Aprendizaje incremental basado en población como buena alternativa al uso de algoritmos genéticos

Luis A. Cruz Prospero, Manuel Mejía Lavalle, José Ruiz Ascencio, Virna V. Vela Rincón

Centro Nacional de Investigación y Desarrollo Tecnológico,
Departamento de Ciencias Computacionales,
Cuernavaca, Morelos, México

{lacp, mlavalle, josera, viryvela}@cenidet.edu.mx

Resumen. En la actualidad han surgido nuevos modelos computacionales que intentan superar a los modelos clásicos de optimización, este es el caso de la Computación Evolutiva, la cual se ha popularizado por los Algoritmos Genéticos y sus diferentes variantes que prometen ser mejores. En este artículo analizaremos las bondades y/o deficiencias del Algoritmo Genético básico y del algoritmo de Aprendizaje Incremental Basado en Población, el cual es un algoritmo de estimación de distribuciones que forma parte del paradigma de la Computación Evolutiva. Se presenta un estudio comparativo de ambos algoritmos que permite establecer, a partir de la experimentación realizada con 7 funciones objetivo, que el algoritmo de Aprendizaje Incremental Basado en Población presenta ventaja significativa en tiempo de ejecución de todas las pruebas, así como en la precisión obtenida en 6 de las 7 funciones objetivo analizadas. Aunque esta ventaja ya había sido reportada, en este artículo se ha experimentado con funciones multimodal con dos incógnitas y en tres dimensiones, que en la actualidad son consideradas difíciles de resolver.

Palabras clave: Optimización de funciones, computación evolutiva, algoritmos genéticos, algoritmo de estimación de distribuciones, aprendizaje incremental basado en población.

Population-Based Incremental Learning as Good Alternative for Genetic Algorithms

Abstract. At present, new computational models that attempt to overcome to the classical optimization models have emerged, this is the case of Evolutionary Computation, which has been popularized by Genetic Algorithms and their different variants that promise to be better. In this article we will discuss the benefits and/or shortcomings of basic Genetic Algorithm and Population-Based Incremental Learning algorithm, which is an estimation of distributions

algorithm and it is part of Evolutionary Computation´s paradigm. A comparative study of both algorithms is presented, here it is established from the experimentation with 7 objective functions that the Population-Based Incremental Learning algorithm presents significant advantage on runtime of all experiments as well as the accuracy obtained in 6 of 7 objective functions analyzed. Although this advantage had already been reported, in this paper we have experimented with multimodal functions with two variables and three dimensions which are considered difficult to solve nowadays.

Keywords: Optimization functions, evolutionary computation, genetic algorithms, estimation of distribution algorithm, population-based incremental learning.

1. Introducción

Los algoritmos de optimización han sido extensamente desarrollados por un largo tiempo. La optimización consiste en tratar variaciones, usando información, de un concepto inicial con el fin de mejorarlo. Muchos problemas de la industria ingenieril, en particular de los sistemas de manufactura, son de naturaleza muy compleja y difícil de resolver por técnicas convencionales de optimización [1].

La Computación Evolutiva se ha encargado de brindar algoritmos que den solución a problemas considerados complejos. Estos algoritmos basados en metaheurísticas cuentan con poblaciones que representan un conjunto de soluciones fiables de un problema dado y que con el uso de técnicas estocásticas hacen que estas poblaciones representen mejores soluciones a través del tiempo [2].

Uno de estos algoritmos son los Algoritmos Genéticos (AG), que desde su surgimiento en 1962, han sido populares dentro de la comunidad científica, siendo una técnica de búsqueda aleatoria y optimización perteneciente al campo de la Inteligencia Artificial que ha ganado gran aceptación. Su principal fundamento se basa en la Teoría de Evolución de Darwin y se complementa con otros conceptos y teorías más recientes del campo de la genética [3].

Los AG están basados en el modelo presentado por John Holland, a raíz de la publicación en 1975 de su libro "*Adaptation in Natural and Artificial systems*", consisten en métodos adaptativos, usados en problemas de búsqueda y optimización de parámetros, basados en la reproducción sexual y el principio de supervivencia del más apto [4].

Una definición más completa es dada por Goldberg: "Los Algoritmos Genéticos son algoritmos de búsqueda basados en la mecánica de la selección natural y de la genética natural. Combinan la supervivencia del más apto entre estructuras de secuencias con un intercambio de información estructurado, aunque aleatorizado, para constituir así un algoritmo de búsqueda que tenga algo de las genialidades de las búsquedas humanas" [5].

Recientemente aparecieron los Algoritmos de Estimación de Distribuciones (EDA), una nueva familia de algoritmos evolutivos, desarrollados con el objetivo principal de

evitar la interrupción de soluciones parciales, teniendo como principal diferencia con los AG, que no existen las operaciones de cruce ni de mutación.

Dentro de la clasificación de los EDA se encuentra el algoritmo de Aprendizaje Incremental Basado en Población (PBIL por sus del inglés, *Population-Based Incremental Learning*), un algoritmo simple que utiliza un vector de probabilidad para generar la población, tomando en cuenta las evaluaciones más altas del vector [6].

Desde que PBIL fue propuesto por primera vez ha demostrado tener superioridad ante los AG [7], sin embargo en este trabajo se presenta el uso de PBIL en distintas pruebas realizadas en la optimización de funciones más complejas, como lo son las funciones multimodal con dos incógnitas.

Este artículo tiene como objetivo comparar el desempeño del AG y PBIL en un espacio de búsqueda amplio, con el fin de determinar la superioridad de uno de ellos en la maximización de funciones objetivo con alto grado de dificultad.

En la Sección 2 se brinda una descripción del problema que se pretende resolver a través de la Computación Evolutiva, así como un breve análisis del AG y PBIL. En la Sección 3 se describen cada una de las funciones objetivo que servirán para el análisis de los algoritmos, mientras que en la Sección 4 se discuten los resultados obtenidos en las pruebas. Finalmente en la Sección 5 se exponen las conclusiones.

2. Optimización por medio de computación evolutiva

2.1. Optimización de funciones

Los sistemas artificiales usados en ingeniería, a menudo son modelados a través de modelos matemáticos que cuentan con funciones y restricciones que definen ciertos procesos. Dichos modelos pueden contar con funciones de tipo lineales o no lineales, las cuales juegan un papel primordial en la descripción cuantitativa y el análisis de problemáticas inmersas en distintas áreas de la industria. Muchos problemas prácticos del mundo real, como la optimización combinacional, diseños de módulos y planeación de rutas, pueden ser transformados en funciones multimodal a través de del modelado y la simulación [8].

Generalmente es difícil resolver problemas sin una formulación matemática; un modelo matemático puede ser usado para representar problemas del mundo real con menos dificultad, participando en la optimización de parámetros que den solución a dichos problemas de forma satisfactoria.

La optimización de funciones es el proceso de encontrar un conjunto de posibles puntos dentro de una región factible de cierto espacio de búsqueda. Lo anterior es con el objetivo de que el valor resultante de la función se convierta, ya sea en el máximo o mínimo, en función del problema y los requerimientos.

Básicamente los problemas de optimización de funciones están compuestos por las siguientes tres partes [9]:

- Función objetivo. Es el modelo matemático del problema de la vida real a resolver.

- Conjunto de variables. Estas variables afectan directamente el valor de la función objetivo.
- Conjunto de restricciones. Las variables pueden tomar ciertos valores y no pueden tomar otros, estos dependerá de las restricciones definidas en el problema.

2.2. Algoritmos basados en computación evolutiva

Algoritmo Genético. Los AG son métodos que proveen de flexibilidad para la búsqueda efectiva de máximos o mínimos globales en problemas de optimización; esto quiere decir que se encargan de encontrar el mejor resultado posible dentro de un conjunto admisible de condiciones para lograr cierto objetivo. Estos algoritmos trabajan con aleatoriedad, usando la inteligencia a la manera de la Madre Naturaleza, probando eventos a lo largo de los años y la supervivencia del más apto.

Estos algoritmos trabajan sobre una población de individuos, donde cada uno de ellos representa una posible solución, cada individuo está compuesto de características, llamadas genes, estas soluciones son codificadas, generalmente por una cadena binaria. La longitud de la cadena dependerá del dominio de los parámetros, así como de la precisión requerida y es definida en (1).

$$2^{m_j-1} < \left(b_j - a_j\right) * 10^n \leq 2^{m_j} - 1 \tag{1}$$

En donde $[a_j, b_j]$ es el dominio establecido para la búsqueda; n es el número de dígitos de precisión requeridos después del punto decimal; mientras m_j serán los bits demandados por el problema.

Como en la naturaleza, la selección proporciona el mecanismo que conduce a mejores soluciones para sobrevivir, estas son asociadas a un valor de aptitud para reflejar qué tan buena es comparada con otras soluciones.

El valor de aptitud está arraigado a la función objetivo en la que se pretende realizar una búsqueda u optimización. Para realizar dicha evaluación es necesario convertir las cadenas binarias a números reales y esto se logra a partir de los siguientes pasos [1]:

- Convertir la cadena binaria a base 10 (Z):

$$z = \sum_{i=0}^{n} b_i 2^i \tag{2}$$

En donde n es la longitud de la cadena y b es valor del bit (0 o 1).

- Calcular el correspondiente número real x a través de:

$$x = a_j + z\left(\frac{b_j - a_j}{2^{m_j} - 1}\right) \tag{3}$$

Los valores más aptos son los que tiene mayores posibilidades de reproducirse en la siguiente generación. La cruza es la combinación del material genético entre los padres, intercambiando partes de las cadenas para producir un nuevo individuo. Por

último, la mutación causa cambios aleatorios en la cadena alterando los genes con una probabilidad establecida. Este ciclo es repetido hasta que se alcanza un criterio de paro, por ejemplo, un número definido de generaciones o cuando no haya cambios en la población [10].

Sintetizando, los pasos a seguir para implementar un AG son los siguientes:

- Generar una población aleatoriamente.
- Evaluar la aptitud de cada uno de los individuos y seleccionar los más aptos.
- Cruzar los individuos para generar una nueva población.
- Mutar aleatoriamente algunos individuos de la población.
- Repetir desde el paso dos hasta cumplir un criterio de paro.

Aprendizaje Incremental Basado en Población. Por otra parte tenemos a PBIL, este algoritmo es el más popular dentro de la clasificación EDA. El algoritmo estándar es una combinación de optimización evolutiva y aprendizaje competitivo [11]. Este algoritmo trabaja con un vector de probabilidades (V_P), el cual codifica una distribución de probabilidad que representa un prototipo de buenas soluciones y que se usa para generar una población de posibles soluciones (vectores binarios) en cada iteración [12]. Cada valor dentro del vector representa la probabilidad de que se pueda generar un uno o un cero, en la posición del gen correspondiente, inicialmente todos los valores tienen el 0.5 de probabilidad.

En cada generación, usando el V_P se obtienen los individuos, cada uno de éstos es evaluado y sólo los mejores son seleccionados. Los individuos seleccionados son usados para actualizar el V_P. La ejecución del algoritmo se detiene cuando el V_P converge, es decir, cuando todos sus elementos sean cero o uno [13].

La actualización del V_P está dado por (4).

$$V_P = V_P * (1.0 - LR) + Mejor\ individuo * L_R, \tag{4}$$

en donde:

- $V_P = Vector\ de\ probabilidad$
- $L_R = Taza\ de\ aprendizaje$

A continuación, se resumen los pasos del algoritmo PBIL para generar una solución:

- Inicializar todos los valores del V_P a 0.5.
- Generar un conjunto de individuos mediante el V_P.
- Evaluar la aptitud de cada uno de los individuos y ordenarlos de acuerdo a su aptitud.
- Actualizar el V_P de acuerdo al mejor individuo.
- El programa se repite desde el paso 2 y finaliza de acuerdo al criterio establecido.

Para actualizar el V_P se debe tomar en cuenta la Tasa de Aprendizaje (L_R), es de fundamental importancia en la implementación, ya que determina la rapidez y exactitud para la obtención de los resultados finales. En otras palabras, el L_R, es el factor de importancia que se le da al mejor individuo para la actualización del V_P.

Existen variantes para mejorar el rendimiento de este algoritmo, pero para este trabajo consideramos el uso del PBIL original.

3. Funciones implementadas y medidas de desempeño

Los problemas reales de optimización, ya sean de minimización, o maximización como es el caso que se abordará en este trabajo, comúnmente cuentan con la característica de no tener una única solución que satisfaga el problema [14]. Es por eso que se aborda, en el desarrollo de este trabajo, un conjunto de siete funciones multimodal [15], con el fin de localizar el máximo global que satisfaga a cada una de ellas por medio de la Computación Evolutiva.

- *Parabolic*. La función *Parabolic* resulta en una gráfica en forma de curva, la cual contiene un vértice y un eje de simetría que divide por la mitad, en forma vertical, a dicha curva dando como resultado dos mitades que son imágenes espejo, una de la otra. La representación en tres dimensiones de esta función se muestra en Fig. 1(a) y es definida por (5):

$$f1(x,y) = x^2 + y^2. \tag{5}$$

- Peaks. Es una función con dos variables. Obtenida por la traslación y ampliación de las distribuciones Gaussianas [16]. Tiene múltiples picos y valles, como se observa en Fig. 1(b). Esta función está definida por (6).

$$\begin{aligned} f2(x,y) = {} & 3(1-x)^2 e^{-(x^2+(y+1)^2)} - 10\left(\frac{x}{5} - x^3 - y^5\right) e^{-(x^2+y^2)} \\ & - \frac{1}{3} e^{-((x+1)^2+y^2)}. \end{aligned} \tag{6}$$

- *Himmelblaus*. La función modificada de *Himmelblaus* cuenta con 4 puntos que maximizan la función objetivo. Dicha función está definida en (7) y se muestra gráficamente en la Fig. 1(c).

$$f3(x,y) = 200 - (x^2 + y^2 - 11)^2 - (x + y^2 - 7)^2. \tag{7}$$

- *Equal peaks*. Como se muestra en la Fig. 1(d) la función *Equal peaks* está constituida por diferentes picos que tienen los mismos máximos [17], esta función se define por (8).

$$f4(x,y) = \cos(x)^2 + sen(y)^2. \tag{8}$$

- *Rastrigin*. Esta función fue propuesta por Rastrigin como una función bidimensional [18]. Esta función presentan una alta dificultad debido al amplio espacio de búsqueda y al alto número de máximos y mínimos locales. Mientras que función *Peaks* consiste en únicamente 3 picos, la función *Rastrigin* consiste en alrededor de 100 picos en un rango considerado [19], como se muestra en la Fig. 1(e). La función *Rastrigin* está definida por (9).

$$f5(x,y) = 20 + (x^2 - 10\cos(2\pi x) + y^2 - 10\cos(2\pi y)). \quad (9)$$

- *Circles*. La función *Circles* contiene múltiples círculos concéntricos (ver Fig. 1(f)). A diferencia de las funciones *Peaks* y *Rastrigin* en donde cada pico es un solo punto, la función *Circles* presenta líneas circulares de picos locales que contienen infinitos picos [20] y son definidos por (10).

$$f6(x,y) = (x^2 + y^2)^{0.25}((sen(50(x^2 + y^2)^{0.1}))^2 + 1.0). \quad (10)$$

- *Schaffer*. Esta función es considerada para las pruebas realizadas en este trabajo debido a la complejidad presentada al contar con diversos máximos y mínimos locales. Se muestra gráficamente en la Fig. 1(g) y es definida por (11).

$$f7(x,y) = 0.5 + \frac{(sen(\sqrt{x^2 + y^2}))^2 - 0.5}{(1 + 0.1(x^2+y^2))^2}. \quad (11)$$

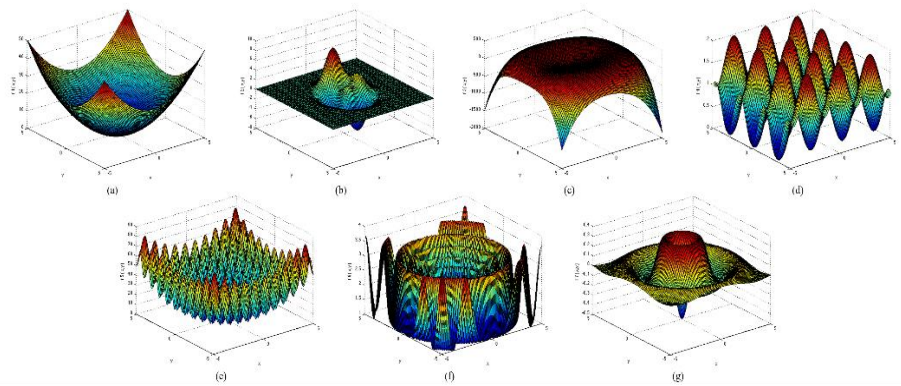


Fig. 1. Funciones objetivo

Con el fin de evaluar el desempeño de los algoritmos AG y PBIL se realizó la búsqueda de los valores que maximizan cada una de las funciones objetivo descritas anteriormente. En la Tabla 1 se muestra el óptimo global que maximiza cada función.

Estos datos servirán como base para la evaluación de cada algoritmo; la cual se realizará a través de cálculo de la precisión del resultado, así como el tiempo de ejecución.

4. Resultados y discusión

Para realizar la experimentación se implementó una plataforma propia de experimentación desarrollada en lenguaje Java, la cual facilitó la modificación de cada uno de los parámetros de ambos algoritmos. Este mismo *software* permitió observar el comportamiento de cada uno de los algoritmos en el proceso de maximización de las siete funciones objetivo establecidas anteriormente.

Con el objetivo de comprobar la eficacia de los AG y PBIL en la búsqueda de maximizar las funciones multimodal, se estableció un espacio de búsqueda en un rango de [-100,100]. Además los parámetros *x* e *y* de cada función se codificaron con cadenas binarias de 15 bits cada una, dando como resultado individuos con 30 genes.

Se plantearon los siguientes parámetros iniciales, para la realización de diferentes pruebas con cada una de las funciones objetivo:

- Número de individuos, 20, 100, 300 y 500.
- Número de generaciones, 20, 200 y 500.

Tabla 1. Óptimos globales

Función objetivo	Óptimo Global
f1(x,y)	20000
f2(x,y)	8.116
f3(x,y)	200
f4(x,y)	2
f5(x,y)	20000
f6(x,y)	23.309
f7(x,y)	0.831

Cada algoritmo cuenta con parámetros que se pueden someter a cambios para, de tal modo, obtener distintos comportamientos [2,7], por lo cual se aplicarán cada una de las pruebas con variaciones en éstos, en el caso de AG su variante se encuentra en la Probabilidad de Mutación (P_M) y Probabilidad de Cruce (P_C) y en PBIL se trabajará variando L_R.

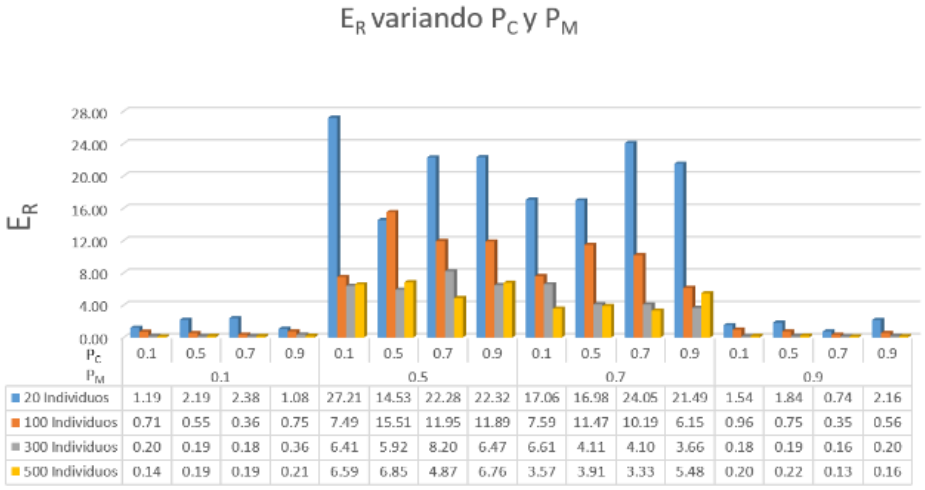
Para la evaluación de cada experimento se ejecutó el programa con los mismos parámetros, excepto la semilla, de forma independiente 5 veces y se reporta el promedio de ésto.

Con el fin de realizar un análisis más detallado de cada algoritmo, primeramente se mostrarán los resultados obtenidos de forma individual. En el caso del AG se experimentó variando la P_M y la P_C. En la Fig. 2 se ejemplifica, usando la función objetivo $f5(x,y)$, el índice de Error Relativo (E_R) obtenido por AG con cada combinación de ambos parámetros, adicionalmente se muestra la variación obtenida con diferente número de individuos.

De acuerdo a los resultados obtenidos en los experimentos, mostrados en la Fig. 2 se logra destacar que el AG presenta dificultades en la precisión de sus resultados cuando la P_M es igual o superior a 0.5, sin embargo al tener una P_M alta, cercana a 1, el E_R vuelve a disminuir; pero cabe destacar que al realizar esta acción se estaría perdiendo el esquema genético característico de los AG, debido a que una P_M cercana a 1 haría mutar a todos los individuos.

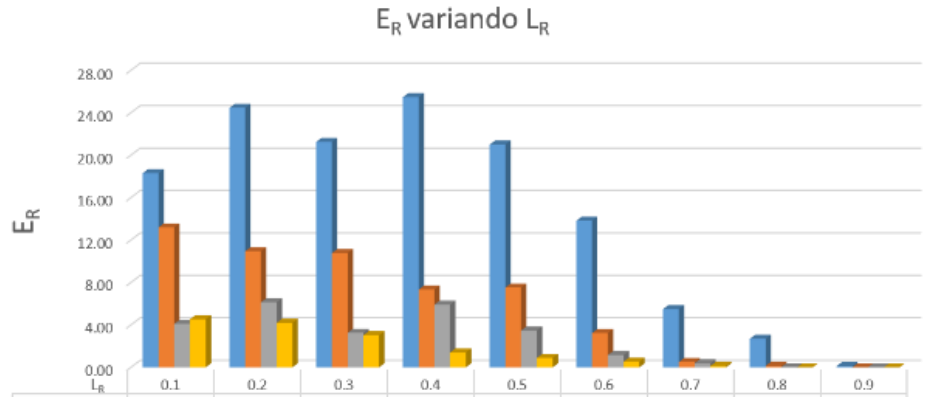
Por otro lado, se observa que al tener una P_C alta (más cercana a 1) el E_R disminuye, aumentando las posibilidades de éxito del algoritmo para localizar el óptimo global.

Pasando a PBIL, se usó como variable en los experimentos el L_R, se midió el desempeño del algoritmo calculando el E_R con diferente número de individuos. Los resultados obtenidos demuestran que al tener un L_R cercano a 1, los resultados tienden a mejorar rápidamente en cada iteración, como se muestra en la gráfica de la Fig. 3.



P_M	0.1				0.5				0.7				0.9			
P_C	0.1	0.5	0.7	0.9	0.1	0.5	0.7	0.9	0.1	0.5	0.7	0.9	0.1	0.5	0.7	0.9
20 Individuos	1.19	2.19	2.38	1.08	27.21	14.53	22.28	22.32	17.06	16.98	24.05	21.49	1.54	1.84	0.74	2.16
100 Individuos	0.71	0.55	0.36	0.75	7.49	15.51	11.95	11.89	7.59	11.47	10.19	6.15	0.96	0.75	0.35	0.56
300 Individuos	0.20	0.19	0.18	0.36	6.41	5.92	8.20	6.47	6.61	4.11	4.10	3.66	0.18	0.19	0.16	0.20
500 Individuos	0.14	0.19	0.19	0.21	6.59	6.85	4.87	6.76	3.57	3.91	3.33	5.48	0.20	0.22	0.13	0.16

Fig. 2. Error relativo obtenido con distintas P_C y P_M



L_R	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
20 Individuos	18.30	24.46	21.25	25.47	21.00	13.82	5.50	2.69	0.11
100 Individuos	13.20	10.94	10.77	7.33	7.52	3.23	0.50	0.12	0.00
300 Individuos	4.10	6.13	3.25	5.91	3.47	1.15	0.39	0.02	0.00
500 Individuos	4.53	4.20	3.03	1.41	0.89	0.54	0.13	0.00	0.00

Fig. 3. Error relativo obtenido con distintos L_R

De acuerdo a los resultados obtenidos en este trabajo, se externa una característica fundamental que hace que PBIL sobresalga a AG más allá de la precisión, y es el tiempo. Esto es debido a que, como se muestra en la Fig. 4, el tiempo de ejecución del algoritmo PBIL no tiende a tener un alto crecimiento al aumentar el número de individuos de la población, mientras tanto en el AG existe un aumento considerable,

como ejemplo, se puede observar que el tiempo de ejecución de PBIL con 500 individuos en la población es de 53.4 ms, mientras que AG le tomó un tiempo de 24,227 ms; tomándole al AG un tiempo que es 453 veces más que el tiempo empleado por PBIL.

Fig. 4. Tiempo de ejecución de AG vs PBIL

Otra característica interesante en el análisis de ambos algoritmos es la forma en que van evolucionando generación a generación debido a que esto es determinante al momento de obtener mejores resultados en menos generaciones. En la Fig. 5, con ayuda de la plataforma de experimentación, podemos observar como PBIL presenta una rápida estabilidad desde la generación 8, mientras que el AG, debido a su aleatoriedad en cada una de sus etapas presenta una alta variabilidad.

Con el fin de brindar una comparativa más amplia de ambos algoritmos en el proceso de maximización de las 7 funciones propuestas, en la Fig. 6 se plasma una gráfica que evidencia la precisión de AG y PBIL.

Para la gráfica de la Fig. 6 se usa como parámetros iniciales, en el caso de AG: P_C=0.9 y P_M=0.1; y en PBIL: L_R=0.9. Estos valores fueron asignados debido a que en general fueron los que obtuvieron mejores resultados en las pruebas.

Como se observa en la Fig. 6 el algoritmo PBIL presenta mejores resultados en la precisión de los funciones objetivo: $f1(x,y)$, $f4(x,y)$, $f5(x,y)$, $f6(x,y)$ y $f\,7(x,y)$. En estas funciones PBIL combina eficiencia con eficacia, sin embargo en $f2(x,y)$ y $f3(x,y)$ el AG resulta ser más preciso.

Finalmente en la Tabla 2 se muestran los mejores resultados obtenidos para cada función objetivo, así como los parámetros usados por cada algoritmo para alcanzar estos resultados.

Con la información de la Tabla 2 se reafirma la superioridad de PBIL en la optimización de funciones debido a su mayor precisión y sobre todo, el tiempo de ejecución con el que logra brindar resultados favorables, debido a que el tiempo promedio de ejecución de los resultados observados en la Tabla 2 es de 178,067.257 ms para AG, mientras que PBIL le tomó un tiempo promedio de 433.25 ms, obteniendo PBIL resultados 411 veces más rápido que el AG.

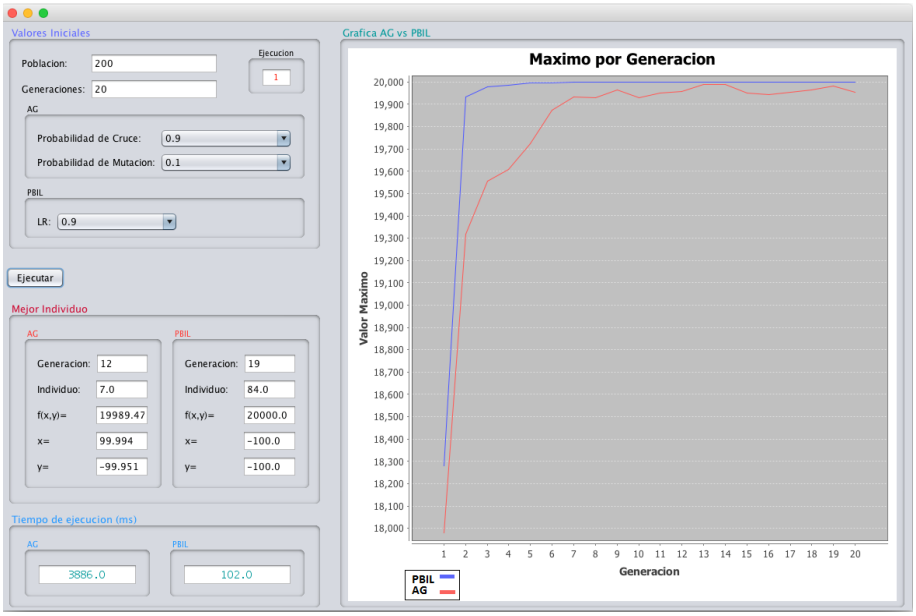


Fig. 5. Estabilidad de AG y PBIL

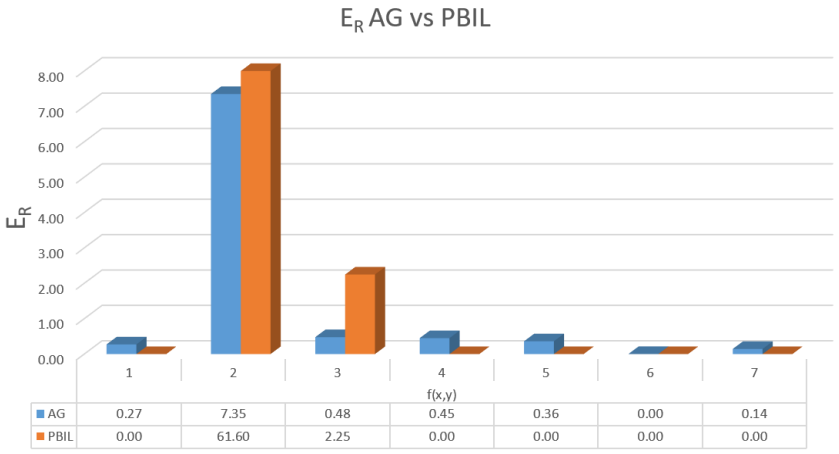


Fig. 6. *E*$_R$ por función (AG vs PBIL)

En precisión PBIL logra alcanzar el 100% en 5 de las 7 funciones evaluadas. Mientras que el AG destaca en f2(x,y) con un 26.28% de superioridad, esto es debido a que como se observa en la Fig. 2(b) existen máximos locales que se encuentran distantes de los demás puntos, con lo que debido a su aleatoriedad en la mutación el AG logra superar. A su vez en la Tabla 2 se aprecia que para AG es requerido un mayor número de individuos, así como de generaciones para alcanzar resultados más cercanos al óptimo, siendo una desventaja apreciable que repercute en el tiempo de ejecución.

Tabla 2. Mejores resultado AG y PBIL

Función	*f1*	*f2*	*f3*	*f4*	*f5*	*f6*	*f7*
			AG				
Generaciones	20	20	500	200	500	20	20
Individuos	500	500	500	500	500	300	500
P_C	0.50	0.90	0.90	0.50	0.50	0.50	0.90
P_M	0.10	0.10	0.10	0.10	0.90	0.10	0.90
E_R	0.17	0.66	0.06	0.02	0.11	0.00	0.00
Tiempo	23829.20	23912.00	466787.80	201536.20	502031.40	7287.60	21087.40
Precisión	99.83	99.34	99.94	99.98	99.89	100.00	100.00
			PBIL				
Generaciones	20	500	500	20	20	20	20
Individuos	100	500	500	500	100	100	300
L_R	0.90	0.70	0.80	0.90	0.90	0.80	0.90
E_R	0.00	26.94	0.64	0.00	0.00	0.00	0.00
Tiempo	11.40	1597.20	1325.00	38.40	12.00	12.80	36.00
Precisión	100.00	73.06	99.36	100.00	100.00	100.00	100.00

5. Conclusiones

La Computación Evolutiva en general tiene gran presencia en la actualidad debido a las aportaciones que ésta ha tenido en la resolución de problemas de optimización.

Desde un AG clásico hasta las diferentes variantes que existen actualmente, como lo es PBIL son herramientas poderosas, siempre y cuando se opte por el algoritmo ideal para cada problema.

En referencia a los resultados obtenidos se puede observar que en el caso de estudio de este trabajo, el cual fue la maximización de funciones multimodal, el algoritmo PBIL presenta una ventaja significativa en cada uno de los rubros evaluados en la fase de experimentación y presentados en la fase de resultados.

Se observa que PBIL presenta resultados más cercanos al óptimo en menor tiempo posible, es por eso que se puede concluir en este artículo que es una mejor opción en la optimización de funciones de maximización, sin embargo PBIL quizás podría ser mejorado si se incluye el mecanismo de mutación en los genes de los individuos: esto será motivo de investigación de nuestro trabajo futuro.

Aunque la Computación Evolutiva trabaja con datos aleatorios, PBIL presenta una ventaja al sustentarse en la probabilidad para garantizar más certeza en sus resultados.

Finalmente de acuerdo a los resultados de la experimentación se observó que para obtener mejores resultados en ambos algoritmos es recomendable usar una población

del doble del número de genes de los individuos. Además, al realizar los experimentos con PBIL se notó que un L_R de 0.9 funciona bien la mayoría de las veces, por lo que sugerimos probar de inicio con este valor.

Referencias

1. Guo, P. Wang, X. Han, Y.: The Enhanced Genetic Algorithms for the Optimization Design. In: 3rd International Conference on Biomedical Engineering and Informatics (BMEI 2010)
2. Khaparde, A.R., Raghuwanshi, M.M., Malik, L.G.: A New Distributed Differential Evolution Algorithm. In: International Conference on Computing, Communication and Automation (ICCCA) (2015)
3. Monzón, C.F., Mariño, R., Bogado, S.L. Validación de la implementación de la Función de Aptitud de un Algoritmo Genético aplicado a la sectorización. Universidad Nac. Del Nord. Comun. Científicas y Tecnológicas, pp. 7–10 (2005)
4. Withley, D.: A genetic algorithm tutorial. Stat. Comput., Vol. 4, No. 2, pp. 65–85 (1994)
5. Gestal, M.: Introducción a los algoritmos genéticos. (2010)
6. Rastegar, R., Hariri, A.: The Population-Based Incremental Learning Algorithm converges to local optima. Nuerocomputing, Vol. 69, No. 13-15, pp. 1772–1775 (2006)
7. Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep. CMU-CS-94-163 (1994)
8. Liu, X., Yang, Z., Pan, W.: An improved adaptive immune genetic algorithm based on information entropy. In: International Conference on Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information Integration (2015)
9. Patalia, T.P., Kulkarni, G.R.: Behavioral Analysis of Genetic Algorithm for Function Optimization. C. U. Shah College of Engineering & Technology, Surendranagar, India, (2010)
10. Rodríguez, R.P., Aguirre, A.H.: Un algoritmo de Estimación de Distribuciones para el problema de secuenciamineto en configuración jobshop flexible. (2005)
11. Mavrovouniotis, M., Yang, S.: Population-Based Incremental Learning with Immigrants Schemes in Changing Environments. IEEE Symposium Series on Computational Intelligence (2015)
12. Fernandez de Viana, I., Cordón, O., Alonso, S., Herrera, F.: La metahurística de optimización basada en colonias de hormigas: modelos y nuevos enfoques. Optim. Intel., pp. 261–314 (2004)
13. Gosling, T., Tsang, E.: Population Based Incremental Learning with Guided Mutation Versus Genetic Algorithms: Iterated Prisioners Dilemma. (2005)
14. Lee, J., Song, S., Yang, Y., Shim, H., Lee, H., Lee, K., Yoon, Y.: Multimodal Function Optimization Based on the Survival of the Fitness kind of the Evolution Strategy. Proceedings of the 29th Annual International Conference of the IEEE EMBS Cité Internationale, Lyon, France, August 23-26 (2007)
15. Hua, Q., Wu, B.: Peak Detection Method for Multimodal Function Optimization. Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, 19-22 August (2007)
16. Reutskiy, S.Y., Chen, C.S.: Approximation of multivariate functions and evaluation of particular solutions using Chebyshev polynomial and trigonometric basis functions. International Journal for Numerical Methods in Engineering, Vol. 67, No. 13, pp. 1811–1829 (2006)

17. Parsopoulos, K., Vrahatis, M.N.: On the computation of all global minimizers through particle swarm optimization. IEEE Transactions on Evolutionary Computation, Vol. 8, No. 3, pp. 211–224 (2004)
18. Törn, A., Zilinskas, A.: Global optimization. New York: Springer (1989)
19. Krishnanand, K.N., Ghose, D.: Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. Swarm Intell, Vol. 3, pp. 87–124 (2008)
20. Muller, S.D., Marchetto, J., Koumoutsakos, S.A.P.: Optimization based on bacterial chemotaxis. IEEE Transactions on Evolutionary Computation, Vol. 6, No. 6, pp. 16–29 (2002)