**, debido a que este problema tiene un frente discontinuo y la medición de la distribución afecta por igual a ambos algoritmos (nótense los valores altos para esta métrica en este problema).

En cuanto a los resultados de las gráficas mostrados en la Figura 3, éstos sirven para reforzar las ventajas que nosotros argumentamos que presenta nuestro algoritmo con respecto al NSGA-II. Estas gráficas corresponden a la ejecución en la media con respecto a la métrica IGD. En todos los casos, el frente de Pareto verdadero fue obtenido por enumeración y se muestra como una línea continua junto a las aproximaciones obtenidas por cada algoritmo, las cuales se muestran con círculos. En las gráficas se puede ver claramente que el NSGA-II no es capaz de alcanzar el frente de Pareto verdadero en las funciones de alta dimensionalidad como lo son **ZDT1** y **ZDT4** pues se encuentra muy lejos del verdadero frente. En contraste, nuestro algoritmo es capaz de alcanzar el verdadero frente en todos los casos. En cuanto a los problemas con restricciones, se observa que el NSGA-II obtiene mejores resultados, pues logra alcanzar la vecindad del frente de Pareto verdadero, aunque no siempre encuentra todas sus regiones, y la distribución de soluciones es de menor calidad cuando se compara con nuestro algoritmo.

Estos resultados nos indican que el NSGA-II, a pesar de ser un algoritmo muy competitivo, no es capaz de lograr convergencia en problemas de alta dimensionalidad cuando se realizan pocas evaluaciones de la función objetivo. Así mismo, en problemas con restricciones, el NSGA-II muestra un desempeño regular. Nuestro objetivo con el algoritmo propuesto en este artículo es presentar una alternativa que requiera de un número relativamente reducido de evaluaciones para alcanzar el frente de Pareto. Tal algoritmo puede ser útil en problemas del mundo real en los cuales el evaluar la función objetivo sea muy costoso (computacionalmente hablando).

IX. Conclusiones y Trabajo Futuro

En este artículo se ha presentado un nuevo algoritmo híbrido basado en la evolución diferencial y la teoría de conjuntos borrosos para resolver problemas multi-objetivo con y sin restricciones. El híbrido propuesto puede obtener resultados muy competitivos en diversos problemas de prueba, a pesar de realizar un número relativamente reducido de evaluaciones de la función de aptitud. Nuestro análisis de resultados indicó que nuestra propuesta obtiene mejores resultados (en general) que el NSGA-II, que es uno de los algoritmos evolutivos multi-objetivo más competitivos que se conocen a la fecha.

TABLA I

RESULTADOS DE DISTANCIA GENERACIONAL INVERTIDA (IGD), COBERTURA (SC), SPREAD (S) PARA LOS 6 PROBLEMAS ADOPTADOS. LOS MEJORES VALORES ESTÁN EN **negritas**. σ SE REFIERE A LA DESVIACIÓN ESTANDAR DE LAS 30 EJECUCIONES INDEPENDIENTES.

	IGD				SC				Spread			
	DEMORS		NSGA-II		DEMORS		NSGA-II		DEMORS		NSGA-II	
Function	$\overline{x}$	σ	$\overline{x}$	σ	$\overline{x}$	σ	$\overline{x}$	σ	$\overline{x}$	σ	$\overline{x}$	σ
ZDT1	**0.0057**	0.0075	0.0688	0.0072	**0.0032**	0.0066	0.9158	0.0549	**0.5046**	0.1518	0.6124	0.0405
ZDT4	**0.1267**	0.2947	6.8941	1.6652	**0.0025**	0.0069	0.3137	0.1787	**0.3550**	0.2992	0.9843	0.0114
KITA	**0.0062**	0.0094	0.0252	0.0407	0.3391	0.0743	**0.1136**	0.0489	**0.3106**	0.2345	0.4283	0.1926
WBM	**0.1179**	0.0660	0.1308	0.07770	0.3614	0.2130	**0.2250**	0.2015	**0.7297**	0.1308	0.7533	0.1175
OSY	0.4093	0.2634	**0.1570**	0.1411	**0.1256**	0.0992	0.4458	0.2239	**0.7349**	0.0645	0.8326	0.1267
TNK	**0.0007**	0.0002	0.0024	0.0015	**0.1298**	0.0449	0.1613	0.06195	0.9077	0.1722	**0.7480**	0.0540

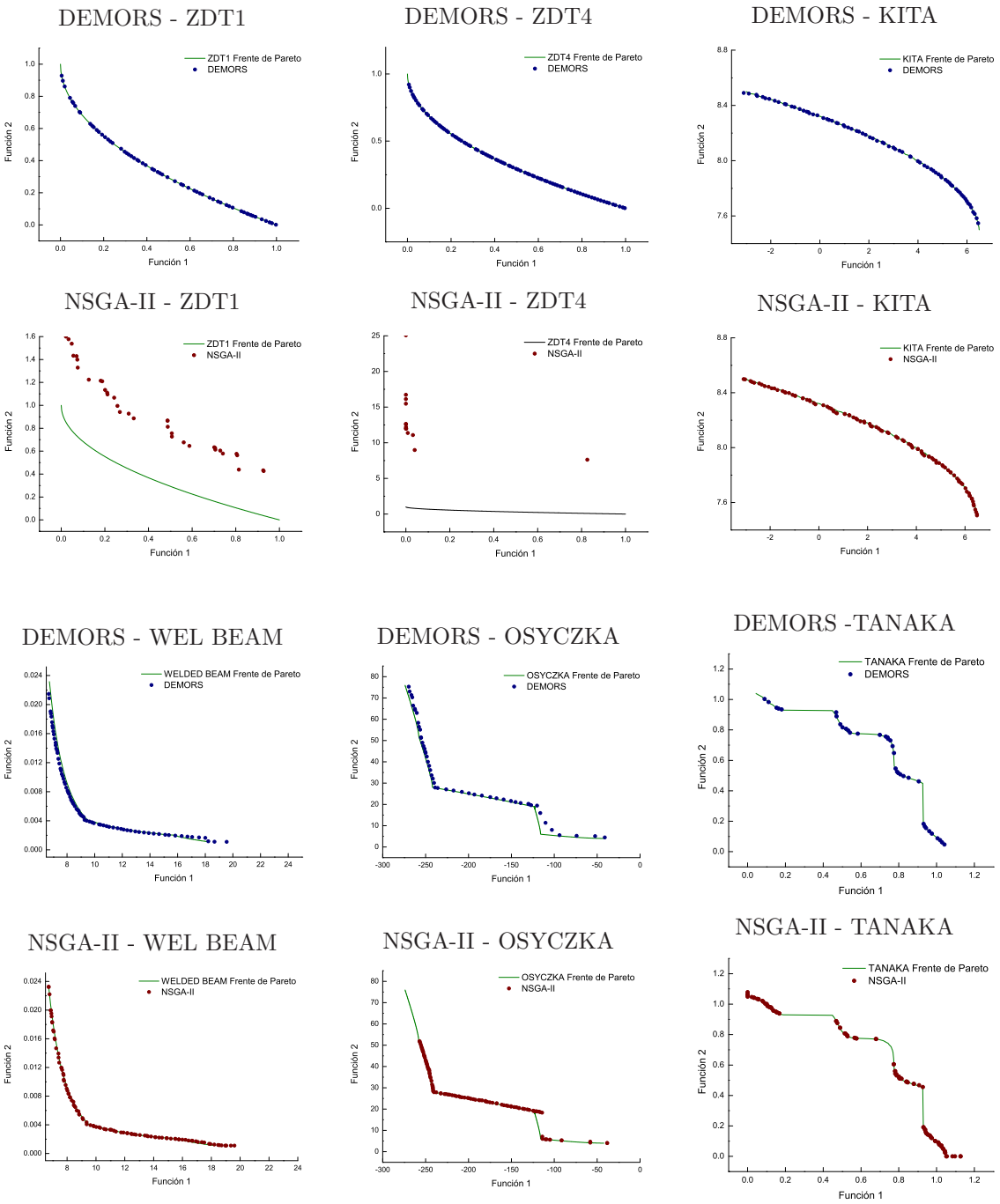

Fig. 3. Frentes de Pareto generados por DEMORS y NSGA-II para las funciones de prueba.

El híbrido presentado en este artículo tiene como objetivo principal de diseño el uso en problemas del mundo real en los cuales cada evaluación de la función objetivo tiene un costo computacionalmente elevado. En tales casos, puede sacrificarse la convergencia y la obtención de una buena distribución de soluciones, en aras de obtener aproximaciones razonablemente buenas en mucho menos tiempo.

Como parte de nuestro trabajo futuro, estamos explorando otros posibles motores de búsqueda para la fase I. Adicionalmente, nos interesa explorar a futuro otros esquemas de búsqueda más agresivos, que aumenten la presión de selección, a fin de lograr convergencia en la fase I con un número menor de evaluaciones. También nos interesa explorar esquemas más complejos de reglas para la exploración local que realizan los conjuntos borrosos, de manera que se haga una búsqueda más refinada en problemas con restricciones. Finalmente, es de nuestro mayor interés el poder evaluar este esquema en problemas del mundo real que hayan sido abordados previamente con otros algoritmos, a fin de poder verificar si nuestro esquema de hibridación realmente produce ahorros importantes en el costo computacional, sin sacrificar de manera significativa la convergencia.

Agradecimientos

El segundo autor agradece el apoyo otorgado por CONACyT, a través del proyecto 42435-Y.

Referencias

[1] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, New York, May 2002, ISBN 0-3064-6762-3.

[2] Rainer Storn and Kenneth Price, "Differential evolution - a simple and efficient adaptative scheme for global optimization over continuous spaces," Technical Report TR-95- 12, International Computer Science, Berkeley, California, March 1995.

[3] Kenneth Price, Rainer M. Storn, and Jouni A. Lampinen, *Differential Evolution : A Practical Approach to Global Optimization*, Springer-Verlag, 2006, ISBN 3-540-20950-6.

[4] Alfredo G. Hernández-Díaz, Luis V. Santana-Quintero, Carlos Coello Coello, Rafael Caballero, and Julián Molina, "A new proposal for multi-objective optimization using differential evolution and rough sets theory," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, New York, NY, USA, 2006, pp. 675–682, ACM Press.

[5] L.A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 1, pp. 338–353, Fall 1965.

[6] Z. Pawlak, "Rough sets," *International Journal of Computer and Information Sciences*, vol. 11, no. 1, pp. 341–356, Summer 1982.

[7] Z. Pawlak, *Rough Sets: Theoretical Aspects of Reasoning about Data*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991, ISBN 0-471-87339-X.

[8] T.Y. Lin, "Special issue on rough sets," *Journal of the Intelligent Automation and Soft Computing*, vol. 2, no. 2, pp. *, Fall 1996.

[9] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler, "Combining convergence and diversity in evolutionary multi-objective optimization," *Evolutionary Computation*, vol. 10, no. 3, pp. 263–282, Fall 2002.

[10] Alfredo G. Hernández-Díaz, Luis V. Santana-Quintero, Carlos A. Coello Coello, and Julián Molina, "Pareto-adaptive ε-dominance," Tech. Rep. EVOCINV-02-2006, Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México, March 2006.

[11] Hussein A. Abbass, "The Self-Adaptive Pareto Differential Evolution Algorithm," in *Congress on Evolutionary Computation (CEC'2002)*, Piscataway, New Jersey, May 2002, vol. 1, pp. 831–836, IEEE Service Center.

[12] Nateri K. Madavan, "Multiobjective Optimization Using a Pareto Differential Evolution Approach," in *Congress on Evolutionary Computation (CEC'2002)*, Piscataway, New Jersey, May 2002, vol. 2, pp. 1145–1150, IEEE Service Center.

[13] Saku Kukkonen and Jouni Lampinen, "An Extension of Generalized Differential Evolution for Multi-objective Optimization with Constraints," in *Parallel Problem Solving from Nature - PPSN VIII*, Birmingham, UK, September 2004, pp. 752–761, Springer-Verlag. Lecture Notes in Computer Science Vol. 3242.

[14] Antony W. Iorio and Xiaodong Li, "Solving rotated multi-objective optimization problems using differential evolution," in *AI 2004: Advances in Artificial Intelligence, Proceedings*. 2004, pp. 861–872, Springer-Verlag, Lecture Notes in Artificial Intelligence Vol. 3339.

[15] Tea Robič and Bodgan Filipič, "DEMO: Differential Evolution for Multiobjective Optimization," in *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, Eds., Guanajuato, México, March 2005, pp. 520–533, Springer. Lecture Notes in Computer Science Vol. 3410.

[16] Luis Vicente Santana-Quintero and Carlos A. Coello Coello, "An Algorithm Based on Differential Evolution for Multi-Objective Problems," *International Journal of Computational Intelligence Research*, vol. 1, no. 2, pp. 151–169, 2005.

[17] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, April 2002.

[18] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, Summer 2000.

[19] Hajime Kita, Yasuyuki Yabumoto, Naoki Mori, and Yoshikazu Nishikawa, "Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm," in *Parallel Problem Solving from Nature—PPSN IV*, Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, Eds., Berlin, Germany, September 1996, Lecture Notes in Computer Science, pp. 504–512, Springer-Verlag.

[20] K. M. Ragsdell and D. T. Phillips, "Optimal design of a class of welded structures using geometric programming," *Journal of Engineering for Industry Series B*, vol. B, no. 98, pp. 1021–1025, 1975.

[21] Andrzej Osyczka and Sourav Kundu, "A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm," *Structural Optimization*, vol. 10, pp. 94–99, 1995.

[22] Masahiro Tanaka, Hikaru Watanabe, Yasuyuki Furukawa, and Tetsuzo Tanino, "GA-Based Decision Support System for Multicriteria Optimization," in *Proceedings of the International Conference on Systems, Man, and Cybernetics*, Piscataway, NJ, 1995, vol. 2, pp. 1556–1561, IEEE.

[23] David A. Van Veldhuizen, *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*, Ph.D. thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.

[24] Kalyanmoy Deb, *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, 2001, ISBN 0-471-87339-X.