Un Algoritmo Coevolutivo para Optimización Multiobjetivo basado en *Clustering*[1]

M. M. Reyes Sierra y C. A. Coello Coello[2]

Proponemos una versión revisada de un algoritmo coevolutivo diseñado para optimización multiobjetivo y cuyo principal énfasis es la eficiencia. La idea básica del enfoque propuesto es concentrar los esfuerzos de la búsqueda en zonas prometedoras que van siendo descubiertas a lo largo del proceso evolutivo como resultado de un mecanismo de *clustering* que se aplica al conjunto de variables de decisión correspondientes al frente de Pareto conocido hasta el momento. El algoritmo propuesto es evaluado usando varias funciones de prueba tomadas de la literatura especializada y comparado con respecto a tres enfoques representativos del estado del arte en optimización evolutiva multiobjetivo.

1 Introducción

Ha habido pocos intentos (hasta recientemente) por desarrollar algoritmos evolutivos multiobjetivo que sean más eficientes (desde un punto de vista computacional) y el uso de conceptos coevolutivos en ese sentido ha sido aún menos frecuente [1]. En este artículo, extendemos una propuesta introducida en [2]. La idea principal del algoritmo propuesto es obtener información a lo largo del proceso evolutivo, que nos permita concentrar la búsqueda en sub-regiones prometedoras del espacio de búsqueda, y usar una sub-población en cada una de ellas. En cada generación, estas diferentes sub-poblaciones "cooperan" y "compiten" para finalmente obtener un único frente de Pareto. El tamaño de cada sub-población se ajusta de acuerdo a su contribución al frente de Pareto actual. El enfoque propuesto usa la malla adaptativa propuesta en [4] para almacenar los vectores no-dominados obtenidos a lo largo del proceso evolutivo.

Esta nueva versión de nuestro algoritmo lleva a cabo un análisis basado en *clustering* sobre el conjunto de variables de decisión del frente de Pareto conocido para identificar las sub-regiones promisorias del espacio de búsqueda. De esta manera, el número de poblaciones requeridas no crece exponencialmente con el número de variables de decisión, como en nuestra propuesta original [2].

[1] El primer autor agradece el apoyo de CONACyT a través de una beca escolar para cursar estudios de posgrado en el CINVESTAV-IPN. El segundo autor agradece el apoyo de CONACyT a través del proyecto número 42435-Y.

[2] CINVESTAV-IPN (Grupo de Computación Evolutiva) Departamento de Ing. Eléctrica, Sección de Computación Av. IPN No. 2508, Col. San Pedro Zacatenco, México D.F. 07300, MÉXICO mreyes@computacion.cs.cinvestav.mx, ccoello@cs.cinvestav.mx

2 Coevolución

La coevolución es un cambio evolutivo recíproco entre especies que interactúan. Las relaciones entre poblaciones de dos especies pueden describirse considerando todos sus posibles tipos de interacción. Los investigadores del área de Computación Evolutiva han desarrollado varios enfoques coevolutivos en los que normalmente dos o más especies se relacionan entre sí usando cualquiera de las posibles relaciones, principalmente competitivas (ej. [5]) o cooperativas (ej. [6]). Además, en la mayoría de los casos, las especies evolucionan de manera independiente a través de un algoritmo genético. La principal característica de estos algoritmos es que la aptitud de un individuo en una población depende de los individuos en otras poblaciones.

3 Trabajo Previo

Parmee y Watson [7] propusieron un enfoque colaborativo en el cual usan una población para cada función objetivo de un problema. En realidad, este método está creado para converger a una única solución compromiso. Sin embargo, a través del uso de penalizaciones, el algoritmo es capaz de mantener diversidad en la población. Las penalizaciones están relacionadas con el rango de variabilidad de los valores de las variables de decisión.

Keerativuttitumrong et.al. [8], Tan et.al. [11] e Iorio y Li [9], propusieron esquemas cooperativos en los cuales se usa una población por cada una de las variables de decisión del problema. Para poder evaluar un individuo en una población, se deben seleccionar individuos de las demás poblaciones para poder formar una solución completa al problema.

En [8], la evolución de cada una de las poblaciones se lleva a cabo a través del algoritmo MOGA de Fonseca y Fleming [3]. El método propuesto en [11] usa un archivo externo para almacenar y actualizar las soluciones no-dominadas encontradas hasta el momento y estas soluciones guían la búsqueda hacia las zonas menos explotadas del espacio de búsqueda. Finalmente, en [9] la evolución de cada una de las poblaciones se lleva a cabo a través del algoritmo NSGA-II [10]. Después de cada generación, se usa un ordenamiento basado en no-dominancia sobre todas las poblaciones de padres e hijos para poder determinar las nuevas poblaciones de padres.

4 Descripción de Nuestra Propuesta

Como en [2], la idea principal de nuestra propuesta es concentrar los esfuerzos de la búsqueda únicamente en las sub-regiones promisorias del espacio de búsqueda. Tales regiones se determinan mediante el uso de un análisis de *clustering*. El proceso evolutivo de nuestro algoritmo está dividido en dos etapas principales:

Primera Etapa. Durante la primera etapa, el algoritmo explora todo el espacio de búsqueda usando una única población de individuos. Dentro de esta población, los individuos son seleccionados a través de la jerarquización de Pareto propuesta por Fonseca y Fleming [3]. Además, el algoritmo usa la malla adaptativa propuesta en [4]. Al final de esta primera etapa, el algoritmo analiza el frente de Pareto actual (almacenado en la malla adaptativa) para poder determinar las regiones promisorias del espacio de búsqueda.

```
1 gen=0
2 poblaciones=0
3 mientras (gen<Gmax) {
      si (gen ³ Gmax/4)
4        si (gen=Gmax/4, Gmax/2,3Gmax/4
           o $ xÎ pob_cero : xÎ frente Pareto actual) {
5                chequeo_poblaciones()
6                clustering()
7                construir_nuevas_poblaciones()}
8     para (i=1;i£poblaciones;i++)
9        si (población i contribuye al frente
                de Pareto actual)
10               evoluciona_y_compite(i)
11    elitismo()
12    reasignación_de_recursos()
13    gen++ }
```

Figura 1. Pseudocódigo de nuestro algoritmo.

En la versión previa de nuestro algoritmo [2], este análisis consistía en estudiar el conjunto de valores de las variables de decisión correspondientes al frente de Pareto actual para poder determinar qué variables del problema eran más críticas. Como resultado de este análisis, se sub-dividía el espacio de búsqueda en diferentes regiones y se generaba un conjunto de poblaciones nuevas de manera que a cada región se le asignaba una población nueva.

En esta nueva versión, llevamos a cabo una análisis de *clustering* sobre los valores de la variables de decisión correspondientes al frente de Pareto actual para poder determinar las regiones promisorias del espacio de búsqueda (figura 1, línea 6). El análisis se lleva a cabo de manera independiente para cada variable. Una vez que conocemos los *clusters* correspondientes a cada una de las variables, procedemos a formar un conjunto de poblaciones nuevas. Cada *cluster* proporciona un intervalo específico (en el cual están contenidas las soluciones que pertenecen a él). De esta manera, para cada variable, obtenemos un conjunto de intervalos. Posteriormente, creamos un conjunto de sub-regiones con base en el producto cartesiano de los conjuntos de intervalos correspondientes a cada variable. A cada sub-región que tenga individuos en el frente de Pareto se le asigna una población nueva (figura 1, línea 7). Ver figura 2.

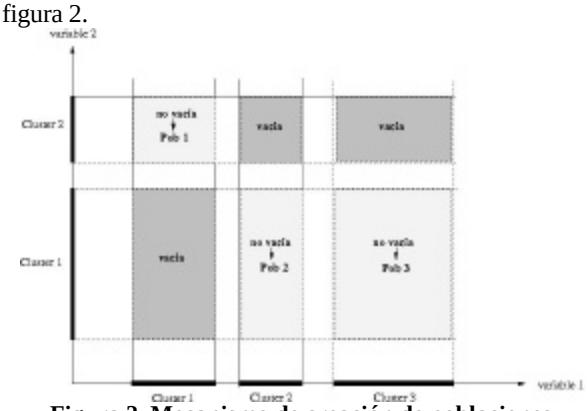

Figura 2. Mecanismo de creación de poblaciones.

Finalmente, usamos una población extra (llamada población cero --pob_{cero}--) para continuar buscando buenas soluciones en las sub-regiones que no están siendo exploradas por las otras poblaciones (para este fin usamos un operador de mutación especial que no permite a los individuos pertenecer a sub-regiones que ya están siendo exploradas).

Segunda Etapa. Al finalizar la primera etapa, tenemos un conjunto de poblaciones buscando en diferentes regiones del espacio de búsqueda. En cada generación, la evolución de todas las poblaciones se lleva a cabo de manera independiente y, posteriormente, los elementos no-dominados de cada población se envían a la malla adaptativa donde "cooperan" y "compiten" para formar un único frente de Pareto.

Nuestro algoritmo es *elitista* (figura 1, línea 11), porque después de la primera generación de la segunda etapa, todas las poblaciones que no proporcionan individuos al frente de Pareto actual son automáticamente eliminadas. Por otra parte, los tamaños de las poblaciones sobrevivientes se ajustan de manera que a cada población se le agregan o eliminan individuos hasta que su tamaño sea proporcional con su contribución al frente de Pareto actual (figura 1, línea 12). Los individuos que son agregados/eliminados son generados/escogidos de manera aleatoria. Así, las poblaciones compiten entre ellas para obtener la mayor cantidad de individuos posible. Ver figura 3.

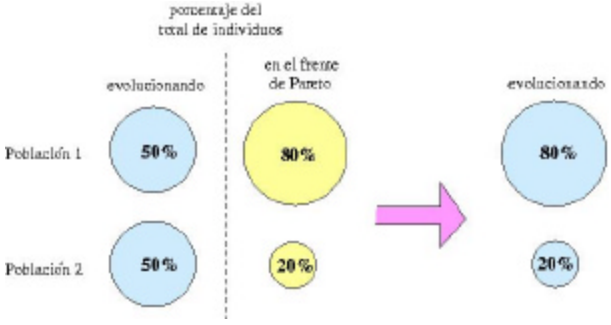

Figura 3. Reasignación de recursos.

Chequeo. Durante la segunda etapa, se lleva a cabo un *chequeo* en momentos específicos del proceso evolutivo. (figura 1, línea 4). El chequeo se lleva a cabo como antes (ver figura 1) [2], pero también cuando la población cero proporciona algún individuo nuevo al frente de Pareto actual.

Cuando se lleva a cabo un *chequeo*, procedemos a determinar cuántas y cuáles poblaciones pueden continuar (aquellas que continúan proporcionando individuos al frente de Pareto actual, es decir, "buenas" poblaciones)(figura 1, línea 5).

Al igual que al final de la primera etapa, se procede a realizar el análisis de *clustering* sobre los valores de las variables de decisión del frente de Pareto actual, y se construye un conjunto de nuevas poblaciones. Los individuos no-dominados de las "buenas" poblaciones se conservan. Todos los buenos individuos se distribuyen en las poblaciones recientemente generadas. Se sigue aplicando el elitismo y el tamaño de cada población se ajusta con base en el criterio descrito anteriormente. Nótese, sin embargo, que hemos definido un tamaño de población mínimo y que, después de cada *chequeo*, cada población debe tener al menos ese tamaño.

Análisis de *Clustering*. Se implementó un algoritmo de *clustering* basado en la naturaleza del algoritmo *k-medias* [12]. Este algoritmo comienza con *k* centroides aleatorios y coloca cada punto del conjunto analizado en el *cluster* correspondiente al centroide más cercano. Posteriormente, se calculan las *medias* de cada *cluster* y el proceso se repite hasta que no haya cambios (tomando a las medias calculadas como nuevos centroides). El algoritmo se detiene cuando se encuentra el mínimo de la suma de las distancias entre cada punto y su centroide correspondiente.

Este algoritmo tiene dos desventajas: (1) depende de los centroides iniciales y (2) requiere el número de *clusters* deseados. Por esta razón, se hicieron dos modificaciones al algoritmo con el fin de superar tales dificultades.

Con respecto a la primera desventaja, buscamos un punto que pueda ser un nuevo centroide: Sea x_i un punto que pertenece a un *cluster* con centroide c_i, y d_{min} la distancia mínima entre dos centroides. Si $d(x_i,c_i)>d_{min}$, x_i será un nuevo centroide. Para mantener constante el número de *clusters*, una vez que se ha seleccionado un punto para ser un nuevo centroide, se escoge uno de los dos centroides más cercanos (entre sí) y se elimina. Con respecto a la segunda desventaja, se usa el siguiente mecanismo [12]: Sea x_i un punto que pertenece al *cluster q* (con centroide c_q) y *K* el número total de *clusters* actual. La distancia promedio entre x_i y los *K* centroides es: $\bar{d}_i = \frac{1}{K}\sum_{k=1}^{K} d(x_i, c_k)$. Creamos un *cluster* nuevo con centroide x_i cuando $\left|d(x_i, c_q) - \bar{d}_i\right| \le \bar{d}_i T$, donde *T* es tal que $0<T<1$. Cuanto mayor sea el valor de *T*, mayor será el número de *clusters* creados. Dado que el mecanismo anterior crea nuevos *clusters*, se procede a eliminar *clusters* cuando los centroides correspondientes son muy cercanos: Si la distancia entre dos centroides es menor que *T* veces la distancia promedio entre centroides, uno de ellos es eliminado.

Parámetros. Nuestro algoritmo requiere los siguientes parámetros (el parámetro *T* del algoritmo de *clustering* se fijó con valor de 1):

(1) Probabilidad de cruza (p_c) y probabilidad de mutación (p_m), (2) Número máximo de generaciones (Gmax) y (3) Tamaño de la población inicial (popsize_init) a usarse durante la primera etapa y tamaño mínimo de las poblaciones secundarias a usarse durante la segunda etapa (popsize_sec).

5 Resultados

Se llevaron a cabo comparaciones cuantitativas (adoptando cuatro métricas) y cualitativas (graficando los frentes de Pareto producidos) con respecto a tres Algoritmos Evolutivos Multi-Objetivo que son representativos del estado del arte en el área: el *Strength Pareto Evolutionary Algorithm 2* (SPEA2) [13], el *Pareto Archived Evolution Strategy* (PAES) [4] y el *Nondominated Sorting Genetic Algorithm II* (NSGA-II) [10].

Para nuestro estudio comparativo, se implementaron cuatro métricas: (1) Razón de Error (RE), (2) Distancia Generacional (DG), (3) Espaciado (E) y (4) Cobertura (C). Las primeras tres métricas son unarias y la última es binaria. La métrica RE calcula la razón de individuos generados por un algoritmo, que no pertenecen al verdadero frente de Pareto. La métrica DC calcula la distancia del frente generado al verdadero frente de Pareto. La métrica E calcula la distribución entre las soluciones generadas por un algoritmo. Finalmente, la métrica C(X,Y) calcula la razón de individuos generados por el algoritmo Y que son dominados por soluciones del algoritmo X. Ver [1] para detalles acerca de estas métricas. Para cada una de las funciones de prueba que se muestran más adelante, se realizaron 30 corridas por algoritmo y un total de 10000 evaluaciones de la función objetivo en cada corrida.

Los parámetros para el SPEA2 fueron $\alpha=\beta=\lambda=100$ y 100 generaciones. Para el NSGA-II se usó popsize=100 y 100 generaciones y en el caso de PAES se hicieron 10000 iteraciones. Todos los algoritmos usaron representación binaria, mutación unifome con una probabilidad (p_m) igual a *1/codesize* y cruza uniforme con probabilidad (p_c) igual a 0.8. Los frentes de Pareto que se mostrarán corresponden con la mediana de las 30 corridas con respecto a la métrica RE.

Con respecto al manejo de restricciones, en el caso del NSGA-II se usó el esquema original con el que cuenta. Sin embargo, dado que los demás algoritmos (incluyendo el nuestro) no cuentan con tal mecanismo, en todos los casos se aplicó una función de penalización dinámica simple sobre los valores de las funciones objetivo para cada individuo no factible.

Función de Prueba 1

Min $f_1(x_1, x_2) = \sqrt[8]{x_1^2 + x_2^2}$

$f_2(x_1, x_2) = \sqrt[4]{(x_1 - 0.5)^2 + (x_2 - 0.5)^2}$

$-5.0 \le x_1, x_2 \le 10.0$

En este caso, nuestro algoritmo usó popsize_init =100, popsize_sec =36 (40 gen).

La tabla 1 muestra los valores de las métricas para cada uno de los algoritmos comparados. Los frentes correspondientes a esta función se muestran en la figura 4.

Función 1					
		CO-MOEA	SPEA2	PAES	NSGA-II
RE	Mejor	0.08	0.00	0.15	0.15
	Mediana	0.26	0.26	0.35	0.24
	Peor	0.48	0.35	1.00	0.36
	Promedio	0.27	0.26	0.38	**0.25**
	Desv. Est.	0.1112	0.0675	0.1990	0.0465
DG	Mejor	0.0003	0.0003	0.0004	0.0008
	Mediana	0.0006	0.0004	0.0008	0.0014
	Peor	0.0013	0.0012	0.0926	0.0027
	Promedio	0.0006	**0.0004**	0.0042	0.0014
	Desv. Est.	0.0003	0.0002	0.0168	0.0005
E	Mejor	0.0026	0.0035	0.0001	0.0056
	Mediana	0.0052	0.0069	0.0103	0.0071
	Peor	0.0074	0.0199	0.0288	0.0224
	Promedio	**0.0052**	0.0081	0.0107	0.0079
	Desv. Est.	0.0012	0.0042	0.0067	0.0030
C(X,		CO-MOEA	SPEA2	PAES	NSGA-II
CO-MOEA		0.00	0.20	0.28	0.17
SPEA2		0.06	0.00	0.30	0.12
PAES		0.04	0.13	0.00	0.05
NSGA-II		0.09	0.15	0.26	0.00
Promedio		**6%**	16%	28%	11%

Tabla 1. Resultados correspondientes a la función 1.

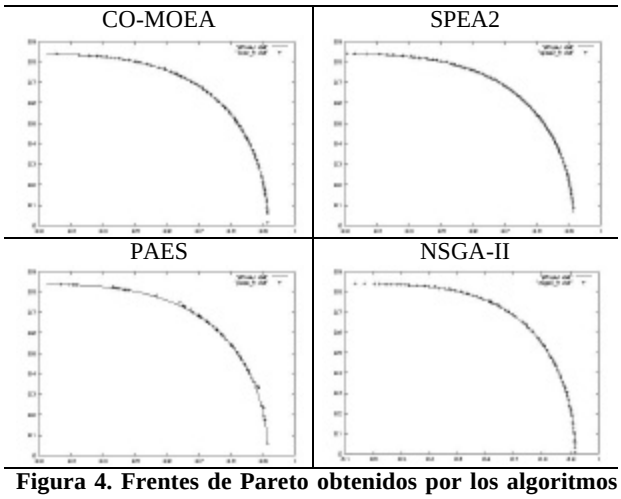

Figura 4. Frentes de Pareto obtenidos por los algoritmos comparados, para la función 1.

Función de Prueba 2

Min $f_1(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 1)^2 + 2$

$f_2(x_1, x_2) = 9x_1 - (x_2 - 1)^2$

$g_1(x_1, x_2) = x_1^2 + x_2^2 - 225 \le 0$

sujeto a $g_2(x_1, x_2) = x_1 - 3x_2 + 10 \le 0$

$-20.0 \le x_1, x_2 \le 20.0$

En este caso, nuestro algoritmo usó popsize_init =80, popsize_sec =30 (30 gen).

La tabla 2 muestra los valores de las métricas para cada uno de los algoritmos comparados. Los frentes correspondientes a esta función se muestran en la figura 5.

Función 2					
		CO-MOEA	SPEA2	PAES	NSGA-II
RE	Mejor	0.09	0.19	0.21	0.30
	Mediana	0.20	0.33	0.43	0.43
	Peor	0.44	0.70	1.00	0.61
	Promedio	**0.22**	0.35	0.56	0.43
	Desv. Est.	0.0815	0.1067	0.2993	0.0782
DG	Mejor	0.0173	0.0221	0.0153	0.0280
	Mediana	0.0259	0.0322	0.0328	0.0454
	Peor	0.0935	0.0838	6.2154	0.0619
	Promedio	**0.0314**	0.0374	0.3822	0.0457
	Desv. Est.	0.0161	0.0137	1.1349	0.0083
E	Mejor	1.2892	0.4774	1.2016	1.3032
	Mediana	1.7032	0.6579	2.8088	1.7254
	Peor	2.2732	1.1101	22.495	1.9354
	Promedio	1.7016	**0.6913**	4.2799	1.7049
	Desv. Est.	0.2232	0.1314	4.0672	0.1442
C(X,		CO-MOEA	SPEA2	PAES	NSGA-II
CO-MOEA		0.00	0.05	0.06	0.12
SPEA2		0.00	0.00	0.09	0.08
PAES		0.01	0.04	0.00	0.03
NSGA-II		0.01	0.08	0.04	0.00
Promedio		**1%**	6%	6%	8%

Tabla 2. Resultados correspondientes a la función 2.

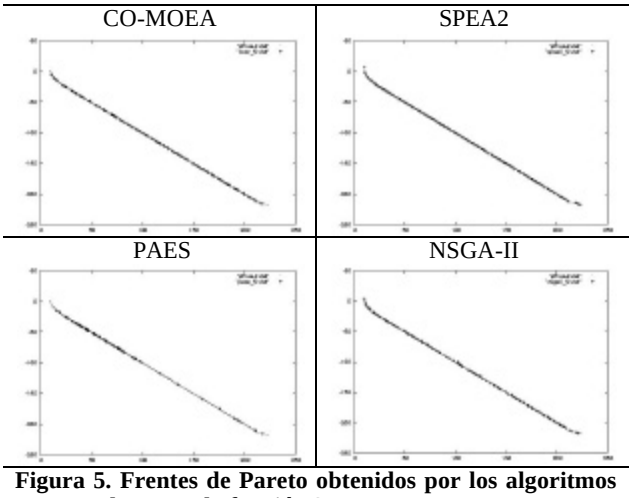

Figura 5. Frentes de Pareto obtenidos por los algoritmos comparados, para la función 2.

Función de Prueba 3

Min $f_1(x_1, x_2) = \frac{(x_1 - 2)^2}{2} + \frac{(x_2 + 1)^2}{13} + 3$

$$f_2(x_1, x_2) = \frac{(x_1 + x_2 - 3)^2}{175} + \frac{(2x_2 - x_1)^2}{17} - 13$$

$$f_3(x_1, x_2) = \frac{(3x_1 - 2x_2 + 4)^2}{8} + \frac{(x_1 - x_2 + 1)^2}{27} + 15$$

sujeto a
$$g_1(x_1, x_2) = x_2 + x_1 - 4 < 0$$
$$g_2(x_1, x_2) = -x_1 - 1 < 0$$
$$g_3(x_1, x_2) = x_1 - 2 - x_2 < 0$$
$$-4.0 \le x_1, x_2 \le 4.0$$

En este caso, nuestro algoritmo usó popsize_init =100, popsize_sec =30 (30 gen). La tabla 3 muestra los valores de las métrica para cada uno de los algoritmos comparados. Los frentes correspondientes a esta función se muestran en la figura 6.

Función 3					
		CO-MOEA	SPEA2	PAES	NSGA-II
RE	Mejor	0.04	0.13	0.00	0.10
	Mediana	0.07	0.22	0.05	0.33
	Peor	0.12	0.33	0.99	0.51
	Promedio	**0.07**	0.22	0.17	0.31
	Desv. Est.	0.0229	0.0409	0.2641	0.1210
DG	Mejor	0.0017	0.0025	0.0013	0.0023
	Mediana	0.0020	0.0033	0.0035	0.0037
	Peor	0.0030	0.0042	0.3506	0.0088
	Promedio	**0.0021**	0.0033	0.0394	0.0049
	Desv. Est.	0.0002	0.0004	0.0856	0.0021
E	Mejor	0.1206	0.0441	0.0117	0.0964
	Mediana	0.1901	0.0777	0.1332	0.1253
	Peor	0.2837	0.1922	5.1365	0.1622
	Promedio	0.1915	**0.0845**	0.5629	0.1242
	Desv. Est.	0.0460	0.0312	1.2163	0.0194
C(X,		CO-MOEA	SPEA2	PAES	NSGA-II
CO-MOEA		0.00	0.04	0.03	0.09
SPEA2		0.01	0.00	0.01	0.17
PAES		0.02	0.02	0.00	0.13
NSGA-II		0.03	0.04	0.04	0.00
Promedio		**2%**	3%	3%	13%

Tabla 3. Resultados correspondientes a la función 3.

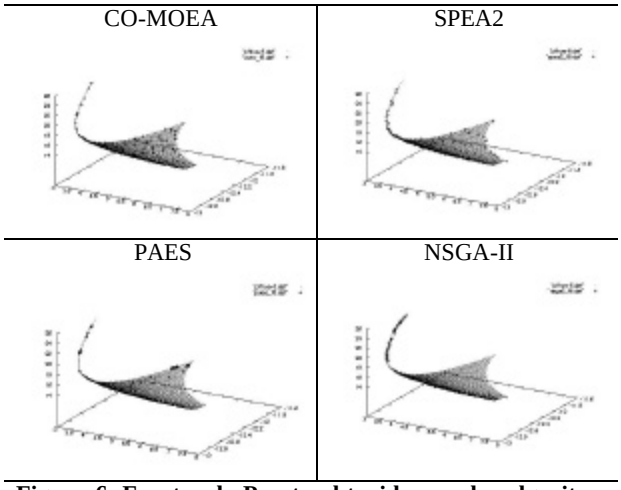

Figura 6. Frentes de Pareto obtenidos por los algoritmos comparados, para la función 3.

6 Discusión de Resultados

En la primera función, los mejores resultados con respecto a la métrica RE los produjo el SPEA2, seguido muy de cerca por los resultados del NSGA-II y nuestro algoritmo. El mejor resultado de PAES es muy similar al mejor resultado del algoritmo NSGA-II, sin embargo, su mediana y su peor resultado son muy pobres. Con respecto a la métrica DG, los mejores resultados son del algoritmo SPEA2 y nuestro algoritmo (ambos son muy similares). El peor resultado promedio nuevamente corresponde a PAES. Respecto a la métrica E, nuestro algoritmo obtuvo los mejores resultados, seguido por el SPEA2, NSGA-II y PAES. Finalmente, con respecto a la métrica C, nuestro algoritmo obtuvo los mejores resultados porque sólo un promedio de 6% de sus puntos fueron dominados por puntos obtenidos por los otros algoritmos, seguido (en promedio) por el NSGA-II con un 11%, SPEA2 con un 16% y PAES con 28%.

Los resultados en la segunda y tercera función son muy similares. Con respecto a las métricas RE y DG, nuestro algoritmo obtuvo los mejores resultados, seguido por el SPEA2, NSGA-II y PAES. Respecto a la métrica E, el algoritmo SPEA2 obtuvo los mejores resultados, seguido (en promedio) por nuestro algoritmo, NSGA-II y PAES en la segunda función, y por NSGA-II, nuestro algoritmo y PAES en la tercera función. En estas funciones, para las tres métricas, aunque PAES obtuvo un mejor resultado que el NSGA-II, el peor resultado de PAES es muy alto.

Al igual que en la primera función, en estas dos funciones nuestro algoritmo obtuvo los mejores resultados con respecto a la métrica C: en la segunda función obtuvo un promedio de 1% de sus puntos dominados por otros algoritmos mientras los porcentajes de SPEA2, PAES y NSGA-II fueron 6%, 6% y 8%, respectivamente, y, en la tercera función obtuvo un promedio de 2% de sus puntos dominados por otros algoritmos mientras los porcentajes de SPEA2, PAES y NSGA-II fueron 3%, 3% y 13%.

En general, nuestro algoritmo obtuvo buenos resultados con respecto a las primeras dos métricas (RE y DG): nuestro algoritmo obtuvo los mejores resultados (funciones 2 y 3) o resultados muy cercanos al mejor (función 1). Con respecto a la tercera métrica (E) el algoritmo SPEA2 obtuvo los mejores resultados en dos de las tres funciones, y nuestro algoritmo obtuvo los mejores resultados en la función 1.

De esta manera, podemos concluir que nuestro algoritmo fue capaz de generar una buena cantidad de puntos pertenecientes al verdadero frente de Pareto de todas las funciones de prueba usadas o bien obtener buenas aproximaciones. Quedando por mejorar la distribución de los conjuntos obtenidos.

Finalmente, respecto a la métrica C, nuestro algoritmo obtuvo los mejores resultados en las tres funciones. En

promedio, considerando las tres funciones de prueba presentadas, nuestro algoritmo obtuvo los mejores resultados con respecto a la métrica C, con un promedio de 3% de sus puntos dominados por los otros algoritmos, seguido de SPEA2 con 8%, NSGA-II con 11% y PAES con 12%. De esta manera, un 97% de los puntos generados por nuestro algoritmo son no-dominados con respecto a los puntos generados por los otros algoritmos, es decir, son de mejor o igual calidad. Así pues, concluimos que nuestro algoritmo obtuvo las mejores aproximaciones a los frentes verdaderos de cada una de las funciones de prueba presentadas.

7 Conclusiones y Trabajo Futuro

Presentamos una nueva versión de un algoritmo coevolutivo multiobjetivo cuya principal idea es detectar las sub-regiones promisorias del espacio de búsqueda con el fin de concentrar los esfuerzos de la búsqueda en ellas. Para este propósito, el algoritmo propuesto aplica una técnica de *clustering* sobre el conjunto de variables de decisión del frente de Pareto conocido. El enfoque propuesto se validó usando varias funciones de prueba de la literatura especializada. Nuestro estudio comparativo mostró que el enfoque propuesto es competitivo con respecto a tres algoritmos que son representativos del estado del arte en el área.

Como parte de nuestro trabajo futuro, consideramos importante disminuir la presión de selección introducida por nuestro esquema de elitismo, dado que en la versión actual de nuestro algoritmo esto puede causar convergencia prematura cuando existe un falso atractor. Dado que el análisis de *clustering* fue aplicado a cada variable de manera independiente, podríamos usar un análisis similar pero aplicado al espacio completo (n-dimensiones) de las variables de decisión.

Referencias

[1] Coello Coello, C. A., Van Veldhuizen, D. A. and Lamont, G. B. Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, New York, 2002.

[2] Coello Coello, C. A. and Reyes Sierra, M., A Coevolutionary Multi-Objective Evolutionary Algorithm, in *Proceedings of 2003 Congress on Evolutionary Computation*, Vol. 1, pp. 482--489, IEEE Press, Canberra, Australia, 2003.

[3] Fonseca C.M. and Fleming P.J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization, In Forrest S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416-423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[4] Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy *Evolutionary Computation*, 8(2): 149-172, 2000.

[5] Paredis, J.: Coevolutionary algorithms. In Bäck, T., Fogel, D.B. Michalewicz, Z., eds.: The Handbook of Evolutionary Computation, 1st. supplement. Institute of Physics Publishing and Oxford University Press (1998) 225-238.

[6] Potter, M., Jong., K.D.: A cooperative coevolutionary approach to function optimization. *In Proceedings from the Fifth Parallel Problem Solving from Nature*, Jerusalem, Israel, Springer-Verlag (1994) 530-539.

[7] Parmee I.C. and Watson A.H.. Preliminary Airframe Design Using Co-Evolutionary Multiobjective Genetic Algorithms, In Banzhaf, W. et al. editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99)*, volume 2, pages 1657-1665, San Francisco, California, July 1999. Morgan Kaufmann.

[8] Keerativuttiumrong N., Chaiyaratana N. and Varavithya V. Multi-objective Co-operative Co-evolutionary Genetic Algorithm, in Merelo J. et al. ed. *Proceedings of Parallel Problem Solving from Nature---PPSN VII*, pp. 288--297, Springer-Verlag, LNCS No. 2439, Granada, Spain 2002.

[9] Iorio A. and Li X. A Cooperative Coevolutionary Multiobjective Algorithm Using Non-dominated Sorting, in Deb K. et al. (editors), *Proceedings of the Genetic and Evolutionary Computation Conference. Part I*, Springer-Verlag, LNCS Vol. 3102, pp. 537--548, Seattle, Washington, USA 2004.

[10] Deb K., Pratap A., Agarwal S., and Meyarivan T. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA--II, *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp. 182--197, April 2002.

[11] Tan K., Chew Y., Lee T. and Yang Y. A Cooperative Coevolutionary Algorithm for Multiobjective Optimization, in *Proceedings of the 2003 IEEE International Conference on Systems, Man and Cybernetics*, Vol. 1, pp. 390--395, IEEE Press, 2003

[12]Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall, New Jersey (1988).

[13] Zitzler E., Laumanns M. and Thiele L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm, in Giannakoglou K. et al., (eds.) *EUROGEN 2001, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pp. 95--100, Athens, Greece, 2002.