COMPARACIÓN DE ALGORITMOS EVOLUTIVOS Y BIO-INSPIRADOS EN PROBLEMAS DE OPTIMIZACIÓN CON RESTRICCIONES

B. López-Ramírez[a], E. Mezura-Montes[b]

[a] Departamento de Ing. en Sistemas Computacionales, Instituto Tecnológico Superior de Irapuato, Carr. Irapuato-Silao Km. 12.5, Irapuato, Gto., México, blopez@lania.edu.mx.
[b] Laboratorio Nacional de Informática Avanzada, Rébsamen 80, Centro, Xalapa, Veracruz, 91000, México, emezura@lania.edu.mx.

RESUMEN

Los algoritmos evolutivos y bio-inspirados agrupan a un conjunto de heurísticas que modelan procesos naturales, como la evolución de las especies y comportamientos sociales en animales o insectos, para resolver complejos problemas de búsqueda. Actualmente, estas técnicas se aplican ampliamente en diversas áreas, tales como la ingeniería, biología, medicina, electrónica, robótica, etc.

A pesar del éxito de dichos algoritmos, los estudios para evaluar cuál de ellos es el más conveniente, dependiendo las características del problema a resolver, son escasos.

Este trabajo presenta una comparación empírica de cuatro algoritmos de este tipo: El algoritmo genético, la estrategia evolutiva, la evolución diferencial y la optimización mediante cúmulos de partículas, todos utilizados en condiciones similares, para resolver el problema de optimización global con restricciones.

Se utilizó un conjunto de funciones de prueba encontrados en la literatura especializada. Los aspectos evaluados se dividen en 2 partes: (1) Resultados finales obtenidos (donde se analiza la calidad y consistencia de los mismos) y (2) analizar su comportamiento al converger.

INTRODUCCIÓN

Los algoritmos evolutivos (AE's) y bio-inspirados son algoritmos heurísticos aplicados para resolver problemas complejos de búsqueda, incluyendo problemas de optimización[2]. Para el resto del documento se utilizará el término AE's para agrupar a los algoritmos evolutivos y también a los bio-inspirados. La mayoría de las investigaciones para aplicar los AEs en problemas de optimización del mundo real, los cuales normalmente tienen restricciones en su modelo, se han enfocado en diseñar un mecanismo adecuado para el manejo de las restricciones adaptándolo a un algoritmo en particular. En la década pasada se propusieron nuevas heurísticas como la Evolución Diferencial (ED)[1] y Particle Swarm Optimización (PSO)[2] para resolver problemas de optimización con restricciones. El objetivo de esta investigación es analizar el comportamiento de diferentes algoritmos utilizando el mismo mecanismo para manejar restricciones y determinar el desempeño de un algoritmo específico (Algoritmo Genético (AG), Estrategias Evolutivas (EE), Cúmulo de partículas (PSO), Evolución Diferencial (ED)) con respecto a las características del problema a resolver. Este

experimento es un punto de partida de una serie de investigaciones que permitirán proveer de información sobre qué AE es más conveniente utilizar por un usuario que tiene un problema de optimización por resolver.
El documento está organizado de la siguiente forma: La Sección II, menciona algunas técnicas previas que motivaron el estudio. La Sección III, presenta la propuesta de estudio, la Sección IV, los resultados y discusiones. Finalmente, en la Sección V se exponen las conclusiones y sugerencias de investigación a futuro.

II. TRABAJO RELACIONADO

La técnica más socorrida para adaptar un AE para resolver problemas con restricciones es el uso de funciones de penalización[2], las cuales "castigan" a las soluciones no factibles (que no cumplen con las restricciones del problema) de manera que las que si las cumplan se vean favorecidas en el proceso de selección y supervivencia de un AE. La principal desventaja de las funciones de penalización es que se debe definir el grado de penalización, lo cual se ha demostrado es dependiente del problema a resolver. Deb[2] propuso un mecanismo alternativo a las funciones de penalización que es muy simple y no requiere de la definición de parámetros extra. Éste consiste de un conjunto de reglas simples basadas en factibilidad que a continuación se detallan:

1. Entre 2 soluciones factibles, la solución de valor más alto es la que gana.
2. Si una solución es factible y la otra es no factible, la solución factible gana.
3. Si ambas soluciones son no factibles, aquella con el valor más bajo en la suma de la violaciones de restricción gana.

Esta técnica ha sido aplicada con éxito, por ejemplo Mezura et al.[1], utilizaron ED en su variante *ED/rand/1/bin*, modificándola para que cada vector padre genere más de un vector hijo. Liang y Suganthan[3] modificaron el PSO en su versión Dynamic Multi-Swarm Optimizer para satisfacer mediante sub-cúmulos de partículas las restricciones del problema; además, usan un método clásico de optimización (Sequential Quadratic Programming) como buscador local.

III. PROPUESTA

En los métodos discutidos en la sección previa, el objetivo principal es modificar la estructura original de un AE para agregarle un mecanismo de manejo de restricciones. Consideremos importante estudiar el comportamiento de los AE's en sus variantes originales sin mayor modificación, pero utilizando un mecanismo sencillo de manejo de restricciones común a todos ellos, con la intención de determinar de forma empírica la calidad y consistencia de los resultados finales. El propósito es conocer el desempeño de cada algoritmo durante el proceso de búsqueda y establecer alguna conexión entre él y las características del problema a resolver. Esta información será útil para los interesados en implementar AE's en la solución de problemas con restricciones, pues sabrán de antemano que algoritmo provee mejores resultados dependiendo de las características del problema dado.
Decidimos implementar 4 algoritmos en sus versiones más comunes: ED en la variante *ED/rand/1/bin*, un AG con representación real, una (μ+λ)-EE y el PSO en versión gbest. El seudo-código de cada uno se presenta en la figura 1.

IV. DISEÑO EXPERIMENTAL Y RESULTADOS

El experimento utiliza 4 funciones de prueba de la literatura especializada[4]. Las características de cada problema, en resumen, se presentan en la Tabla I, donde ρ es una métrica sugerida[4] para estimar el radio entre la región factible y el espacio de búsqueda y fue estimada de la siguiente manera: $\rho = |F|/|S|$, donde $|F|$ es el número de soluciones factibles y $|S|$ es el total de número de soluciones generadas aleatoriamente, $S = 1{,}000{,}000$ en este trabajo. Definimos dos experimentos para: (1) Determinar la calidad de cada algoritmo en sus resultados finales (considerando el mejor resultado encontrado) y su consistencia (el mejor valor de media y desviación estándar) y (2) analizar el comportamiento de convergencia de cada algoritmo
A continuación detallamos los parámetros de cada técnica: AG con representación real: tamaño de población=100, generaciones=3000, porcentaje de cruza=0.3, porcentaje de mutación=0.3, cruza aritmética simple, mutación uniforme, torneo de selección binario y reemplazo generacional.

A) Inicio
Generar una población aleatoria inicial con un tamaño = popsize
Evaluar cada individuo de la población
Para i = 1 hasta MaxGenerations
j=0
Mientras(j <= popsize)
El vector padre "j" genera un vector hijo "j´" variante *ED/rand/1/bin*
Evaluar el vector hijo "j´"
Comparar los vectores padre "j" e hijo "j" con **reglas de factibilidad**
El mejor permanecerá para la siguiente generación
j = j + 1
Fin de Mientras
Fin de Para
Fin

B) Inicio
Generar una población aleatoria inicial con un tamaño = popsize
Evaluar cada individuo de la población
Para i=1 hasta MaxGenerations
j=0
Mientras(j <= popsize)
Seleccionar 2 individuos de *popsize* utilizando **reglas de factibilidad**
Aplicar cruza aritmética simple para crear dos hijos "j" y "j + 1"
Aplicar mutación a "j" y "j+ 1"
Evaluar hijos "j" y "j +1"
j = j + 2
Fin de Mientras
Reemplazo generacional. (Los hijos permanecen y los padres son eliminados).
Fin de Para
Fin

C) Inicio
Generar una población inicial con tamaño = μ
Evaluar cada individuo en la población
Para i = 1 hasta MaxGenerations
j=0
Mientras (j <= λ)
Selecciona 3 individuos aleatoriamente en μ
Aplicar recombinación para crear un hijo "j"
Aplicar mutación Gaussiana a "j"
Evaluar el hijo "j"
j = j + 1
Fin de Mientras
Escoger de entre los $\mu + \lambda$ individuos a los μ
Mejores **usando reglas de factibilidad**
Fin de Para
Fin

D) Inicio
Generar una población aleatoria inicial de tamaño = *parsize*
Evaluar cada individuo en el cúmulo
Para i = 1 hasta MaxGenerations
Seleccionar el líder del cúmulo **con reglas de factibilidad**
j=0
Mientras (j <= *parsize*)
Aplicar formula de vuelo a la partícula "j"
Aplicando fórmula con factor de inercia.
Evaluar la nueva posición de la partícula Actualizar el valor pbest (memoria) de la partícula "j" usando las **reglas de factibilidad**
j = j + 1
Fin de Mientras
Fin de Para
Fin

Figura 1. Seudo-códigos de los 4 algoritmos implementados. A) ED, B) AG, C) EE y D) PSO.

EE versión (100 + 300)-EE: generaciones=750, recombinación panmítica intermedia aplicados a los parámetros de la técnica y a las variables de decisión, mutación Gaussiana no-correlacional. ED en su versión *ED/rand/1/bin*: tamaño de población=100, generaciones=3000, CR=aleatoriamente entre (0.8, 1.0), F= generado aleatoriamente entre (0.3, 0.9). PSO en su versión global best: parsize=40, generaciones=7500, fórmula del vuelo con peso de inercia, C1=3.9, C2=0.1, W=1.5. En la Tabla II se indican las medidas estadísticas (mejor, media y desviación estándar) de las 30 corridas en las 4 funciones de prueba. La calidad la determina la medida del "mejor" cuando su valor es igual al mejor reportado, mientras la consistencia se mide por medio de la desviación estándar y la media de las 30 corridas.

Problema	n	Tipo de función	ρ	DL	DN	a
g02	20	No lineal	99.9971%	0	2	1
g04	5	cuadrática	52.1230%	0	6	2
g10	8	lineal	0.0010 %	3	3	6
g18	9	cuadrática	0.0000 %	0	13	6

Tabla I. Características de los problemas de prueba. n es el número de variables de decisión, ρ es el radio estimado entre la región factible y el espacio de búsqueda, DL es el número de funciones de restricción de desigualdad lineal, DN es el número de funciones de restricción de desigualdad no lineal, "a" es el número restricciones activas.

En todos los problemas ED obtiene los resultados más competitivos con base en consistencia y calidad, seguido por EE y el AG. PSO no alcanza soluciones factibles en el problema g18 (que es el que mayor número de restricciones tiene) y es el más inconsistente en las restantes funciones de prueba. En el problema con función objetivo lineal (g10) sólo ED y EE alcanzaron el óptimo de manera consistente. En problemas con función objetivo no lineal (g02, g04 y g18) el mejor fue ED. Es importante notar que en el problema con mayor dimensionalidad (g02) ED fue el más competitivo. AG y PSO se estancaron en soluciones óptimas locales y no pudieron alcanzar el óptimo en ninguno de los problemas. Como resultado general de este primer experimento podemos decir que ED es el algoritmo más robusto aún en problemas con alta dimensionalidad incluso con una función objetivo no lineal. EE es muy competitiva, pero se ve afectada por la alta dimensionalidad. AG encuentra soluciones factibles de manera consistente, pero se queda atrapado en soluciones óptimas locales. Por último, PSO es muy inconsistente e incluso no encuentra soluciones factibles en problemas con 13 restricciones de desigualdad no lineal.

Problema y mejor solución conocida		**Comparación de cuatro AEs**			
	Estadísticas	**ED**	**AG**	**EE**	**PSO**
g02 −0.803619	Mejor	**−0.803619**	−0.768219	−0.803535	−0.59166
	Media	**−0.797815**	−0.744961	−0.778222	−0.5111
	Desv. Est	**−0.009048**	−0.017392	−0.022281	−0.02626
g04 -30665.54	Mejor	**−30665.54**	30654.50	**−30665.54**	−30637.40
	Media	**−30665.54**	− 30582.50	**−30665.54**	−30613.20
	Desv. Est.	**0**	− 40.8	**0**	− 11.99
g10 7049.248	Mejor	**7049.248**	7226.913	**7049.248**	9700.121
	Media	**7049.248**	9469.82	**7049.248**	10524.58
	Desv. Est.	**0**	1478.97	**0**	435.067
g18 −0.866025	Mejor	**−0.866025**	−0.852457	**−0.866025**	-
	Media	**−0.866025**	−0.452928	−0.866024	-
	Desv. Est.	**0**	−1.790335	**0**	-

Tabla II. Resultados estadísticos de las 4 funciones de prueba en 30 ejecuciones independientes. El resultado en negritas significa que alcanza el mejor reportado en la literatura."-" Significa que no se encontraron soluciones factibles.

En nuestro segundo experimento, mostramos las gráficas de convergencia (número de evaluaciones contra mejor valor de la función objetivo, de la corrida en el valor de la mediana de las 30 realizadas) de los cuatro algoritmos en los cuatro problemas de prueba, todas en la Figura 2.

Cada curva inicia cuando la primera solución factible es encontrada. Podemos notar que siempre ED es de los primeros en encontrar soluciones factibles y siempre converge de manera consistente y rápida (curvas con triángulos). EE tiene un comportamiento similar, pero su convergencia es un poco más lenta (curvas con círculos). Por último AG y PSO, aunque encuentran soluciones factibles al inicio del proceso, tienen problemas para mejorar las soluciones previamente encontradas. PSO no aparece para g18 pues no encontró soluciones factibles.

Como observaciones generales de ambos experimentos tenemos las siguientes:

1. ED fue el algoritmo que obtuvo resultados de calidad de manera consistente y con una convergencia rápida en todos los problemas, sin importar las características de estos últimos.
2. EE obtuvo resultados muy competitivos, pero con una convergencia más lenta que ED y fue afectada en presencia de una alta dimensionalidad.
3. AG pudo obtener soluciones factibles en todos los problemas, pero se estancó en óptimos locales.
4. PSO mostró los resultados más pobres y en un problema no pudo encontrar soluciones factibles.

IV. CONCLUSIONES Y TRABAJO FUTURO

En este artículo se presentó un comparativo empírico de cuatro algoritmos evolutivos y bio-inspirados para resolver problemas de optimización con restricciones. Utilizando cuatro funciones de prueba encontradas en la literatura especializada se determinó que el algoritmo más robusto fue ED, que pudo resolver las cuatro funciones de prueba con el mejor desempeño.

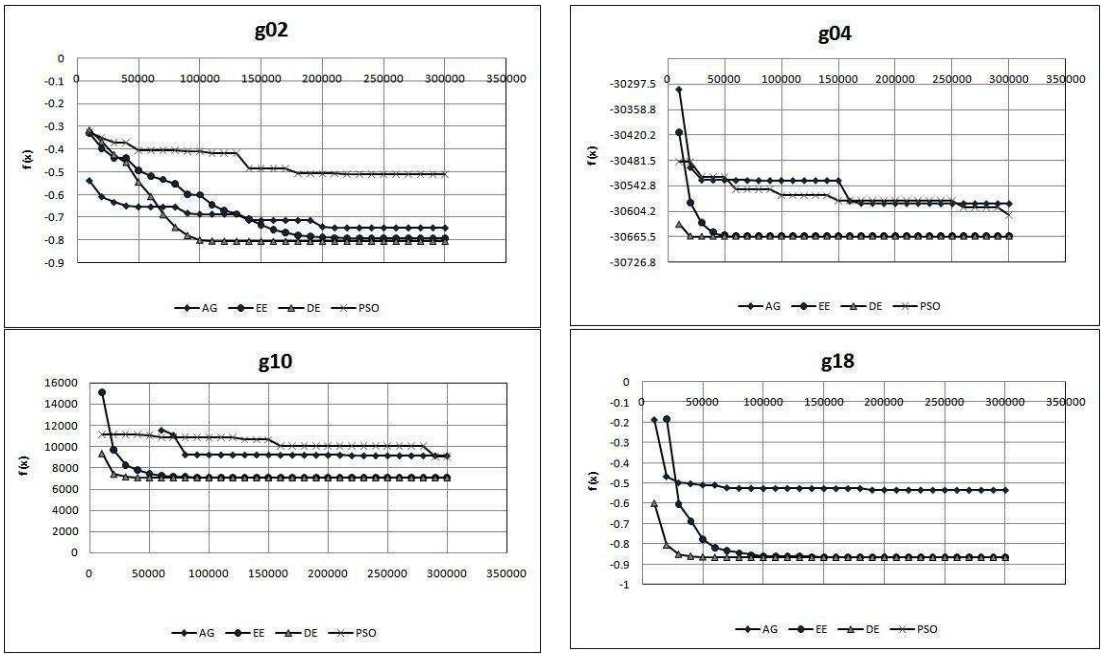

Figura 2. Gráficas de convergencia de los 4 algoritmos en los 4 problemas de prueba.

EE fue muy competitiva pero fue afectada por la dimensionalidad alta. AG es capaz de generar soluciones factibles más no de mejorarlas hasta aproximar el óptimo global. Finalmente, PSO tuvo problemas de convergencia prematura (a óptimos locales) y no pudo obtener soluciones factibles en uno de los cuatro problemas de prueba. Como futuros caminos de investigación se plantean los siguientes puntos: (1) El análisis a detalle de la capacidad de ED para resolver de manera competitiva los problemas con diferentes características y (2) Estudiar los motivos del mal desempeño del PSO en este estudio.

REFERENCIAS

1. E. Mezura-Montes, J. Velázquez-Reyes, and C.A. Coello Coello. Modified differential evolution for constrained optimization. In *Proceedings of the Congress on Evolutionary Computation 2006 (CEC'2006)*, Vancouver, BC, Canada, July 2006, IEEE Service Center, pages 333–339.
2. Kalyanmoy Deb. An Efficient Constraint Handling Method for Genetic Algorithms, *Computer Methods in Applied Mechanics and Engineering*, 186(2/4), 2000, pages 311—338.
3. J. J. Liang and P.N. Suganthan. Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism. In *Proceedings of the Congress on Evolutionary Computation 2006 (CEC'2006)*, Vancouver, BC,Canada, July 2006., IEEE Service Center, pages 316–323.
4. J. J. Liang, T.P. Runarsson, E. Mezura-Montes, P.N. Suganthan M. Clerc, C.A. Coello Coello, and K. Deb. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. Technical Report available at: http://www.lania.mx/~emezura/publications/publications-2006.html, March, 2006.

*** El primer autor agradece a CONACyT el apoyo para cursar estudios de Maestría en el LANIA. El segundo autor agradece el apoyo de CONACyT mediante el proyecto 52048-Y.**