Un Algoritmo Cultural para Optimización Evolutiva Multiobjetivo

Ricardo Landa Becerra y Carlos A. Coello Coello

Resumen— **En este artículo se realiza la primera propuesta para extender un algoritmo cultural de manera que pueda lidiar con problemas multiobjetivo. Dicha propuesta utiliza programación evolutiva, jerarquización de Pareto y elitismo (una población secundaria). La propuesta es validada usando varios ejemplos tomados de la literatura especializada. Los resultados se comparan con respecto al NSGA-II, que es representativo del estado del arte en optimización evolutiva multiobjetivo.**

Palabras clave— **optimización evolutiva multiobjetivo, algoritmos culturales, programación evolutiva.**

I. Introducción

En años recientes, las técnicas de computación evolutiva han sido ampliamente aplicadas a problemas de optimización multiobjetivo, obteniendo resultados alentadores [1], [2]. Una ventaja que presentan las técnicas de computación evolutiva sobre otras técnicas, es su facilidad para generar un conjunto de soluciones en una sola ejecución. En optimización multiobjetivo, esto es importante, porque al tomador de decisiones se le debe presentar el conjunto más completo posible de soluciones compromiso entre todos los objetivos del problema, para que él las evalúe y elija la solución final que se aplicará en práctica.

Los algoritmos culturales [3] son una técnica que intentan incorporar conocimiento del dominio extraido durante el mismo proceso de búsqueda, con el objetivo hacer más eficiente dicho proceso. Los algoritmos culturales han sido aplicados con éxito en varios tipos de problemas de optimización (por ejemplo, optimización mono-objetivo con restricciones [4], [5], [6], [7]), pero hasta ahora no se había reportado ningún algoritmo cultural para optimización multiobjetivo que utilice jerarquización de Pareto.

CINVESTAV-IPN, Evolutionary Computation Group
Depto. de Ing. Elect./Sección de Computación,
Av. IPN No. 2508, Col. San Pedro Zacatenco
México, D. F. 07300,
rlanda@computacion.cs.cinvestav.mx,
ccoello@cs.cinvestav.mx

II. Conceptos Básicos

Definition 1 (Problema de Optimización Multiobjetivo (POM) General): *Encontrar el vector* $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ *que satisfaga las* m *restricciones de desigualdad:*

$$g_i(\vec{x}) \geq 0 \quad i = 1, 2, \ldots, m \tag{1}$$

las p *restricciones de igualdad*

$$h_i(\vec{x}) = 0 \quad i = 1, 2, \ldots, p \tag{2}$$

y optimice el vector función

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T \tag{3}$$

donde $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ *es el vector de las variables de decisión.* □

Definition 2 (Optimalidad de Pareto): *Un punto* $\vec{x}^* \in \Omega$ *(*Ω *es la región factible) es* **óptimo en el sentido de Pareto** *si para cada* $\vec{x} \in \Omega$ *y* $I = \{1, 2, \ldots, k\}$ *tanto,*

$$\forall_{i \in I}(f_i(\vec{x}) = f_i(\vec{x}^*)) \tag{4}$$

o, haya al menos un $i \in I$ *tal que*

$$f_i(\vec{x}) > f_i(\vec{x}^*) \tag{5}$$

□

En palabras, esta definición dice que $\vec{x}^*$ es un óptimo de Pareto si no existe vector $\vec{x}$ factible que haga decrementar algún criterio sin causar un aumento simultáneo en al menos algún otro. El significado de la frase "óptimo de Pareto" se considera con respecto a todo el espacio de las variables de decisión a menos que se indique lo contrario.

Definition 3 (Dominancia de Pareto): *Un vector* $\vec{u} = (u_1, \ldots, u_k)$ *se dice que domina a otro* $\vec{v} = (v_1, \ldots, v_k)$ *(denotado por* $\vec{u} \preceq \vec{v}$*) si y sólo si* u *es parcialmente menor que* v*, por ejemplo,* $\forall i \in \{1, \ldots, k\},\ u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$. □

Definition 4 (Conjunto de óptimos de Pareto): *Dado un POM* $\vec{f}(x)$*, el conjunto de óptimos de Pareto (*$\mathcal{P}^*$*) se define como:*

$$\mathcal{P}^* := \{x \in \Omega \mid \neg\exists\, x' \in \Omega \ \ \vec{f}(x') \preceq \vec{f}(x)\}. \tag{6}$$

□

Definition 5 (Frente de Pareto): *Para un POM $\vec{f}(x)$ y conjunto de óptimos de Pareto $\mathcal{P}^*$ dado, el frente de Pareto ($\mathcal{PF}^*$) se define como:*

$$\mathcal{PF}^* := \{\vec{u} = \vec{f} = (f_1(x), \ldots, f_k(x)) \mid x \in \mathcal{P}^*\}. \quad (7)$$

□

III. Nociones de Algoritmos Culturales

Los algoritmos culturales fueron desarrollados por Robert G. Reynolds, como un complemento a la metáfora que usan los algoritmos evolutivos, que se habían concentrado en los conceptos genéticos, y de la selección natural [3]. Los algoritmos culturales están basados en las teorías de algunos sociólogos y arqueólogos, que han tratado de modelar la evolución cultural. Tales investigadores indican que la evolución cultural puede ser vista como un proceso de herencia en dos niveles: el nivel micro-evolutivo, que consiste en el material genético heredado por los padres a sus descendientes, y el nivel macro-evolutivo, que es el conocimiento adquirido por los individuos a través de las generaciones, y que una vez codificado y almacenado, sirve para guiar el comportamiento de los individuos que pertenecen a una población [8], [9]. Reynolds intenta captar ese fenómeno de herencia doble en los algoritmos culturales [3]. El objetivo es incrementar las tasas de aprendizaje o convergencia, y, de esta manera, que el sistema responda mejor a un gran número de problemas [10]. Los algoritmos culturales operan en dos espacios. Primero, el espacio de la población, como en todos los métodos de computación evolutiva, en el que se tiene un conjunto de individuos. Cada individuo tiene un conjunto de características independientes de los otros, con las que es posible determinar su aptitud. A través del tiempo, tales individuos podrán ser reemplazados por algunos de sus descendientes, obtenidos a partir de un conjunto de operadores aplicados a la población. Y el segundo espacio es el de creencias, donde se almacenarán los conocimientos que han adquirido los individuos en generaciones anteriores. La información contenida en este espacio debe ser accesible a cualquier individuo, quien puede utilizarla para modificar su comportamiento. Para unir ambos espacios, se establece un protocolo de comunicación, que dicta las reglas del tipo de información que se debe intercambiar entre los espacios. Por ejemplo, para la actualización del espacio de creencias se incorporan las experiencias individuales de un grupo selecto de individuos, el cual se obtiene con la función de *aceptación* de entre

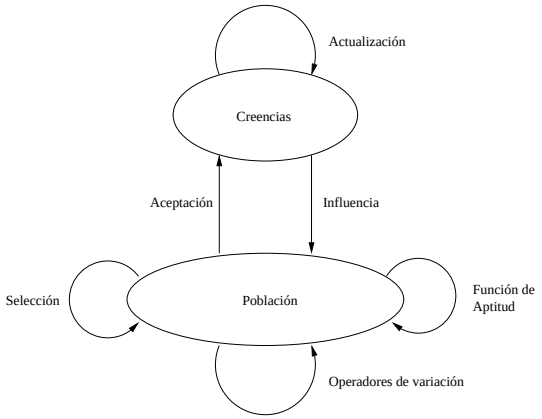

Fig. 1. Espacios de un algoritmo cultural

toda la población. Por otro lado, los operadores de variación de los individuos (como la recombinación o la mutación), así como la selección, son modificados por la función de *influencia*. La función de influencia ejerce cierta presión, para que los individuos resultantes de la aplicación de los operadores se acerquen a los comportamientos deseables, y se alejen de los indeseables, según la información almacenada en el espacio de creencias. Estas dos funciones, la de aceptación y la de influencia, son mediante las cuales se establece la comunicación entre los espacios de la población y de creencias. Estas interacciones entre los espacios pueden apreciarse en la figura 1 [11].

Siguiendo este modelo, hemos construido un algoritmo cultural para optimización multiobjetivo, tratando de aprovechar las cualidades que se le atribuyen.

IV. Técnica Propuesta

Esta técnica es un algoritmo cultural con programación evolutiva (o CAEP, *Cultural Algorithm with Evolutionary Programming* [12]) El pseudo-código se muestra en el Algoritmo 1, donde se ven claramente las similitudes con la programación evolutiva tradicional [13], y también están los pasos donde se incluye el espacio de creencias.

El problema que se trata de resolver tiene n variables de decisión y k funciones objetivo. La población consiste de un conjunto de individuos, cada uno de los cuales representa una posible solución al problema. Cada individuo contiene las n variables de decisión, las cuales no están codificadas, por tratarse de un algoritmo basado en la programación evolutiva. La población es inicializada con p individuos generados aleatoriamente, mediante una distribución uniforme dentro de los intervalos para cada variable dados como datos de entrada. El fichero externo que se menciona

Algoritmo 1 Estructura básica del algoritmo cultural multiobjetivo

Generar la población inicial de tamaño p
Evaluar la población inicial
Inicializar el espacio de creencias
Repetir
  Aplicar el operador de mutación
    para generar p hijos
    (ahora hay $2p$ individuos en la población)
  Evaluar cada hijo
  Realizar los torneos binarios, eligiendo
    aleatoriamente c contrincantes para cada
    individuo. Las decisiones de los torneos
    estarán influidas por la información
    almacenada en el espacio de creencias.
  Seleccionar los p individuos con mayor
    número de victorias en los torneos,
    para formar la población de la
    siguiente generación
  Agregar los nuevos individuos no dominados
    al fichero externo
  Actualizar el espacio de creencias con los
    individuos aceptados
Mientras no se cumpla la condición de finalización

en el Algoritmo 1 es una población secundaria donde se almacenan los individuos no dominados. Este fichero tiene la finalidad de no perder las soluciones no dominadas encontradas a lo largo del proceso evolutivo, y contendrá el resultado final que se le presentará al tomador de decisiones. Este fichero externo tiene un tamaño máximo q, que corresponde con el número de soluciones que el tomador de decisiones desea que se le presenten como salida del algoritmo. A continuación se detalla la estructura del espacio de creencias, así como el resto de los pasos del algoritmo.

A. Estructura del Espacio de Creencias

El espacio de creencias tiene dos partes: la parte normativa fenotípica, y una rejilla, que se usa para enfatizar la generación de soluciones no dominadas distribuidas uniformemente a lo largo del frente de Pareto. Esta rejilla es una variación de la propuesta por Knowles y Corne [14].

La parte del conocimiento normativo fenotípico contiene únicamente los límites inferior y superior, l_{fi} y u_{fi}, de los intervalos para cada función objetivo ($i = 1, \ldots, k$) dentro de los cuales se construirá una rejilla, que se usará para ubicar cada solución no dominada en una especie de sistema coordenado, donde los valores de las funciones objetivo se utilizan para ubicar cada solución (ver figura 2).

l_{f1}	u_{f1}	l_{f2}	u_{f2}	$\cdots$	l_{fk}	u_{fk}

Fig. 2. Parte normativa fenotípica

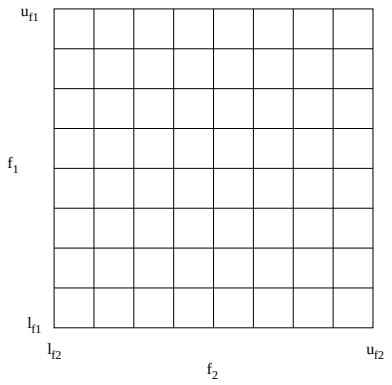

Fig. 3. Rejilla en el espacio de creencias para un problema con dos funciones objetivo. En este caso, $s_1 = s_2 = 8$ (ocho sub-intervalos en cada dimensión).

Teniendo esos intervalos, sólo se requiere conocer el número de sub-intervalos iguales en los que se dividirán cada uno (s_i, con $i = 1, \ldots, k$), para poder construir la rejilla en el espacio fenotípico. Un ejemplo de esta rejilla se muestra en la figura 3.

Como resultado, se tendrán $s_1 s_2 \ldots s_k$ celdas, que serán todas de las mismas dimensiones. Los valores s_i son parámetros de entrada del algoritmo. Para cada celda se almacena la cuenta de los individuos no dominados del fichero externo que estén dentro de cada una. Esto es útil para distribuir adecuadamente las soluciones no dominadas entre las celdas, evitando que se agrupen todos en una zona única del frente de Pareto.

B. Inicialización del Espacio de Creencias

Para inicializar el espacio de creencias es necesario que ya exista una población inicial, porque se utilizarán los individuos no dominados de esa población (puede demostrarse que toda población de tamaño diferente de cero contiene al menos un individuo no dominado [15]).

B.1 Inicialización de la Parte Normativa Fenotípica

La inicialización de la parte normativa fenotípica del espacio de creencias consiste en encontrar los valores extremos de cada función objetivo que se encuentren en los individuos no dominados de la población inicial. Esos extremos se almacenarán en l_{fi} y u_{fi}, para ubicar la rejilla en la región donde se encuentran los individuos no dominados conocidos hasta el momento.

B.2 Inicialización de la Rejilla

La rejilla se crea tomando como intervalos los valores almacenados en la parte normativa fenotípica, y se divide utilizando los parámetros de entrada s_i. Los contadores de los individuos no dominados dentro de cada celda se inicializan con cero.

C. Actualización del Espacio de Creencias

La rejilla del espacio de creencias se actualiza a cada generación, mientras que la parte normativa fenotípica se actualiza a cada $g_{normativa}$ generaciones, siendo $g_{normativa}$ un parámetro de entrada del algoritmo.

C.1 Actualización de la Rejilla

Para actualizar la rejilla simplemente se incrementan los contadores de los individuos no dominados con todos los individuos recién agregados al fichero externo durante la generación actual. La actualización de la rejilla es bastante simple, y esa es la razón por la que se ejecuta a cada generación. Para la actualización de esta parte del espacio de creencias, la función de aceptación utiliza la población del fichero externo, y elige únicamente a los individuos nuevos en esa población.

C.2 Actualización de la Parte Normativa Fenotípica

La actualización de la parte normativa fenotípica no se realiza a cada generación, porque implica una reconstrucción de la rejilla y, por tanto, el hacer esto a cada generación impactaría el costo computacional del algoritmo. Para su realización también se utiliza la población del fichero externo.

Nuevamente, como en la inicialización de esta parte normativa fenotípica, es necesario identificar los valores extremos en cada función objetivo de los individuos que se encuentran en el fichero externo. Esos valores se almacenan en l_{fi} y u_{fi} para $i = 1, \ldots, k$. Con los nuevos valores, se reconstruye la rejilla, que abarcará todo el frente de Pareto actual (figura 4) y que posiblemente se habría extendido más allá de los límites de la rejilla anterior.

Después de reinicializar la rejilla, todos los contadores estarán en cero, y es necesario agregar todos los individuos del fichero externo al contador de su celda correspondiente, para que el espacio de creencias esté listo nuevamente para su uso.

D. Mutación

La información almacenada en el espacio de creencias pertenecen al espacio fenotípico de nuestro problema, por lo que es difícil aplicarla

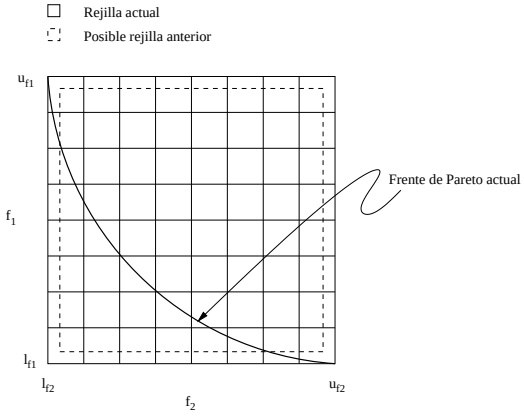

Fig. 4. La actualización de la parte normativa fenotípica del espacio de creencias tiene como finalidad que todo el frente de Pareto actual esté justo dentro de la rejilla.

a la autoadaptación de la mutación (que se realiza en el espacio genotípico). Dado este inconveniente, dejamos a los parámetros de la mutación como parámetros de entrada del algoritmo, a ser definidos por el usuario.

La mutación Gaussiana para nuestro algoritmo, entonces, sigue la siguiente expresión:

$$x_i' = x_i + N(0, \sigma)$$

donde: x_i es la variable i del individuo x, x_i' es la variable i del nuevo individuo x' resultante de la mutación, y $N(\mu, \sigma)$ es una variable aleatoria con una distribución normal con media μ y desviación estándar σ. Para este algoritmo, μ siempre será cero, y σ es un parámetro dado por el usuario. La mutación se realiza para $i = 1, \ldots, n$, y opera sobre los p individuos de la población principal, por lo que al final de este proceso se tendrá una población de tamaño $2p$.

E. Selección por Torneo

La selección por torneo se efectúa considerando a la población principal de tamaño $2p$. Cada individuo se enfrentará contra otros c individuos, elegidos al azar de la población principal. Las reglas para el torneo son:

1. Si un individuo domina al otro, gana el individuo no dominado.
2. Si no son comparables, o sus valores de las funciones objetivo son iguales, entonces:

a) Si ambos están dentro de la rejilla del espacio de creencias, gana el que se encuentre en una celda menos poblada (según el contador de las celdas).

b) Si alguno cae fuera de la rejilla, gana el que esté fuera.

La primera regla es clara, estamos prefiriendo individuos no dominados para acercarnos al verdadero frente de Pareto. Luego, es en el segundo

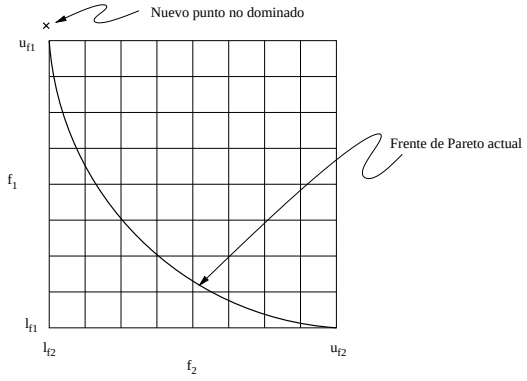

Fig. 5. Torneo cuando un punto está fuera de la rejilla. El punto encontrado es un nuevo extremo del frente de Pareto actual, y es importante conservarlo.

punto donde se aprecia la influencia del espacio de creencias en la decisión del torneo. El primer inciso del segundo punto tiene como finalidad distribuir las soluciones de manera equitativa entre las celdas, y construir un frente de Pareto más uniforme. Finalmente, si se presenta el caso del segundo inciso, significa que hemos encontrado una solución que está más allá del frente de Pareto conocido. Dado que esta solución generará una nueva porción del frente de Pareto, es importante conservarla. En la figura 5 se muestra el caso en el que un individuo está fuera de la rejilla.

Una vez concluidos los torneos, se selecciona a los individuos con un número mayor de victorias para pasar a la siguiente generación. Puede verse que estos torneos son similares a los del NPGA [16], donde se compara contra una fracción de la población. La diferencia es que en el NPGA se jerarquiza contra toda esa fracción de la población, mientras que en este algoritmo la comparación se hace uno a uno, contabilizando el número de victorias (como se hace comúnmente en la programación evolutiva).

F. Adición de Individuos al Fichero Externo

El fichero externo debe contener únicamente individuos no dominados, sin repetirlos. Para agregar individuos a este fichero, se tienen las siguientes reglas:

1. Si el individuo que se pretende agregar es dominado por algún individuo del fichero externo, entonces el individuo no se debe agregar.
2. Si el individuo que se pretende agregar domina a algún individuo del fichero externo, entonces se introduce en su lugar, pero continúa comparándose contra todos los demás. Si el mismo individuo, ya agregado, dominara a algún otro, éste (el dominado) es eliminado del fichero externo.
3. Si el individuo que se pretende agregar no es dominado ni domina a ningún otro en el fichero externo, y el tamaño actual de este fichero externo es menor que su tamaño máximo q, entonces se puede agregar el nuevo individuo al final.
4. Si el individuo que se pretende agregar no es dominado ni domina a ningún otro en el fichero externo, y el tamaño actual de este fichero externo es su tamaño máximo q, entonces se busca algún individuo del fichero externo cuya celda contenga más individuos que la celda a la que pertenece el individuo que se pretende agregar, y se reemplaza el individuo anterior con el nuevo. Con esto se obtiene una mejor distribución de los individuos no dominados entre las celdas.

Aunque el fichero externo esté lleno, su contenido seguirá cambiando durante la ejecución para obtener una mejor distribución. Al final, este fichero externo tendrá el frente de Pareto que se mostrará al tomador de decisiones, quien elegirá una de las soluciones para utilizarla en el problema que se quiere resolver.

V. Comparación de Resultados

Para validar la técnica propuesta en este artículo, se utilizó un conjunto de funciones de prueba, sugerido por Coello en [2]. La primera función, POM1, fue propuesta por Schaffer [17]; POM2 es el segundo problema multiobjetivo de Fonseca [18]; el tercer problema fue propuesto por Poloni en [19]. Todas estas funciones tienen niveles de dificultad variables. Hay problemas con frentes de Pareto cóncavos o convexos, contínuos o discontinuos, con dos o tres funciones objetivo, con lo que se puede apreciar el funcionamiento del algoritmo bajo distintas condiciones. Las funciones de prueba se muestran a continuación.

POM1. Minimizar:

$$\vec{f}(x) = (f_1(x), f_2(x))$$

donde:

$$\begin{aligned} f_1(x) &= x^2 \\ f_2(x) &= (x-2)^2 \end{aligned}$$

y $-10^5 \leq x \leq 10^5$.

POM2. Minimizar:

$$\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}))$$

donde:

$$f_1(\vec{x}) = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right)$$

$$f_2(\vec{x}) = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right)$$

y $-4 \leq x_i \leq 4$ $(i = 1, 2, 3)$.

POM3. Maximizar:

$$\vec{f}(\vec{x}) = (f_1(x, y), f_2(x, y))$$

donde:

$$\begin{aligned} f_1(x, y) &= -[1 + (A_1 - B_1)^2 + (A_2 - B_2)^2] \\ f_2(x, y) &= -[(x + 3)^2 + (y + 1)^2] \end{aligned}$$

con:

$$\begin{aligned} A_1 &= 0{,}5 \sin 1 - 2 \cos 1 + \sin 2 - 1{,}5 \cos 2 \\ A_2 &= 1{,}5 \sin 1 - \cos 1 + 2 \sin 2 - 0{,}5 \cos 2 \\ B_1 &= 0{,}5 \sin x - 2 \cos x + \sin y - 1{,}5 \cos y \\ B_2 &= 1{,}5 \sin x - \cos x + 2 \sin y - 0{,}5 \cos y \end{aligned}$$

y $-\pi \leq x, y \leq \pi$.

Los parámetros que se utilizaron para nuestro CAEP son los siguientes: $p = 6$, $q = 100$, $G_{máx} = 35,000$, $g_{normativa} = 20$, $s_i = 10$ $(i = 1, \ldots, k)$, $c = \frac{p}{2} = 5$, $\sigma = 1$. El número de evaluaciones de las funciones objetivo durante una ejecución del algoritmo se calcula con $p(G_{máx} + 1)$. En este caso se llevaron a cabo 210,006 evaluaciones de las funciones objetivo en cada ejecución.

La comparación de resultados se hizo con respecto al NSGA-II [20], con los siguientes parámetros: tamaño de la población = 100, número máximo de generaciones = 2100, probabilidad de cruce = 0.9, probabilidad de mutación = $\frac{1}{n}$, parámetro SBX = 10 y parámetro de mutación = 100. Los valores para estos parámetros son sugeridos por Deb en el mismo código, excepto el número de generaciones, que se eligió para aproximar el número de evaluaciones de las funciones objetivo hechas por nuestra técnica; con estos parámetros, el NSGA-II realizó 210,100 evaluaciones de las funciones objetivo en cada ejecución.

En la tabla I se muestran las estadísticas de las métricas de tasa de error (ER) [15], distancia generacional (GD) [15] y espaciado (SP) [21] para 10 ejecuciones de cada algoritmo.

La figura 6 muestra el comportamiento promedio de cada algoritmo (CAEP y NSGA-II) en la primera función de prueba. A partir de los resultados mostrados en la tabla I puede verse que CAEP obtuvo en este caso, mejores valores promedio para las tres métricas utilizadas.

La figura 7 se muestra el comportamiento promedio de los 2 algoritmos en POM2. En este caso, nuestra técnica muestra tasas de error altas (alrededor del 90 %), mientras que el NSGA-II tiene tasas de error de 50 % aproximadamente. En la distancia generacional, ambos muestran valores bajos. También en el espaciado el NSGA-II muestra resultados ligeramente superiores a CAEP.

La figura 8 se muestra el comportamiento promedio de los 2 algoritmos en POM3. En este caso, el NSGA-II mostró una distancia generacional más baja que nuestra técnica, pero su tasa de error es mayor. Este problema tiene un verdadero Frente de Pareto discontinuo, por lo que la medida de espaciado puede no reflejar exactamente el desempeño de un algoritmo. En esta métrica, el NSGA-II obtuvo valores más pequeños que CAEP.

VI. Conclusiones y Trabajo Futuro

En este artículo hemos introducido lo que parece ser la primera propuesta para utilizar un algoritmo cultural para optimización evolutiva multiobjetivo basándose en el concepto de óptimo de Pareto. Hemos mostrado que el agregar un espacio de creencias a un algoritmo evolutivo puede resultar beneficioso y representa una alternativa viable para resolver problemas multiobjetivo. El pequeño estudio comparativo presentado en este artículo valida la viabilidad de la propuesta, aunque se requiere todavía un estudio más detallado donde se comparen resultados con respecto a otros algoritmos evolutivos multiobjetivo usando más funciones de prueba. Adicionalmente, existen todavía algunas limitantes de CAEP que deben analizarse. Por ejemplo, CAEP tiende a perder diversidad muy rápidamente en algunos problemas. Para lidiar con este problema, estamos trabajando actualmente en un esquema adicional para mantener diversidad que sea eficiente (computacionalmente hablando). Así mismo, también nos interesa mejorar el mecanismo utilizado para generar una distribución uniforme de soluciones no dominadas a lo largo del frente de Pareto, debido a que el mecanismo actual tiene todavía algunas deficiencias.

Agradecimientos

El primer autor agradece el apoyo brindado por el Consejo Nacional de Ciencia y Tecnología (CONACyT) a través de una beca para cursar estudios de posgrado en la sección de Computación del departamento de Ingeniería Eléctrica del CINVESTAV-IPN. El segundo autor agradece el apoyo brindado por CONACyT a través del proyecto 34201-A.

Referencias

[1] Carlos A. Coello Coello, "An Updated Survey of GA-Based Multiobjective Optimization Tech-

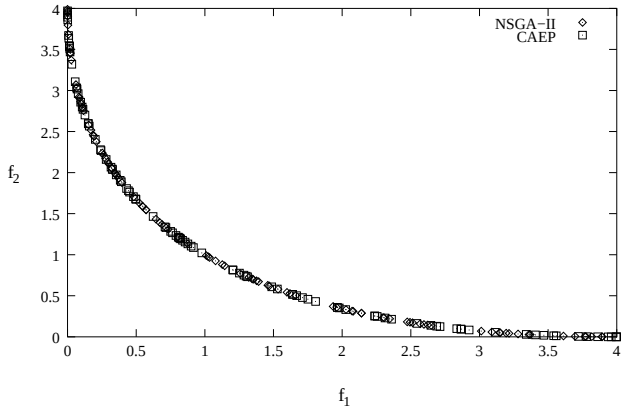

Fig. 6. Comparación de soluciones entre nuestro algoritmo (CAEP) y el NSGA-II para **POM 1**.

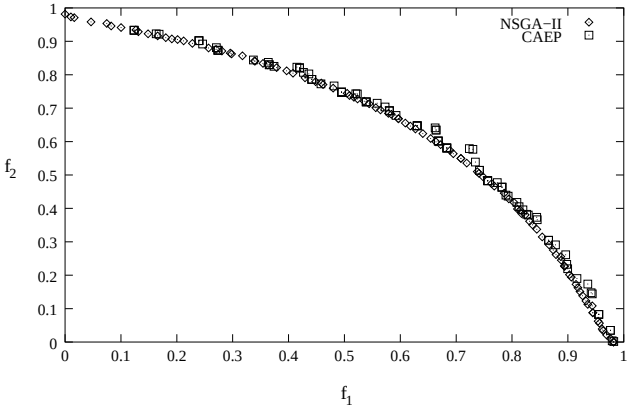

Fig. 7. Comparación de soluciones entre nuestro algoritmo (CAEP) y el NSGA-II para **POM 2**.

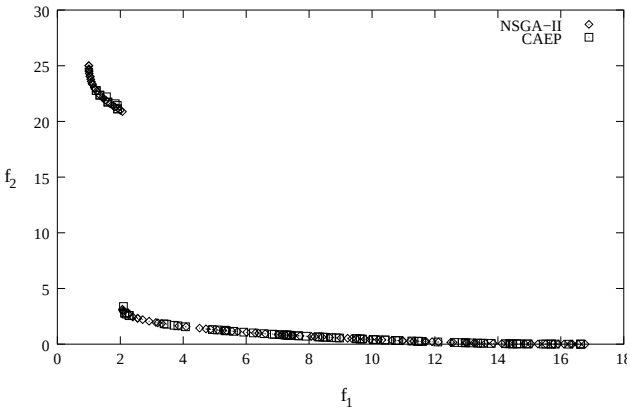

Fig. 8. Comparación de soluciones entre nuestro algoritmo (CAEP) y el NSGA-II para **POM 3**.

niques," *ACM Computing Surveys*, vol. 32, no. 2, pp. 109–143, June 2000.

[2] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, New York, 2002.

[3] Robert G. Reynolds, "An Introduction to Cultural Algorithms," in *Proceedings of the Third Annual Conference on Evolutionary Programming*, A. V. Sebald and L. J. Fogel, Eds., pp. 131–139. World Scientific, River Edge, New Jersey, 1994.

[4] Robert G. Reynolds, Zbigniew Michalewicz, and M. Cavaretta, "Using cultural algorithms for constraint handling in GENOCOP," in *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, Eds., pp. 298–305. MIT Press, Cambridge, Massachusetts, 1995.

[5] Xidong Jin and Robert G. Reynolds, "Using Knowledge-Based Evolutionary Computation to Solve Nonlinear Constraint Optimization Problems: a Cultural Algorithm Approach," in *1999 Congress on Evolutionary Computation*, Washington, D.C., July 1999, pp. 1672–1678, IEEE Service Center.

[6] Xidong Jin and Robert G. Reynolds, "Mining Knowledge in Large-Scale Databases Using Cultural Algo-

TABLA I

COMPARACIÓN DE RESULTADOS

POM		ER		GD		SP	
		CAEP	NSGA-II	CAEP	NSGA-II	CAEP	NSGA-II
1	Media	0.003	0.008059	0.000946	0.061548	0.037342	0.058298
	Desv. est.	0.006749	0.005818	0.000046	0.002711	0.002674	0.009385
	Min.	0	0	0.000899	0.058940	0.034057	0.043401
	Máx.	0.02	0.017857	0.001045	0.067989	0.042604	0.075381
2	Media	0.911	0.574	0.001103	0.000280	0.010931	0.007241
	Desv. est.	0.046774	0.034059	0.000171	0.000034	0.001040	0.000741
	Min.	0.84	0.53	0.000934	0.000230	0.009493	0.006015
	Máx.	0.98	0.63	0.001514	0.000349	0.012157	0.008287
3	Media	0.149	0.209	0.015321	0.001941	0.607229	0.092216
	Desv. est.	0.069194	0.041218	0.026480	0.000078	1.025953	0.008415
	Min.	0.06	0.15	0.001942	0.001822	0.065285	0.081569
	Máx.	0.28	0.28	0.076256	0.002107	2.80319	0.106568

rithms with Constraint Handling Mechanisms," in *Proceedings of the Congress on Evolutionary Computation 2000 (CEC'2000)*, Piscataway, New Jersey, July 2000, vol. 2, pp. 1498–1506, IEEE Service Center.

[7] Carlos A. Coello Coello and Ricardo Landa Becerra, "Adding Knowledge and Efficient Data Structures to Evolutionary Programming: A Cultural Algorithm for Constrained Optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, W.B. Langdon, E.Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A.C. Schultz, J. F. Miller, E. Burke, and N.Jonoska, Eds., San Francisco, California, July 2002, pp. 201–209, Morgan Kaufmann Publishers.

[8] A. C. Renfrew, "Dynamic Modeling in Archaeology: What, When, and Where?," in *Dynamical Modeling and the Study of Change in Archaelogy*, S. E. van der Leeuw, Ed. Edinburgh University Press, Edinburgh, Scotland, 1994.

[9] W. H. Durham, *Co-evolution: Genes, Culture, and Human Diversity*, Stanford University Press, Stanford, California, 1994.

[10] Benjamin Franklin and Marcel Bergerman, "Cultural algorithms: Concepts and experiments," in *Proc. of the 2000 Congress on Evolutionary Computation*, Piscataway, NJ, 2000, pp. 1245–1251, IEEE Service Center.

[11] Robert G. Reynolds, "Cultural algorithms: Theory and applications," in *New Ideas in Optimization*, David Corne, Marco Dorigo, and Fred Glover, Eds., pp. 367–377. McGraw-Hill, London, 1999.

[12] Chan-Jin Chung and Robert G. Reynolds, "CAEP: An Evolution-based Tool for Real-Valued Function Optimization using Cultural Algorithms," *Journal on Artificial Intelligence Tools*, vol. 7, no. 3, pp. 239–292, 1998.

[13] Lawrence J. Fogel, *Artificial Intelligence through Simulated Evolution. Forty Years of Evolutionary Programming*, John Wiley & Sons, Inc., New York, 1999.

[14] Joshua D. Knowles and David W. Corne, "Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy," *Evolutionary Computation*, vol. 8, no. 2, pp. 149–172, 2000.

[15] 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.

[16] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg, "A Niched Pareto Genetic Algorithm for Multiobjective Optimization," in *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, Piscataway, New Jersey, June 1994, vol. 1, pp. 82–87, IEEE Service Center.

[17] J. David Schaffer, "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms," in *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*. 1985, pp. 93–100, Lawrence Erlbaum.

[18] Carlos M. Fonseca and Peter J. Fleming, "Multiobjective Genetic Algorithms Made Easy: Selection, Sharing, and Mating Restriction," in *Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, Sheffield, UK, September 1995, pp. 42–52, IEE.

[19] Carlo Poloni, Giovanni Mosetti, and Stefano Contessi, "Multiobjective Optimization by GAs: Application to System and Component Design," in *Computational Methods in Applied Sciences '96: Invited Lectures and Special Technological Sessions of the Third ECCOMAS Computational Fluid Dynamics Conference and the Second ECCOMAS Conference on Numerical Methods in Engineering*, Chichester, 1996, pp. 258–264, Wiley.

[20] 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.

[21] J. R. Schott, "Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization," M.S. thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.